

# Turtlebot2i simulation environment setup instructions

Author: Zuchen Zhang

If you don't want to go through this tedious work log file, you can download the Turtlebot2i simulation environment from my Github repository, and follow the instructions in README to use it. My Github repository address:

[https://github.com/Roboticist999/turtlebot2i\\_simulaiton\\_environment](https://github.com/Roboticist999/turtlebot2i_simulaiton_environment)

1. If you are using VMware and Gazebo crashes on start, try to uncheck Accelletation 3D Graphics in the setting of virtual machine display

<http://answers.gazebosim.org/question/13214/virtual-machine-not-launching-gazebo/>

I was able to launch Gazebo 7.1.0 on Ubuntu 16.04 by unchecking "Accelerate 3D Graphics" option in the virtual machine settings. Seems like the problem has to do with how VMWare handles the graphics acceleration.

answered Jul 10 '16  
m\_eltar 11 • 2 • 1

**Comments**

I tried your solution, and then it worked, the message "vmw\_ioctl\_command error Invalid argument" was gone. I have no idea why the "Accelerate 3D Graphics" has the impact on the issue. It just happens in Ubuntu 14 and higher version.  
buckmole ( Aug 27 '16 )

Thanks it solved my problem of not being able to open fetch robot in gazebo simulation! It's strange tho, cause I actually can other robots except for that particular robot.  
allenmon ( Oct 26 '16 )

I've unselected the "Accelerate 3D Graphics" option, however, now when trying to log into the guest account, i just get a blank screen. has this happened to anyone else?  
tabuzzy ( Jan 16 '17 )

[add a comment](#)

<https://kb.vmware.com/s/article/59146>

I'm using VMware workstation, and it works on my virtual machine. But by unchecking this, the gazebo simulation graphics could be not fluent.

**For Fusion:**

1. Shutdown the Virtual Machine.
2. From the VMware Fusion menu bar, select **Window > Virtual Machine Library**.
3. Select a virtual machine and click **Settings**.
4. In the **Settings** Window, in the **System Settings** section, select **Display**.
5. Uncheck **Accelerate 3D graphics**.

**For Workstation:**

1. Shutdown the virtual machine.
2. Select the virtual machine and select **VM > Settings**.
3. On the **Hardware** tab, select **Display**.
4. Uncheck **Accelerate 3D graphics**.
5. Click **OK**.

2. Do commands I marked green and yellow in  
[Full Build Instructions · Interbotix\\_turtlebot2i Wiki.pdf](#)

When doing RealSense package installation, I meet this problem:

```
turtlebot2i@ubuntu:~/librealsense$ ./scripts/patch-realsense-ubuntu-xenial.sh
bash: ./scripts/patch-realsense-ubuntu-xenial.sh: No such file or directory
turtlebot2i@ubuntu:~/librealsense$ █
```

And I find a explanation that says we can omit it

<https://github.com/IntelRealSense/librealsense/issues/2712>

3. Now if you run `roslaunch turtlebot2i_gazebo turtlebot2i_world.launch`, you will meet this problem:

```
[WARN] [1583684534.209884, 0.000000]: Controller Spawner couldn't find the expected controller_manager ROS interface.  
[turtlebot/controller_spawner-5] process has finished cleanly  
log file: /home/turtlebot2i/.ros/log/e5150fa0-6158-11ea-8593-000c29704efd/turtlebot-controller_spawner-5*.log  
[ INFO] [1583684597.169451054, 0.024000000]: waitForService: Service [/gazebo/set_physics_properties] is now available.  
[ INFO] [1583684597.170404371, 0.026000000]: waitForService: Service [/gazebo/set_physics_properties] is now available.  
[ INFO] [1583684597.326663890, 0.188000000]: Physics dynamic reconfigure ready.  
[ INFO] [1583684597.339738038, 0.200000000]: Physics dynamic reconfigure ready.
```

I found solution in the following website:[https://github.com/ros-simulation/gazebo\\_ros\\_demos/issues/30](https://github.com/ros-simulation/gazebo_ros_demos/issues/30)

So I did 2 steps, from [http://gazebosim.org/tutorials?tut=ros\\_installing](http://gazebosim.org/tutorials?tut=ros_installing)

- ROS Kinetic:

```
sudo apt-get install ros-kinetic-gazebo-ros-pkgs ros-kinetic-gazebo-ros-control
```

From [https://github.com/ros-simulation/gazebo\\_ros\\_demos/issues/30](https://github.com/ros-simulation/gazebo_ros_demos/issues/30)



ccpuwanan commented on 5 Apr 2019

This could be helpful

```
$ sudo apt-get install ros-kinetic-joint-state-controller  
$ sudo apt-get install ros-kinetic-effort-controllers  
$ sudo apt-get install ros-kinetic-position-controllers
```

5

2

4. If you run `rosrun turtlebot2i_gazebo turtlebot2i_world.launch` again, you will meet several error of this kind:

```
n_controller
[ERROR] [1583685281.449252306]: Exception thrown while initializing controller arm_elbow_flex_position_controller.
Could not find resource 'arm_elbow_flex_joint' in 'hardware_interface::EffortJointInterface'.
[ERROR] [1583685281.449337281]: Initializing controller 'arm_elbow_flex_position_controller' failed
[ERROR] [1583685282.461545, 0.000000]: Failed to load arm_elbow_flex_position_controller
```

This is because some controller names don't match in ROS files provided in Turtlebot2i packages.

Type `gedit $(rospack find phantomx_pincher_arm_description)/urdf/phantomx_pincher_arm.xacro` in a terminal, you can find the file defining joints of the `phantomx_pincher_arm`, set the viewing format to XML will help you view the file:

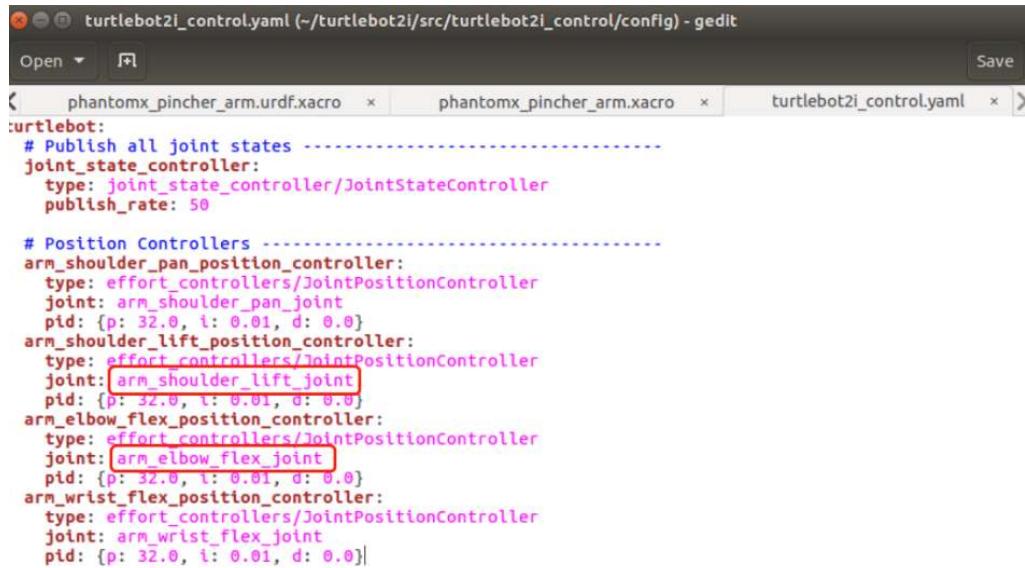
```
phantomx_pincher_arm.xacro (~/turtlebot2i/src/phantomx_pincher_a...phantomx_pincher_arm_description/urdf)
Open Save
phantomx_pincher_arm.xacro x turtlebot2i_control.yaml x turtlebot_gazebo.urdf.xacro x
<pincher_forearm parent= arm_pincher_link name= arm_forearm_color= ${color}
    vlimit="${joints_vlimit}" llimit="${elbow_llimit}"
    ulimit="${elbow_ulimit}"
    <origin xyz="0.0325 0 0.104" rpy="0 0 0"/>
</pincher_forearm>

<transmission name="arm_forearm_transmission">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="arm_forearm_joint">
        <hardwareInterface>EffortJointInterface</hardwareInterface>
    </joint>
    <actuator name="arm_forearm_actuator">
        <hardwareInterface>EffortJointInterface</hardwareInterface>
        <mechanicalReduction>1</mechanicalReduction>
    </actuator>
</transmission>

<!-- wrist joint -->
<pincher_wrist parent="arm_forearm_link" name= arm_wrist_color= ${color}
    vlimit="${joint_vlimit}"
    ulimit="${wrist_ulimit}"
    <origin xyz="0.104 0 0" rpy="0 0 0"/>
</pincher_wrist>
<transmission name="arm_wrist_flex_transmission">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="arm_wrist_flex_joint">
        <hardwareInterface>EffortJointInterface</hardwareInterface>
    </joint>
    <actuator name="arm_wrist_flex_actuator">
        <hardwareInterface>EffortJointInterface</hardwareInterface>
        <mechanicalReduction>1</mechanicalReduction>
    </actuator>
</transmission>

<!-- gripper -->
<xacro:include filename="$(find phantomx_pincher_arm_description)/urdf/
```

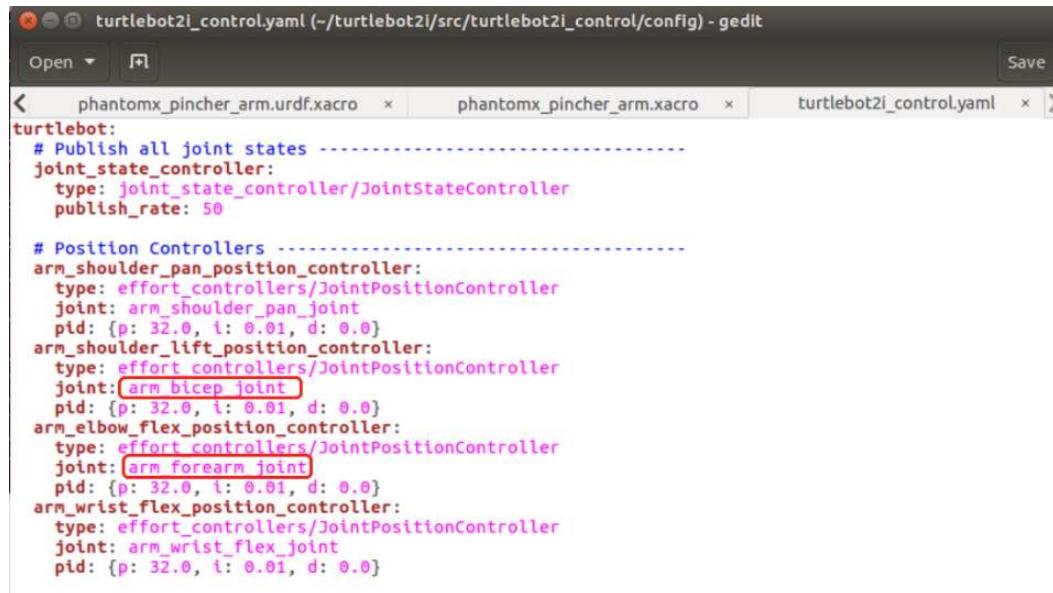
Open another terminal, type `gedit $(rospack find turtlebot2i_control)/config/turtlebot2i_control.yaml` you'll find how the arm controllers are defined, and you will find the joint names don't match.....



```
turtlebot:
  # Publish all joint states -----
  joint_state_controller:
    type: joint_state_controller/JointStateController
    publish_rate: 50

  # Position Controllers -----
  arm_shoulder_pan_position_controller:
    type: effort_controllers/JointPositionController
    joint: arm_shoulder_pan_joint
    pid: {p: 32.0, i: 0.01, d: 0.0}
  arm_shoulder_lift_position_controller:
    type: effort_controllers/JointPositionController
    joint: arm_shoulder_lift_joint
    pid: {p: 32.0, i: 0.01, d: 0.0}
  arm_elbow_flex_position_controller:
    type: effort_controllers/JointPositionController
    joint: arm_elbow_flex_joint
    pid: {p: 32.0, i: 0.01, d: 0.0}
  arm_wrist_flex_position_controller:
    type: effort_controllers/JointPositionController
    joint: arm_wrist_flex_joint
    pid: {p: 32.0, i: 0.01, d: 0.0}
```

So change the 2 joint names according to the definition in phantomx\_pincher\_arm.xacro, and don't forget to save it. Then it should look like this:



```
turtlebot:
  # Publish all joint states -----
  joint_state_controller:
    type: joint_state_controller/JointStateController
    publish_rate: 50

  # Position Controllers -----
  arm_shoulder_pan_position_controller:
    type: effort_controllers/JointPositionController
    joint: arm_shoulder_pan_joint
    pid: {p: 32.0, i: 0.01, d: 0.0}
  arm_shoulder_lift_position_controller:
    type: effort_controllers/JointPositionController
    joint: arm_bicep_joint
    pid: {p: 32.0, i: 0.01, d: 0.0}
  arm_elbow_flex_position_controller:
    type: effort_controllers/JointPositionController
    joint: arm_forearm_joint
    pid: {p: 32.0, i: 0.01, d: 0.0}
  arm_wrist_flex_position_controller:
    type: effort_controllers/JointPositionController
    joint: arm_wrist_flex_joint
    pid: {p: 32.0, i: 0.01, d: 0.0}
```

5. If you run `rosrun turtlebot2i_gazebo turtlebot2i_world.launch` again, you will find an error like this:

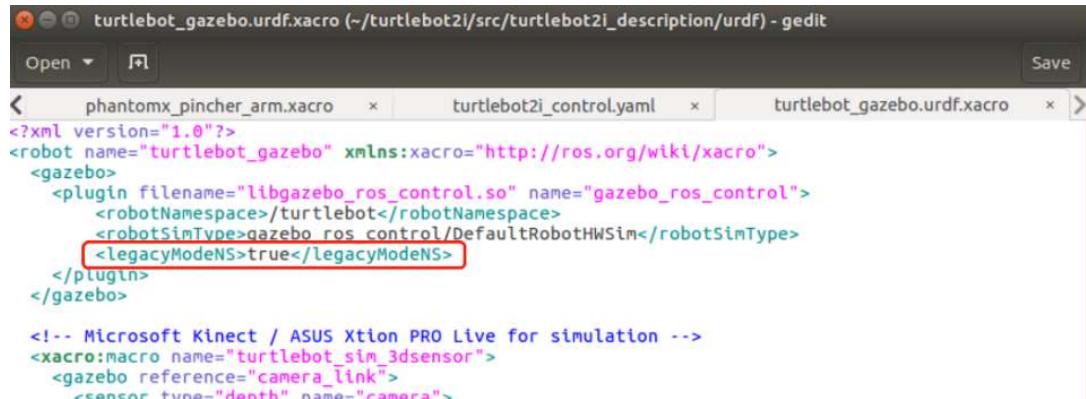
```
[ INFO] [1583686421.097990116]: Loaded gazebo_ros_control.  
[ INFO] [1583686421.098399113]: Loading gazebo_ros_control plugin  
[ERROR] [1583686421.098400956]: GazeboRosControlPlugin missing <legacyModeNS> while using DefaultRobotHWSim, defaults to true.  
This setting assumes you have an old package with an old implementation of DefaultRobotHWSim, where the robotNamespace is disregarded and absolute paths are used instead.  
If you do not want to fix this issue in an old package just set <legacyModeNS> to true.
```

I found solution of this error on this

website:[https://answers.ros.org/question/292444/gazebo\\_ros\\_control-plugin-gazeboroscontrolplugin-missing-legacymodens-defaultrobothwsim/](https://answers.ros.org/question/292444/gazebo_ros_control-plugin-gazeboroscontrolplugin-missing-legacymodens-defaultrobothwsim/)

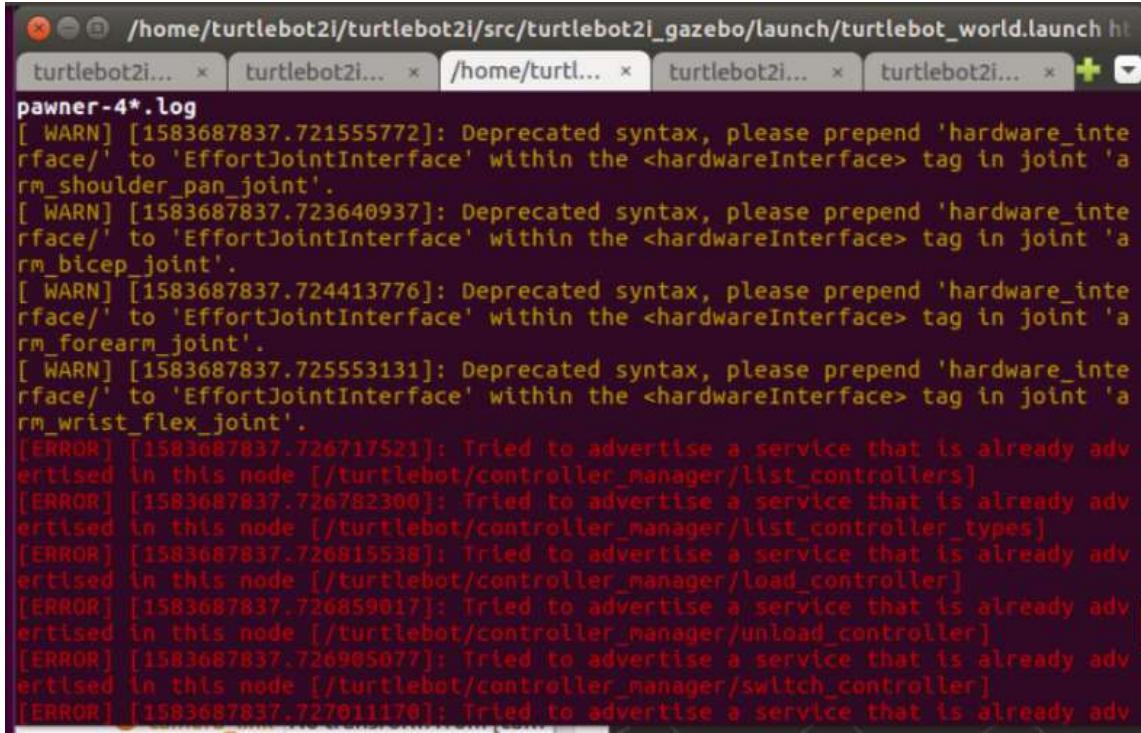
Type `gedit`

`~/turtlebot2i/src/turtlebot2i_description/urdf/turtlebot_gazebo.urdf.xacro` in a terminal, and add 1 line of text as suggested by the answer above, and don't forget to save it.



```
<!-- Microsoft Kinect / ASUS Xtion PRO Live for simulation -->  
<xacro:macro name="turtlebot_sim_3dsensor">  
  <gazebo reference="camera_link">  
    <sensor type="depth" name="camera">
```

## 6. Now if you run again, you will still see these error



```
pawner-4*.log
[ WARN] [1583687837.721555772]: Deprecated syntax, please prepend 'hardware_interface/' to 'EffortJointInterface' within the <hardwareInterface> tag in joint 'arm_shoulder_pan_joint'.
[ WARN] [1583687837.723640937]: Deprecated syntax, please prepend 'hardware_interface/' to 'EffortJointInterface' within the <hardwareInterface> tag in joint 'arm_bicep_joint'.
[ WARN] [1583687837.724413776]: Deprecated syntax, please prepend 'hardware_interface/' to 'EffortJointInterface' within the <hardwareInterface> tag in joint 'arm_forearm_joint'.
[ WARN] [1583687837.725553131]: Deprecated syntax, please prepend 'hardware_interface/' to 'EffortJointInterface' within the <hardwareInterface> tag in joint 'arm_wrist_flex_joint'.
[ERROR] [1583687837.726717521]: Tried to advertise a service that is already advertised in this node [/turtlebot/controller_manager/list_controllers]
[ERROR] [1583687837.726782300]: Tried to advertise a service that is already advertised in this node [/turtlebot/controller_manager/list_controller_types]
[ERROR] [1583687837.726815538]: Tried to advertise a service that is already advertised in this node [/turtlebot/controller_manager/load_controller]
[ERROR] [1583687837.726859017]: Tried to advertise a service that is already advertised in this node [/turtlebot/controller_manager/unload_controller]
[ERROR] [1583687837.726905077]: Tried to advertise a service that is already advertised in this node [/turtlebot/controller_manager/switch_controller]
[ERROR] [1583687837.727011170]: Tried to advertise a service that is already adv
```

For the orange error, I found explanation here. <http://wiki.ros.org/urdf/XML/Transmission>

The transmission has the following elements:

<type> (one occurrence)

Specifies the transmission type.

<joint> (one or more occurrences)

A joint the transmission is connected to. The joint is specified by its **name** attribute, and the following sub-elements:

<hardwareInterface> (one or more occurrences)

Specifies a supported joint-space **hardware interface**. Note that the value of this tag should be **EffortJointInterface** when this transmission is loaded in Gazebo and **hardware\_interface/EffortJointInterface** when this transmission is loaded in RobotHW.

<actuator> (one or more occurrences)

An actuator the transmission is connected to. The actuator is specified by its **name** attribute, and the following sub-elements:

<mechanicalReduction> (optional)

Specifies a mechanical reduction at the joint/actuator transmission. This tag may not be needed for all transmissions.

<hardwareInterface> (optional) (one or more occurrences)

Specifies a supported joint-space **hardware interface**. Note that <hardwareInterface> tag should only be specified here for ROS releases prior to Indigo. The correct place to specify this tag is in <joint> tag. More details about this can be found [here](#).

So it seems that it should be **EffortJointInterface** when we do simulation in Gazebo. And it is **EffortJointInterface** in **phantomx\_pincher\_arm.xacro**

So I just leave it here.



```
phantomx_pincher_arm.xacro (~/turtlebot2i/src/phantomx_pincher_a...phantomx_pincher_arm_description/urdf)
Open Save
phantomx_pincher_arm.xacro x turtlebot2i_control.yaml x turtlebot_gazebo.urdf.xacro x >
<?xml version="1.0"?>
<!-- Arm description for PhantomX Pincher arm -->
<robot name="phantomx_pincher_arm" xmlns:xacro="http://ros.org/wiki/xacro">
  <xacro:include filename="$(find phantomx_pincher_arm_description)/urdf/arm_hw...
```

The code block shows a portion of the `phantomx_pincher_arm.xacro` file. A red box highlights the following XML snippet:

```
</joint>
<actuator name="arm_shoulder_pan_actuator">
  <hardwareInterface>EffortJointInterface</hardwareInterface>
  <mechanicalReduction>1</mechanicalReduction>
</actuator>
</transmission>
```

For the red error, I have figured out neither whether it will affect simulation nor its solution.

7. Now open Rviz, and subscribe to /camera/rgb/image\_raw topic. You will find Gazebo die, code 134. Now you need to update Gazebo by typing following commands in terminal.

```
sudo sh -c 'echo "deb http://packages.osrfoundation.org/gazebo/ubuntu-stable `lsb_release -cs` main" > /etc/apt/sources.list.d/gazebo-stable.list'
```

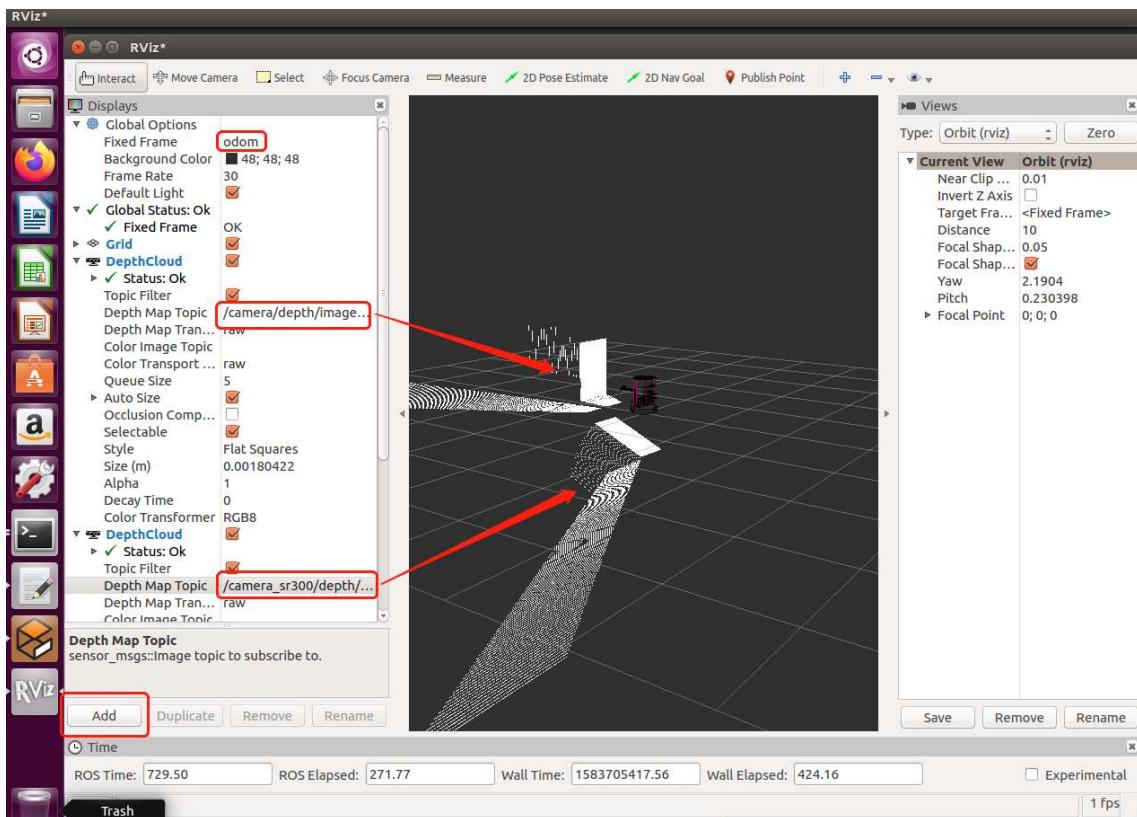
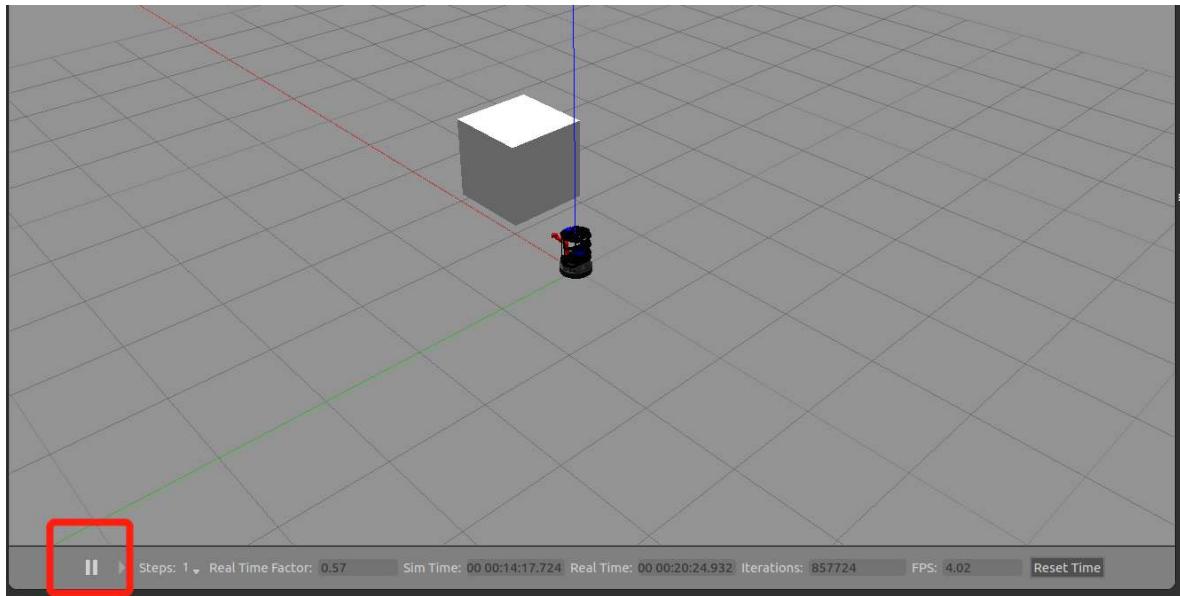
```
wget http://packages.osrfoundation.org/gazebo.key -O - | sudo apt-key add -
```

```
sudo apt-get update
```

```
sudo apt-get install gazebo7
```

Reference <https://stackoverflow.com/questions/55181205/gazebo-crashing-when-subscribing-to-scan>

8. Open Rviz using `rosrun rviz rviz`, and click add, then add the components as shown in the following figure, plus RobotModel which is not shown in the figure to visualize the robot. Then you can see 1 depth cloud is on the wrong direction. Make sure your Gazebo is not paused, or there will be no message published to topics. You can also change the “paused” argument to false in `turtlebot_world.launch`.



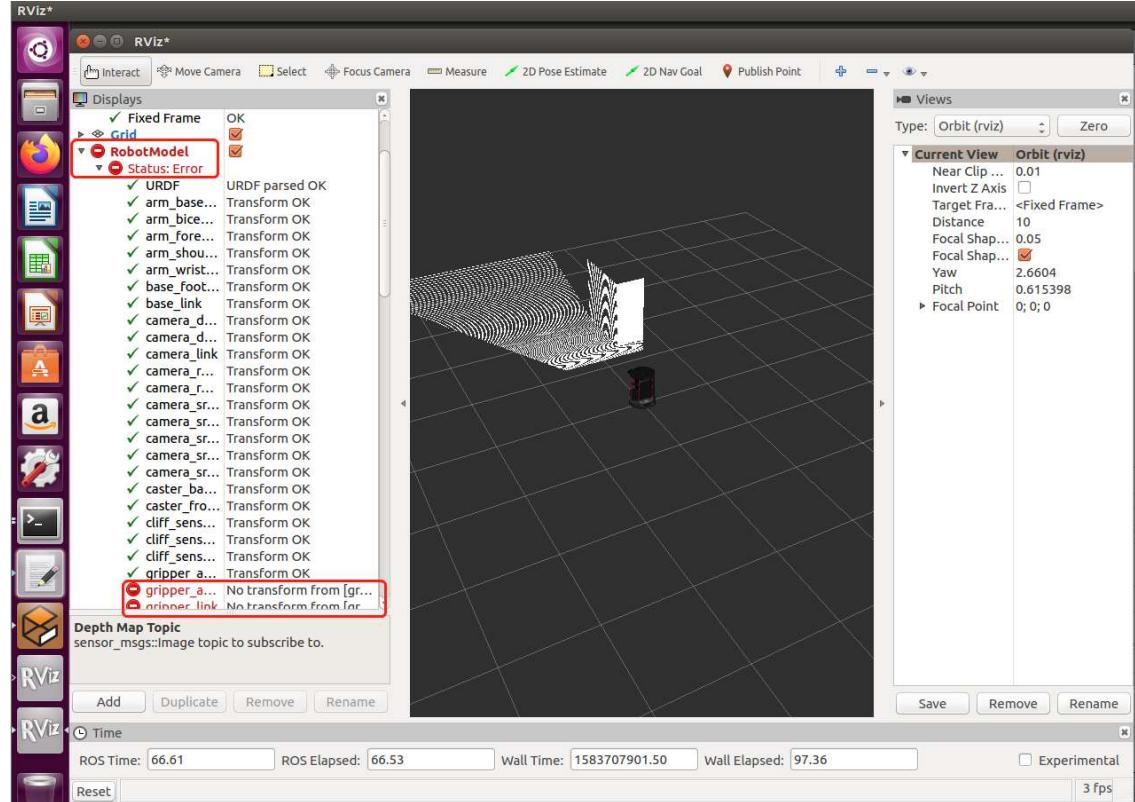
To fix the wrong orientation problem, type

```
gedit $(rospack find turtlebot2i_description)/urdf/sensors/sr300.urdf.xacro
```

in your terminal. Change the number "0" to "-1.16", make sure you are making modification for the right camera, then restart Gazebo and Rviz.

```
<!--  
== Depth joints & links ==<joint name="camera_sr300_depth_joint" type="fixed">  
  <origin xyz="0 0 ${sr300_can_depth_offset} 0" rpy="0 0 0" />  
  <parent link="camera_sr300_rgb_frame" />  
  <child link="camera_sr300_depth_frame" />  
</joint>  
<link name="camera_sr300_depth_frame"/>  
  
<joint name="camera_sr300_depth_optical_joint" type="fixed">  
  <origin xyz="0 0 0" rpy="-${M_PI/2} 0 ${M_PI/2}" />  
  <parent link="camera_sr300_depth_frame" />  
  <child link="camera_sr300_depth_optical_frame" />  
</joint>  
<link name="camera_sr300_depth_optical_frame"/>
```

Now you can still see some error in RobotModel, which is that some frames doesn't have transformation. I haven't figured out the solution yet, and I think it might won't affect my simulation at current stage.



9. Now you may want to generate a map using teleop by rosrun turtlebot\_teleop keyboard.launch, make sure you Gazebo is not paused, but you will find the robot doesn't move in Gazebo. To solve this, add the following test in your turtlebot\_world.launch, before </launch> at the end. To reach this file, use gedit \$(rospack find turtlebot2i\_gazebo)/launch/turtlebot\_world.launch

```
<!-- Velocity
muxer -->
<node pkg="nodelet" type="nodelet"
      name="mobile_base_nodelet_manager" args="manager"/>
<node pkg="nodelet" type="nodelet" name="cmd_vel_mux"
      args="load yocs_cmd_vel_mux/CmdVel_muxNodelet
            mobile_base_nodelet_manager">
<param name="yaml_cfg_file" value="$(find
            turtlebot2i_bringup)/param/mux.yaml" />
<remap from="cmd_vel_mux/output"
      to="mobile_base/commands/velocity"/>
</node>
```

```
turtlebot_world.launch (~/turtlebot2i/src/turtlebot2i_gazebo/launch) - gedit
Save
<!-- Velocity
muxer -->
<node pkg="nodelet" type="nodelet"
      name="mobile_base_nodelet_manager" args="manager"/>
<node pkg="nodelet" type="nodelet" name="cmd_vel_mux"
      args="load yocs_cmd_vel_mux/CmdVel_muxNodelet
            mobile_base_nodelet_manager">
<param name="yaml_cfg_file" value="$(find
            turtlebot2i_bringup)/param/mux.yaml" />
<remap from="cmd_vel_mux/output"
      to="mobile_base/commands/velocity"/>
</node>

<!-- We resume the logic in empty_world.launch, changing only the name of the world to be
launched -->
<include file="$(find gazebo_ros)/launch/empty_world.launch">
<arg name="world_name" value="$(find turtlebot2i_gazebo)/worlds/turtlebot.world"/>
<arg name="debug" value="$(arg debug)" />
<arg name="gui" value="$(arg gui)" />
<arg name="paused" value="$(arg paused)" />
<arg name="use_sim_time" value="$(arg use_sim_time)" />
<arg name="headless" value="$(arg headless)" />
</include>

<!-- Load the URDF into the ROS Parameter Server --
<arg name="urdf_file" default="$(find xacro)/xacro --inorder '$(find turtlebot2i_description)/
robots/$(arg base)_$(arg stacks)_$(arg 3d_sensor).urdf.xacro'" />
<param name="robot_description" command="$(arg urdf_file)" />

<!-- Run a python script to send a service call to gazebo_ros to spawn a URDF robot -->
<node name="urdf_spawner" pkg="gazebo_ros" type="spawn_model" respawn="false" output="screen"
      args="-urdf -model turtlebot -param robot_description"/>

<!-- ros_control turtlebot launch file -->
<include file="$(find turtlebot2i_control)/launch/turtlebot2i_control.launch" />

<!-- Velocity muxer -->
<node pkg="nodelet" type="nodelet" name="mobile_base_nodelet_manager" args="manager"/>
<node pkg="nodelet" type="nodelet" name="cmd_vel_mux"
      args="load yocs_cmd_vel_mux/CmdVel_muxNodelet
            mobile_base_nodelet_manager">
<param name="yaml_cfg_file" value="$(find turtlebot2i_bringup)/param/mux.yaml" />
<remap from="cmd_vel_mux/output" to="mobile_base/commands/velocity"/>
</node>

</launch>
```

XML ▾ Tab Width: 8 ▾ Ln 23, Col 1 ▾ INS

Then restart Gazebo, you will find the you can control the robot using keyboard.

10. Then I tried `roslaunch turtlebot2i Bringup turtlebot_mapping.launch`, and I've made some modifications but it still can't build map yet.

Tried to run the demo first

Changed parameters

```
<?xml version="1.0"?>
<launch>
  <arg name="new_rtabmap" default="false"/>
  <arg name="localization" default="false"/>
  <arg name="rviz" default="false"/>
  <arg name="rtabmapviz" default="false"/>
```

```
<?xml version="1.0"?>
<launch>
  <arg name="new_rtabmap" default="false"/>
  <arg name="localization" default="false"/>
  <arg name="rviz" default="true"/>
  <arg name="rtabmapviz" default="true"/>
```

Run `roslaunch turtlebot2i Bringup turtlebot2i_demo1.launch`, and the program died, showing arbotix-9 died, so check the log file

After checking minimal.launch

```
turtlebot2i-simulation@ubuntu:~/turtlebot2i/src/turtlebot2i_Bringup$ echo
$TURTLEBOT_SIMULATION
false
turtlebot2i-simulation@ubuntu:~/turtlebot2i/src/turtlebot2i_Bringup$ export TURTLEBOT_SIMULATION=true
turtlebot2i-simulation@ubuntu:~/turtlebot2i/src/turtlebot2i_Bringup$ roslaunch turtlebot2i_Bringup turtlebot2i_demo1.launch
```

Run again, same error prompt, check the log file

```
< /turtlebot2i_mapping.launch > /turtlebot2i_demo1.launch > arbotix-9.log > minimal.launch > arbotix-9.log >
[rosrpc_client][INFO] 2020-03-08 17:19:37,220: init_node, name[arbotix], pid[56500]
[xmlrpc][INFO] 2020-03-08 17:19:37,220: XML-RPC server binding to 0.0.0.0:8
[xmlrpc][INFO] 2020-03-08 17:19:37,230: Started XML-RPC server [http://ubuntu:46721/]
[rosrpc.impl.masterslave][INFO] 2020-03-08 17:19:37,230: _ready: http://ubuntu:46721/
[rosrpc.init][INFO] 2020-03-08 17:19:37,230: ROS Slave URI: [http://ubuntu:46721/]
[xmlrpc][INFO] 2020-03-08 17:19:37,232: xml rpc node: starting XML-RPC server
[rosrpc.registration][INFO] 2020-03-08 17:19:37,233: Registering with master node http://localhost:1131
[rosrpc.init][INFO] 2020-03-08 17:19:37,331: registered with master
[rosrpc.core][INFO] 2020-03-08 17:19:37,331: initializing /rosout core topic
[rosrpc.rosout][INFO] 2020-03-08 17:19:37,355: connected to core topic /rosout
[rosrpc.simtime][INFO] 2020-03-08 17:19:37,379: /use_sim_time is not set, will not subscribe to simulated time [/clock] topic
[rosrpc.core][INFO] 2020-03-08 17:19:37,607: signal_shutdown [atexit]
[rosrpc.impl.masterslave][INFO] 2020-03-08 17:19:37,630: atexit
```

`grep -r "use_sim_time" "/opt/ros/kinetic/share/"`

```
/opt/ros/kinetic/share/rtabmap_ros/launch/rtabmap.launch: <arg name="use_sim_time"
  default="false"/>
/opt/ros/kinetic/share/rtabmap_ros/launch/rtabmap.launch: <param if="$arg use_
sim_time" name="use_sim_time" value="true"/>
```

Check it, and comment the original one and set the default value to true

```
< /turtlebot2i_demo1.launch > arbotix-9.log > minimal.launch > arbotix-9.log > *rtabmap.launch >
<arg name="rtabmapviz" default="true" />
<arg name="rviz" default="false" />
<!-- Localization-only mode -->
<arg name="localization" default="false"/>
<!-- sim time for convenience, if playing a rosbag -->
<arg name="use_sim_time" default="false"/>-->
<arg name="use_sim_time" default="true"/>
<param if="$arg use_sim_time" name="use_sim_time" value="true"/>
```

Run again still doesn't work, same log content again

## Noticed

Check that file

```
④ arbotix_driver (~~/turtlebot2/src/arbotix_ros/arbotix_python/bin) - eedit
Open ▾

< turtlebot2_demo1.launch x minimal.launch x rtabmap.launch x arbotix-9.log x arbotix_driver x
class ArbotixROS(ArbotiX):
    def __init__(self):
        pause = False

        # load configurations
        port = rospy.get_param("~port", "/dev/arbotix")
        baud = int(rospy.get_param("~baud", "115200"))

        self.rate = rospy.get_param("~rate", 100.0)
        self.fake = rospy.get_param("~sim", False)

        self.use_sync_read = rospy.get_param("~sync_read", True) # use sync read?
        self.use_sync_write = rospy.get_param("~sync_write", True) # use sync write?

        # setup publishers
        self.diagnostics = DiagnosticsPublisher()
        self.joint_state_publisher = JointStatePublisher()

        # start an arbotix driver
        if not self.fake:
            Arbotix.__init__(self, port, baud)
            rospy.sleep(1.0)
            rospy.loginfo("Started Arbotix connection on port " + port + ".")
        else:
            rospy.loginfo("Arbotix being simulated.")

    # setup joints
    self.joints = dict()
    for name in rospy.get_param("~joints", dict()).keys():
        joint_type = rospy.get_param("~joints/" + name + "/type", "dynamixel")
        if joint_type == "dynamixel":
            self.joints[name] = DynamixelServo(self, name)
        elif joint_type == "prismatic":
            self.joints[name] = PrismaticDynamixelServo(self, name)
        else:
            joint_type == "hhuhu_caron"
    Python Tab Width: 8 Ln 72, Col 34 INS
```

Change the parameter to true

```
arbotix_driver (-/turtlebot2l/src/arbotix_ros/arbotix_python/bin) - gedit
Open ▾
turtlebot2l_demo1.launch x minimal.launch x rtbmap.launch x arbotix-9.log x arbotix_driver x
class ArbotixROS(Arbotix):
    def __init__(self):
        pause = False

        # load configurations
        port = rospy.get_param("~port", "/dev/arbotix")
        baud = int(rospy.get_param("~baud", "115200"))

        self.rate = rospy.get_param("~rate", 100)
        # self.fake = rospy.get_param("~sim", False)
        self.fake = rospy.get_param("~sim", True)

        self.use_sync_read = rospy.get_param("~sync_read",True) # use sync read?
        self.use_sync_write = rospy.get_param("~sync_write",True) # use sync write?

        # setup publishers
        self.diagnostics = DiagnosticsPublisher()
        self.join_state_publisher = JointStatePublisher()

        # start an arbotix driver
        if not self.fake:
            Arbotix__init__(self, port, baud)
            rospy.sleep(1.0)
            rospy.loginfo("started Arbotix connection on port " + port + ".")
        else:
            rospy.loginfo("Arbotix being simulated.")

        # setup joints
        self.joints = dict()
        for name in rospy.get_param("~joints", dict()).keys():
            joint_type = rospy.get_param("~joints/" + name + "/type", "dynamixel")
            if joint_type == "dynamixel":
                self.joints[name] = DynamixelServo(self, name)
            elif joint_type == "prismatic":
                self.joints[name] = PrismaticDynamixelServo(self, name)

```

I decide to leave it here first, I remember there are redirection issues, I want to solve them then come back to this.

Searched for "no matching device found"

[http://wiki.ros.org/astra\\_camera](http://wiki.ros.org/astra_camera), it shows udev rule, so I want to try udev rules part on Full build instructions <https://github.com/Interbotix/turtlebot2i/wiki/Full-Build-Instructions>

Get stuck here

## udev rules

---

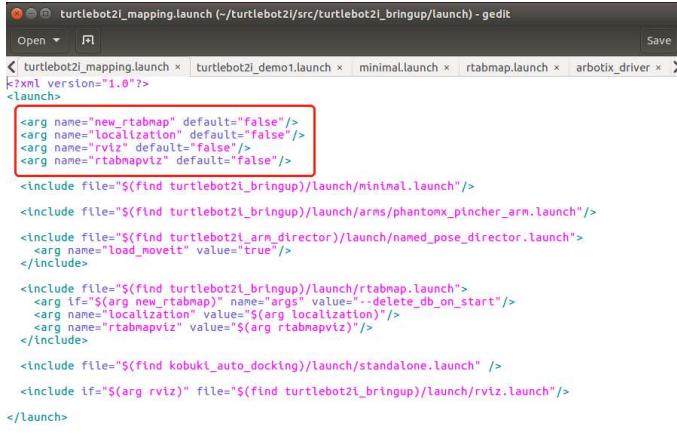
```
cd ~/turtlebot2i/  
goros  
rosrun kobuki_ftdi create_udev_rules  
rosrun astra_camera create_udev_rules  
cd ~/turtlebot2i/src/turtlebot2i_misc  
gedit 99-turtlebot2i.rules
```

- At this point we need to execute a couple commands that will return the serial number for the ArbotiX and the Kobuki. Remove and/or Re-plug both the Arbotix and the Kobuki USB cables and ensure they are powered on before running these commands:
  - `udevadm info -a -n /dev/ttyUSB0 | grep '{serial}' | head -n1`
  - `udevadm info -a -n /dev/ttyUSB1 | grep '{serial}' | head -n1`
- The output from these commands are the serial numbers used to fill in the blanks of lines 6 and 7 of the 99-turtlebot2i.rules file.

It seems that the demo1.launch and its dependenies are too complecated, so run mapping.launch

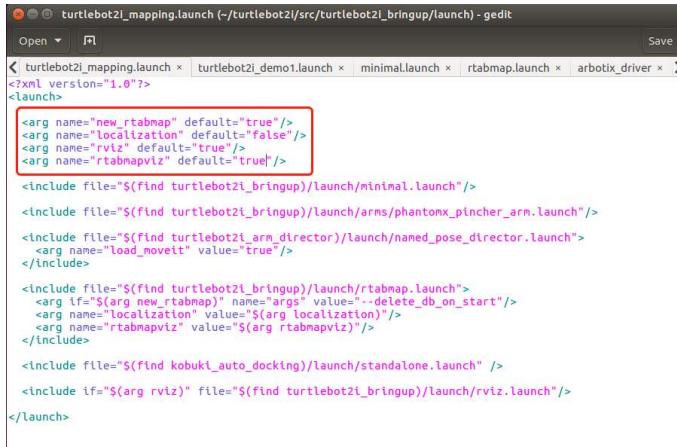
It seems that the demo1.launch and its dependenies are too complecated, so run mapping.launch

Modified these parameters



```
<?xml version="1.0"?>
<launch>
  <arg name="new_rtabmap" default="false"/>
  <arg name="localization" default="false"/>
  <arg name="rviz" default="false"/>
  <arg name="rtabmapviz" default="false"/gt;

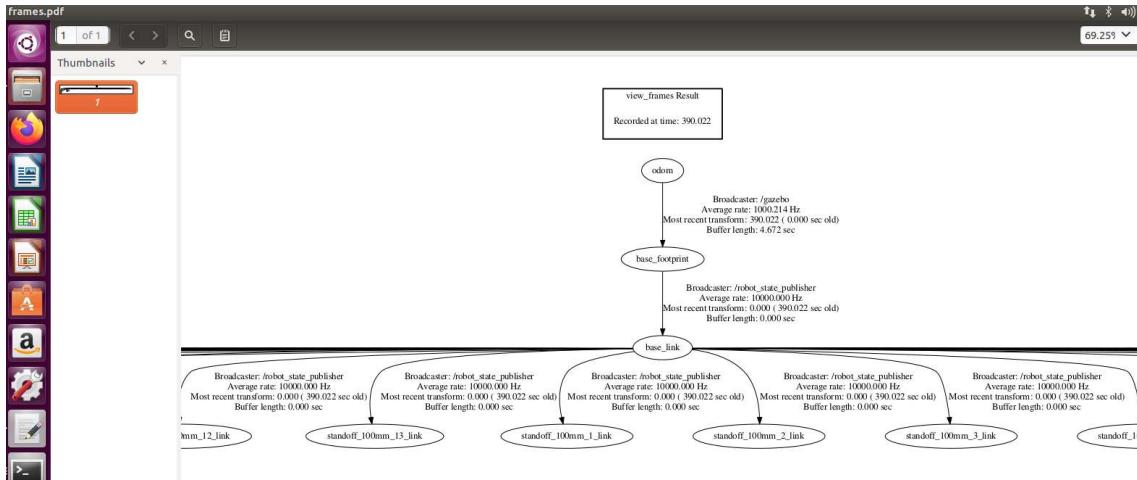
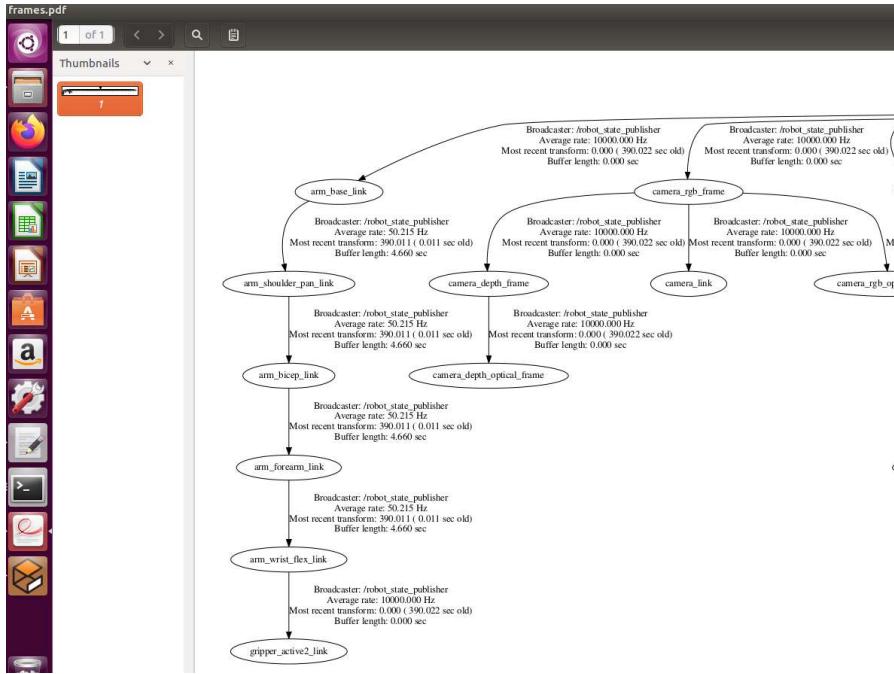
  <include file="$(find turtlebot2i_bringup)/launch/minimal.launch">/
  <include file="$(find turtlebot2i_bringup)/launch/arms/phantomx_pincher_arm.launch">
  <include file="$(find turtlebot2i_arm_director)/Launch/named_pose_director.launch">
    <arg name="load_moveit" value="true"/>
  </include>
  <include file="$(find turtlebot2i_bringup)/launch/rtabmap.launch">
    <arg if="$(arg new_rtabmap)" name="args" value="--delete_db_on_start"/>
    <arg name="localization" value="$(arg localization)"/>
    <arg name="rtabmapviz" value="$(arg rtabmapviz)"/>
  </include>
  <include file="$(find kobuki_auto_docking)/launch/standalone.launch" />
  <include if="$(arg rviz)" file="$(find turtlebot2i_bringup)/launch/rviz.launch"/>
</launch>
```



```
<?xml version="1.0"?>
<launch>
  <arg name="new_rtabmap" default="true"/>
  <arg name="localization" default="false"/>
  <arg name="rviz" default="true"/>
  <arg name="rtabmapviz" default="true"/gt;

  <include file="$(find turtlebot2i_bringup)/launch/minimal.launch">/
  <include file="$(find turtlebot2i_bringup)/launch/arms/phantomx_pincher_arm.launch">
  <include file="$(find turtlebot2i_arm_director)/Launch/named_pose_director.launch">
    <arg name="load_moveit" value="true"/>
  </include>
  <include file="$(find turtlebot2i_bringup)/launch/rtabmap.launch">
    <arg if="$(arg new_rtabmap)" name="args" value="--delete_db_on_start"/>
    <arg name="localization" value="$(arg localization)"/>
    <arg name="rtabmapviz" value="$(arg rtabmapviz)"/>
  </include>
  <include file="$(find kobuki_auto_docking)/launch/standalone.launch" />
  <include if="$(arg rviz)" file="$(find turtlebot2i_bringup)/launch/rviz.launch"/>
</launch>
```

I think mapping should need Gazebo, so I launched Gazebo first, and I remember there are TF issues between map->odom and odom->base\_footprint, so I checked TF after launching Gazebo



So we can see that the TF tree should be `map->odom->base_footprint->base_link` and there should be frames on `arm_base_link` branch

Now run `turtlebot2i_mapping.launch` and wait for error....

Arbotix died again, but I found something new

```
turtlebot... x turtlebo... x turtlebo... x turtlebo... x turtlebo... x turtlebo... x turtlebo... x +
```

Traceback (most recent call last):  
 File "/home/turtlebot2i-simulation/turtlebot2i/src/arobotix\_ros/arobotix\_python/bin/arobotix\_driver", line 181, in <module>  
 a = ArobotixROS()  
 File "/home/turtlebot2i-simulation/turtlebot2i/src/arobotix\_ros/arobotix\_python/bin/arobotix\_driver", line 75, in \_\_init\_\_  
 Arobotix.\_init\_\_(self, port, baud)  
 File "/home/turtlebot2i-simulation/turtlebot2i/src/arobotix\_ros/arobotix\_python/src/arobotix\_python/arobotix.py", line 55, in \_\_init\_\_  
 self.\_ser.open()  
 File "/usr/lib/python2.7/dist-packages/serial/serialposix.py", line 294, in open  
 raise SerialException(msg errno, "could not open port %s: %s" % (self.\_port,  
 serial.serialutil.SerialException: [Errno 2] could not open port /dev/arobotix: [Errno 2] No such file or directory: '/dev/arobotix'  
[INFO] [158371963.291707455, 1880.738000000]: rtabmapviz: Using configuration from [/opt/ros/melodic/share/rtabmap\_ros/launch/config/rtabviz.launch]  
[INFO] [158371963.291707455, 1880.738000000]: rtabmapviz process has died! pid 100062, exit code 1, cmd /home/turtlebot2i-simulation/turtlebot2i/src/arobotix\_ros/arobotix\_python/bin/arobotix\_driver --name=arobotix --log=/home/turtlebot2i-simulation/.ros/log/c2fc135c-01a6-11ea-aace-000c2fa8d089/arobotix\_0.log  
[INFO] [158371963.291707455, 1880.738000000]: log file: /home/turtlebot2i-simulation/.ros/log/c2fc135c-01a6-11ea-aace-000c2fa8d089/arobotix\_0.log

It seems that the program is looking for real port, so there maybe simulation flag, I

want to check it.

I added 1 line `rospy.set_param('~sim', True)`

```
arbotix_driver (~turtlebot2/src/arbotix_ros/arbotix_python/bin) - eedit

Open ▾  arbotix_driver (~turtlebot2/src/arbotix_ros/arbotix_python/bin) - eedit
Save

< turtlebot2_demo1.launch > minimal.launch > rtabmap.launch > arbotix_driver > arbotix.8.log >

< /home/ctrluser/catkin_ws/src/arbotix_ros/arbotix_node/arbotix_node.py>
# name : [ControllerClass, pause]
controller_types = { "follow_controller" : FollowController,
                     "diff_controller" : DiffController,
                     "omni_controller" : OmniController,
                     "linear_controller" : LinearControllerAbsolute,
                     "linear_controller_l" : LinearControllerIncremental }

#####
# Main ROS interface
<class ArbotixROS(Arbotix):>

    def __init__(self):
        pause = False

        # load configurations
        port = rospy.get_param("~port", "/dev/arbotix")
        baud = int(rospy.get_param("~baud", '115200'))

        self.rate = rospy.get_param("~rate", 100.0)
        rospy.set_param('~sim', True)
        self.fake = rospy.get_param('~sim', False)

        self.use_sync_read = rospy.get_param("~sync_read",True)          # use sync read?
        self.use_sync_write = rospy.get_param("~sync_write",True)        # use sync write?

        # setup publishers
        self.diagnostics = DiagnosticsPublisher()
        self.joint_state_publisher = JointStatePublisher()

        # start an arbotix driver
        if not self.fake:
            Arbotix.__init__(self, port, baud)
            rospy.sleep(1.0)
            rospy.loginfo("Started Arbotix connection on port " + port + ".")
        else:
            rosrun.lnainfo("Arbotix heino simulated.")

    
```

## New error, never met before

```
turtlebot... x turtlebo... x turtlebo... x turtlebo... x turtlebo... x turtlebo... x +
```

```
process[dock_drive-47]: started with pid [120695]
process[vrz-48]: started with pid [120705]
[ INFO] [1583720636.13979446]: Starting node...
[ INFO] [1583720636.403407276]: Starting node...
[ INFO] [1583720636.648639336, 2761.447680000]: Initializing nodelet with 4 work
ers, threads
[INFO] [1583720636.663251, 2761.4598000]: Started parallel Gripper Controller.
[INFO] [1583720637.21380473]: Initializing nodelet with 12 worker threads.
[ERROR] [1583720637.332734, 2762.0478000]: bad callback: <bound method GripperAct
ionController.stateCb of <__main__.GripperActionController instance at 0x7fedde
92b00>>
rbeckback (most recent call last):
  File "/opt/ros/kinetic/lib/python2.7/dist-packages/rospy/topics.py", line 758,
    in _on_message_callback
      cb(msg)
  File "/home/turtlebot2-simulation/turtlebot2/src/arobotix_ros/arobotix_control
ers/bin/gripper_controller", line 245, in stateCb
    if self._joint in msg.names:
AttributeError: GripperActionController instance has no attribute 'joint'
```

I think these errors are because of remapping of topics in some files

```
[ INFO] [1503720652.384783982, 2770.002800000]: No matching device found.... waiting for devices. Reason: std::cxx11::string astra_wrapper::AstraDriver::resolveDeviceURI(const string&) @ /home/turtlebot2i-simulation/turtlebot2i/src/ros_astra_camera/src/astra_driver.cpp @ 709 : Invalid device number 1, there are 0 devices connected.
[ WARN] [1503720654.090875800, 2770.922000000]: /rtabmap/rtabmapviz: Did not receive data since 5 seconds! Make sure the input topics are published ("$ rostopic hz my_topic") and the timestamps in their header are set. If topics are coming from different computers, make sure the clocks of the computers are synchronized ("ntpdate"). If topics are not published at the same rate, you could increase "queue_size" parameter (current=10).
/rtabmap/rtabmapviz subscribed to (approx sync):
/camera/depth_registered/image_rect_color,
/camera/depth_registered/image_raw,
/camera/rgb/camera_info,
/scan
[INFO] [1583720655.335087, 2771.5780000]: arm_controller: Done.
[ INFO] [1583720655.346617539, 2771.584000000]: Completed trajectory execution with status SUCCEEDED .
[ INFO] [1583720655.346617539, 2771.584000000]: /rtabmap/rtabmap: Waiting for device. Reason: std::cxx11::string astra_wrapper::AstraDriver::resolveDeviceURI(const string&) @ /home/turtlebot2i-simulation/turtlebot2i/src/ros_astra_camera/src/astra_driver.cpp @ 709 : Invalid device number 1, there are 0 devices connected.
[ WARN] [1583720657.984524628, 2772.944000000]: /rtabmap/rtabmap: Did not receive data since 5 seconds! Make sure the input topics are published ("$ rostopic hz my_topic") and the timestamps in their header are set. If topics are coming from different computers, make sure the clocks of the computers are synchronized ("ntpdate"). If topics are not published at the same rate, you could increase "queue_size" parameter (current=10).
/rtabmap/rtabmap subscribed to (approx sync):
/camera/depth_registered/image_raw,
/camera/depth_registered/image_rect_color,
/camera/rgb/camera_info,
/scan
```

Wait, maybe the red error is because of controller conflict? Which means controller is

loaded twice?

I think that's right, I shut down gazebo and red error is gone, only yellow warning remains, now I think I should comment the controller part in turtlebot2i\_world.launch

03/09/2020

Checking file contents, which will be screenshot in the next document.

Wait, I think it now may not be the controller issue, as when we operate real robot, the controller should be imbedded?

I find some part repeated in Simulation files and Physical robot operating files

```
turtlebot2i_world.launch (~/turtlebot2i/src/turtlebot2i_gazebo/launch) - gedit
Open ▾ Save
<!--<arg name="world_file" default="$(env TURTLEBOT_GAZEBO_WORLD_FILE)" />-->

<!-- We resume the logic in empty_world.launch, changing only the name of the world to be
launched -->
<include file="$(find gazebo_ros)/launch/empty_world.launch">
<!--<arg name="world_name" value="$(arg world_file)" />-->
<arg name="world_name" value="$(find turtlebot2i_gazebo)/worlds/test2.world"/>
<arg name="debug" value="$(arg debug)" />
<arg name="gui" value="$(arg gui)" />
<arg name="paused" value="$(arg paused)" />
<arg name="use_sim_time" value="$(arg use_sim_time)" />
<arg name="headless" value="$(arg headless)" />
</include>

<!-- Load the URDF into the ROS Parameter Server -->
<arg name="urdf_file" default="$(find xacro)/xacro --inorder '$(find turtlebot2i_description)/
robots/$(arg base)_$(arg stacks)_$(arg 3d_sensor).urdf.xacro'" />
<param name="robot_description" command="$(arg urdf_file)" />

<!-- Run a python script to send a service call to gazebo_ros to spawn a URDF robot -->
<node name="urdf_spawner" pkg="gazebo_ros" type="spawn_model" respawn="false" output="screen"
args="-urdf -model turtlebot -param robot_description"/>

<!-- ros_control turtlebot launch file -->
<include file="$(find turtlebot2i_control)/launch/turtlebot2i_control.launch" />

<!-- Velocity muxer -->
<node pkg="nodelet" type="nodelet" name="mobile_base_nodelet_manager" args="manager"/>
<node pkg="nodelet" type="nodelet" name="cmd_vel_mux"
args="load yocs_cmd_vel_mux/CmdVelMuxNodelet mobile_base_nodelet_manager">
<param name="yaml_cfg_file" value="$(find turtlebot2i_bringup)/param/mux.yaml" />
<remap from="cmd_vel_mux/output" to="mobile_base/commands/velocity"/>
</node>

</launch>
```

```
mobile_base.launch.xml (~/turtlebot2i/src/turtlebot2i_bringup/launch/includes/kobuki) - gedit
Open Save
empty_world.launch x standalone.launch x turtlebot2i_control.launch x mobile_base.launch.xml x >
<!--
  Kobuki's implementation of turtlebot's mobile base.
-->
<launch>
  <arg name="serialport"/> <!-- TODO: use the serialport parameter to set the serial port of kobuki -->

  <node pkg="nodelet" type="nodelet" name="mobile_base_nodelet_manager" args="manager"/>
  <node pkg="nodelet" type="nodelet" name="mobile_base" args="load kobuki_node/KobukiNodelet mobile_base_nodelet_manager">
    <rosparam file="$(find kobuki_node)/param/base.yaml" command="load"/>
    <param name="device_port" value="$(arg serialport)" />

    <remap from="mobile_base/odom" to="odom"/>
    <!-- Don't do this - force applications to use a velocity mux for redirection
        <remap from="mobile_base/commands/velocity" to="cmd_vel"/>
    -->
    <remap from="mobile_base/enable" to="enable"/>
    <remap from="mobile_base/disable" to="disable"/>
    <remap from="mobile_base/joint_states" to="joint_states"/>
  </node>

  <!-- velocity commands multiplexer -->
  <node pkg="nodelet" type="nodelet" name="cmd_vel_mux" args="load yocs_cmd_vel_mux/CmdVelMuxNodelet mobile_base_nodelet_manager">
    <param name="yaml_cfg_file" value="$(find turtlebot2i_bringup)/param/mux.yaml"/>
    <remap from="cmd_vel_mux/output" to="mobile_base/commands/velocity"/>
  </node>

  <!-- bumper/cliff to pointcloud -->
  <include file="$(find turtlebot2i_bringup)/launch/includes/kobuki/bumper2pc.launch.xml"/>
</launch>
```

minimal.launch (~/turtlebot2i/src/turtlebot2i\_bringup/launch) - gedit

```

<launch>
  <!-- Turtlebot2i -->
  <arg name="base" default="$(env TURTLEBOT_BASE)" doc="mobile base type"
[kobuki]"/>
  <arg name="stacks" default="$(env TURTLEBOT_STACKS)" doc="stack type displayed
in visualisation/simulation [interbotix]"/>
  <arg name="3d_sensor" default="$(env TURTLEBOT_3D_SENSOR)" doc="3d sensor types
[zr300, astra]"/>
  <arg name="simulation" default="$(env TURTLEBOT_SIMULATION)" doc="set flags to indicate
this turtle is run in simulation mode."/>
  <arg name="serialport" default="$(env TURTLEBOT_SERIAL_PORT)" doc="used by create to
configure the port it is connected on [/dev/ttyUSB0, /dev/ttyS0]"/>

  <!--Robot-->
  <!--Description-->
  <arg name="urdf_file" default="$(find xacro)/xacro --inorder '$(find turtlebot2i_description)/
robots/$(arg base)_$(arg stacks)_$(arg 3d_sensor).urdf.xacro'" />
  <param name="robot_description" command="$(arg urdf_file)" />

  <!-- important generally, but specifically utilised by the current app manager -->
  <param name="robot/name" value="$(optenv ROBOT turtlebot)"/>
  <param name="robot/type" value="turtlebot"/>

  <node pkg="robot_state_publisher" type="robot_state_publisher" name="robot_state_publisher">
    <param name="publish_frequency" type="double" value="5.0" />
  </node>
  <node pkg="diagnostic_aggregator" type="aggregator_node" name="diagnostic_aggregator" >
    <rosparam command="load" file="$(find turtlebot2i_bringup)/param/$(arg base)/
diagnostics.yaml" />
  </node>

  <!--Mobile Base-->
  <include file="$(find turtlebot2i_bringup)/launch/includes/$(arg base)/mobile_base.launch.xml">
    <arg name="serialport" value="$(arg serialport)" />
  </include>
</launch>

```

turtlebot2i\_control.launch (~/turtlebot2i/src/turtlebot2i\_control/launch) - gedit

```

<launch>
  <!-- Load joint controller configurations from YAML file to parameter server -->
  <rosparam file="$(find turtlebot2i_control)/config/turtlebot2i_control.yaml" command="load"/>

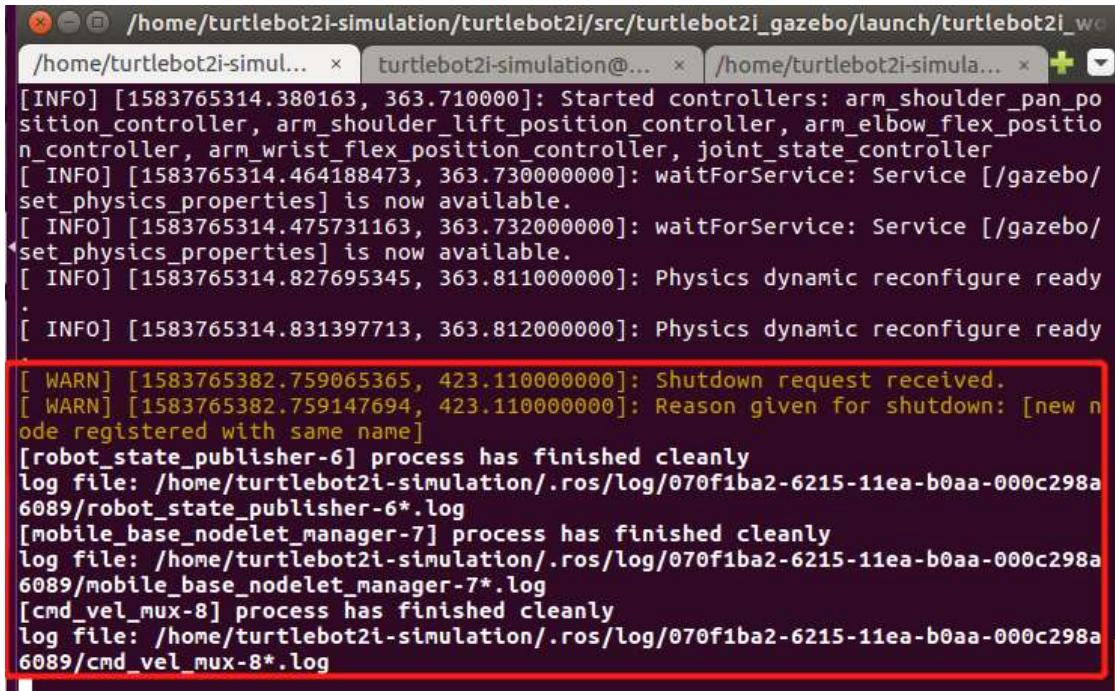
  <!-- load the controllers -->
  <node name="controller_spawner" pkg="controller_manager" type="spawner" respawn="false"
    output="screen" ns="/turtlebot" args="arm_shoulder_pan_position_controller
    arm_shoulder_lift_position_controller arm_elbow_flex_position_controller
    arm_wrist_flex_position_controller joint_state_controller"/>

  <!-- convert joint states to TF transforms for rviz, etc -->
  <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher"
    respawn="false" output="screen">
    <remap from="/joint_states" to="/turtlebot/joint_states" />
  </node>
</launch>

```

GG, I don't know why, but not modifying any file, and run turtlebot2i\_world.launch and turtlebot2i\_mapping.launch. Rviz is showing a robot with arm, which doesn't have an arm last night.

But Gazebo shows error, and I think it matches the my previous conclusion of repeatedly launching nodes. Nothing is in conflict, unbelievable.



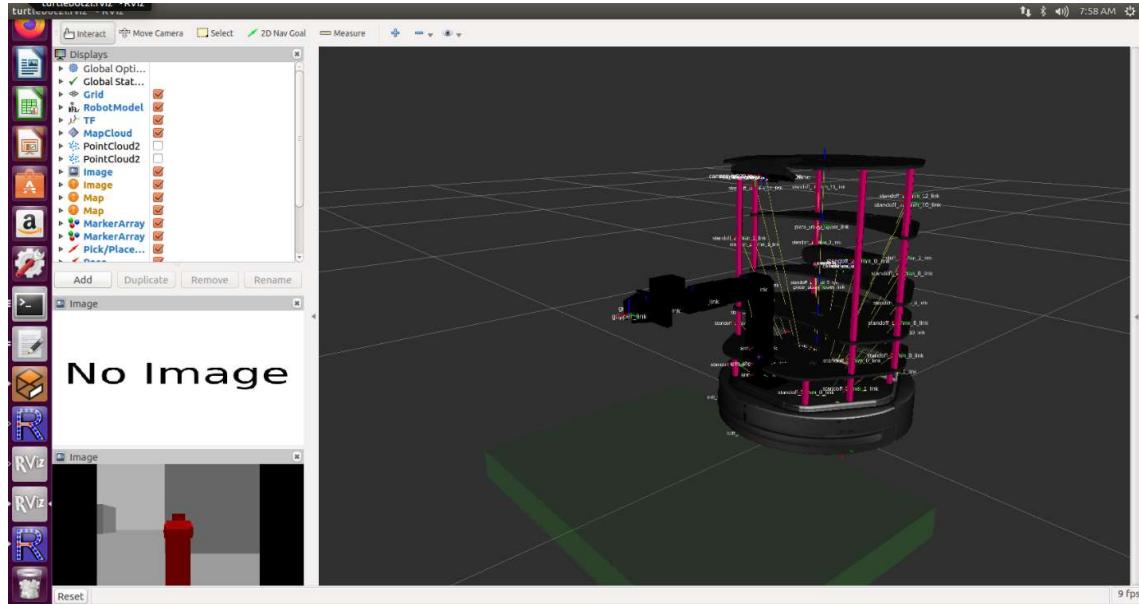
```
[INFO] [1583765314.380163, 363.710000]: Started controllers: arm_shoulder_pan_position_controller, arm_shoulder_lift_position_controller, arm_elbow_flex_position_controller, arm_wrist_flex_position_controller, joint_state_controller
[INFO] [1583765314.464188473, 363.730000000]: waitForService: Service [/gazebo/set_physics_properties] is now available.
[INFO] [1583765314.475731163, 363.732000000]: waitForService: Service [/gazebo/set_physics_properties] is now available.
[INFO] [1583765314.827695345, 363.811000000]: Physics dynamic reconfigure ready
.
[INFO] [1583765314.831397713, 363.812000000]: Physics dynamic reconfigure ready

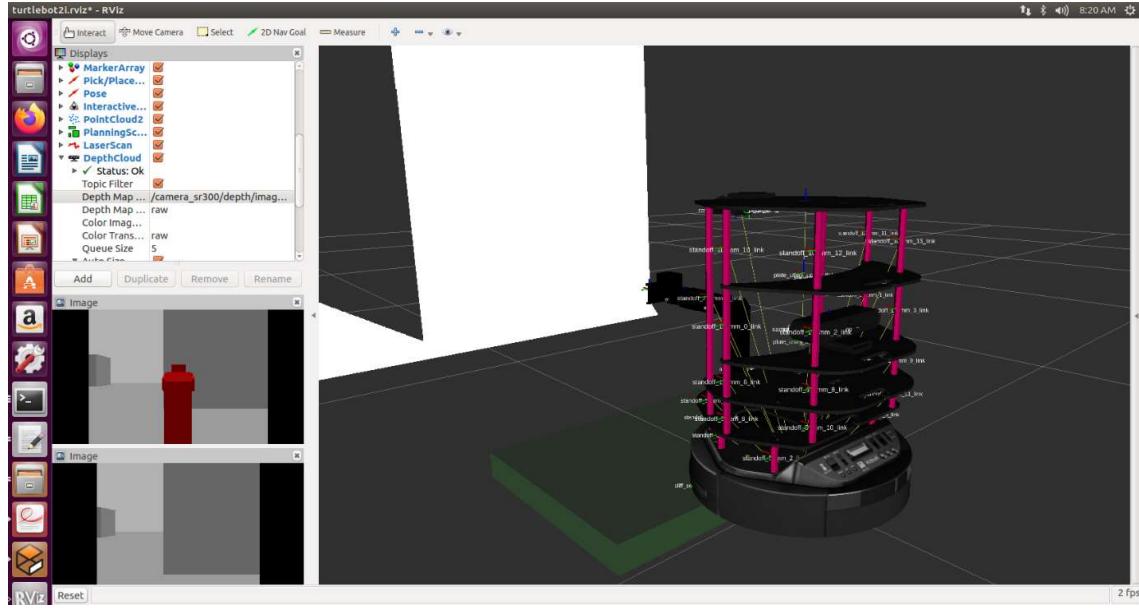
[WARN] [1583765382.759065365, 423.110000000]: Shutdown request received.
[WARN] [1583765382.759147694, 423.110000000]: Reason given for shutdown: [new node registered with same name]
[robot_state_publisher-6] process has finished cleanly
log file: /home/turtlebot2i-simulation/.ros/log/070f1ba2-6215-11ea-b0aa-000c298a6089/robot_state_publisher-6*.log
[mobile_base_nodelet_manager-7] process has finished cleanly
log file: /home/turtlebot2i-simulation/.ros/log/070f1ba2-6215-11ea-b0aa-000c298a6089/mobile_base_nodelet_manager-7*.log
[cmd_vel_mux-8] process has finished cleanly
log file: /home/turtlebot2i-simulation/.ros/log/070f1ba2-6215-11ea-b0aa-000c298a6089/cmd_vel_mux-8*.log
```

And turtlebot2i\_mapping.launch is showing this error as before

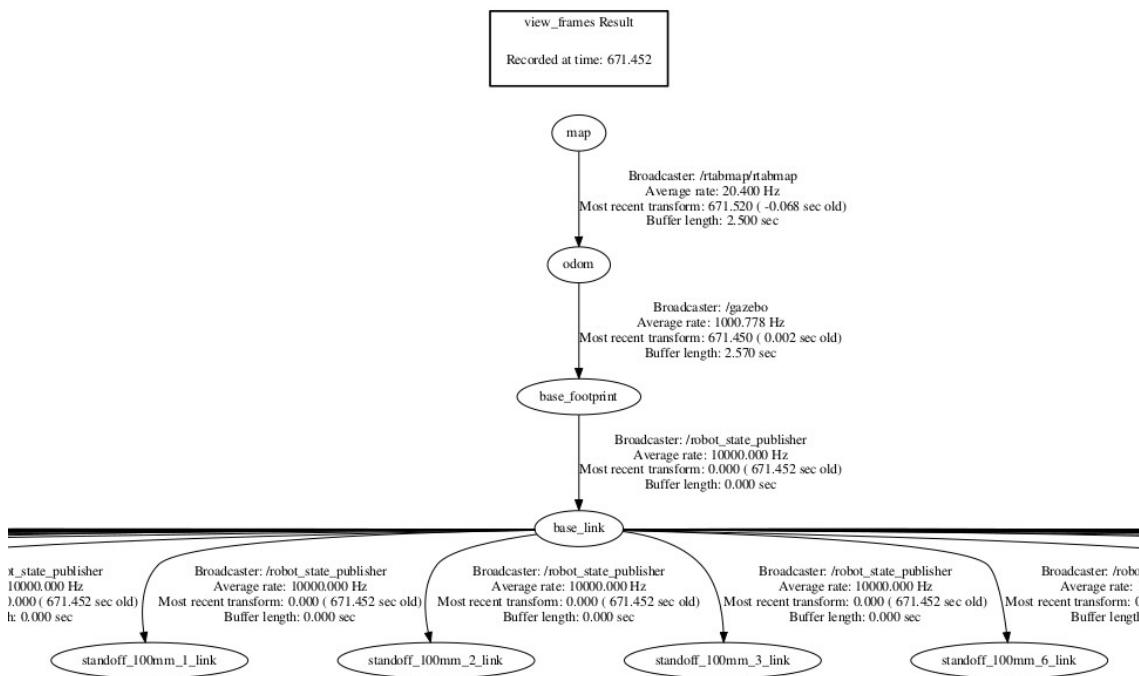
```
[ INFO] [1583765465.013387689, 462.287000000]: No matching device found.... waiting for devices. Reason: std::cxx11::string astra_wrapper::AstraDriver::resolveDeviceURI(const string@) @ /home/turtlebot2-simulation/turtlebot2/src/ros_astra_camera/src/astra_driver.cpp @ 709 : Invalid device number 1, there are 0 devices connected.
[ INFO] [1583765468.014161024, 463.823000000]: No matching device found.... waiting for devices. Reason: std::cxx11::string astra_wrapper::AstraDriver::resolveDeviceURI(const string@) @ /home/turtlebot2-simulation/turtlebot2/src/ros_astra_camera/src/astra_driver.cpp @ 709 : Invalid device number 1, there are 0 devices connected.
[ INFO] [1583765471.014933976, 465.434000000]: No matching device found.... waiting for devices. Reason: std::cxx11::string astra_wrapper::AstraDriver::resolveDeviceURI(const string@) @ /home/turtlebot2-simulation/turtlebot2/src/ros_astra_camera/src/astra_driver.cpp @ 709 : Invalid device number 1, there are 0 devices connected.
[ WARN] [1583765472.114239976, 465.934000000]: /rtabmap/rtabmapviz: Did not receive data since 5 seconds! Make sure the input topics are published ("$ rostopic hz my_topic") and the timestamps in their header are set. If topics are coming from different computers, make sure the clocks of the computers are synchronized ("ntpdate"). If topics are not published at the same rate, you could increase "queue_size" parameter (current=10).
/rtabmap/rtabmapviz subscribed to (approx sync):
/rtabmap/odom,
/rtabmap/rgb/image_rect_color,
/camera/depth_registered/image_raw,
/camera/rgb/camera_info,
/scan
[ WARN] [1583765473.337768997, 466.589000000]: /rtabmap/rtabmap: Did not receive data since 5 seconds! Make sure the input topics are published ("$ rostopic hz my_topic") and the timestamps in their header are set. If topics are coming from different computers, make sure the clocks of the computers are synchronized ("ntpdate"). If topics are not published at the same rate, you could increase "queue_size" parameter (current=10).
/rtabmap/rtabmapviz subscribed to (approx sync):
/rtabmap/rgb/image_raw,
/camera/depth_registered/image_raw,
/camera/rgb/camera_info,
/scan
[ INFO] [1583765474.015295917, 466.931000000]: No matching device found.... waiting for devices. Reason: std::cxx11::string astra_wrapper::AstraDriver::resolveDeviceURI(const string@) @ /home/turtlebot2-simulation/turtlebot2/src/ros_astra_camera/src/astra_driver.cpp @ 709 : Invalid device number 1, there are 0 devices connected.
[ INFO] [1583765477.015619766, 468.483000000]: No matching device found.... waiting for devices. Reason: std::cxx11::string astra_wrapper::AstraDriver::resolveDeviceURI(const string@) @ /home/turtlebot2-simulation/turtlebot2/src/ros_astra_camera/src/astra_driver.cpp @ 709 : Invalid device number 1, there are 0 devices connected.
[ INFO] [1583765480.016142363, 470.051000000]: No matching device found.... waiting for devices. Reason: std::cxx11::string astra_wrapper::AstraDriver::resolveDeviceURI(const string@) @ /home/turtlebot2-simulation/turtlebot2/src/ros_astra_camera/src/astra_driver.cpp @ 709 : Invalid device number 1, there are 0 devices connected.
[ WARN] [1583765481.78706335, 470.936000000]: /rtabmap/rtabmapviz: Did not receive data since 5 seconds! Make sure the input topics are published ("$ rostopic hz my_topic") and the timestamps in their header are set. If topics are coming from different computers, make sure the clocks of the computers are synchronized ("ntpdate"). If topics are not published at the same rate, you could increase "queue_size" parameter (current=10).
/rtabmap/rtabmapviz subscribed to (approx sync):
/rtabmap/odom,
/rtabmap/rgb/image_rect_color,
/camera/depth_registered/image_raw,
/camera/rgb/camera_info,
/scan
[ INFO] [1583765483.016959105, 471.583000000]: No matching device found.... waiting for devices. Reason: std::cxx11::string astra_wrapper::AstraDriver::resolveDeviceURI(const string@) @ /home/turtlebot2-simulation/turtlebot2/src/ros_astra_camera/src/astra_driver.cpp @ 709 : Invalid device number 1, there are 0 devices connected.
[ WARN] [1583765483.036501746, 471.589000000]: /rtabmap/rtabmap: Did not receive data since 5 seconds! Make sure the input topics are published ("$ rostopic hz my_topic") and the timestamps in their header are set. If topics are coming from different computers, make sure the clocks of the computers are synchronized ("ntpdate"). If topics are not published at the same rate, you could increase "queue_size" parameter (current=10).
```

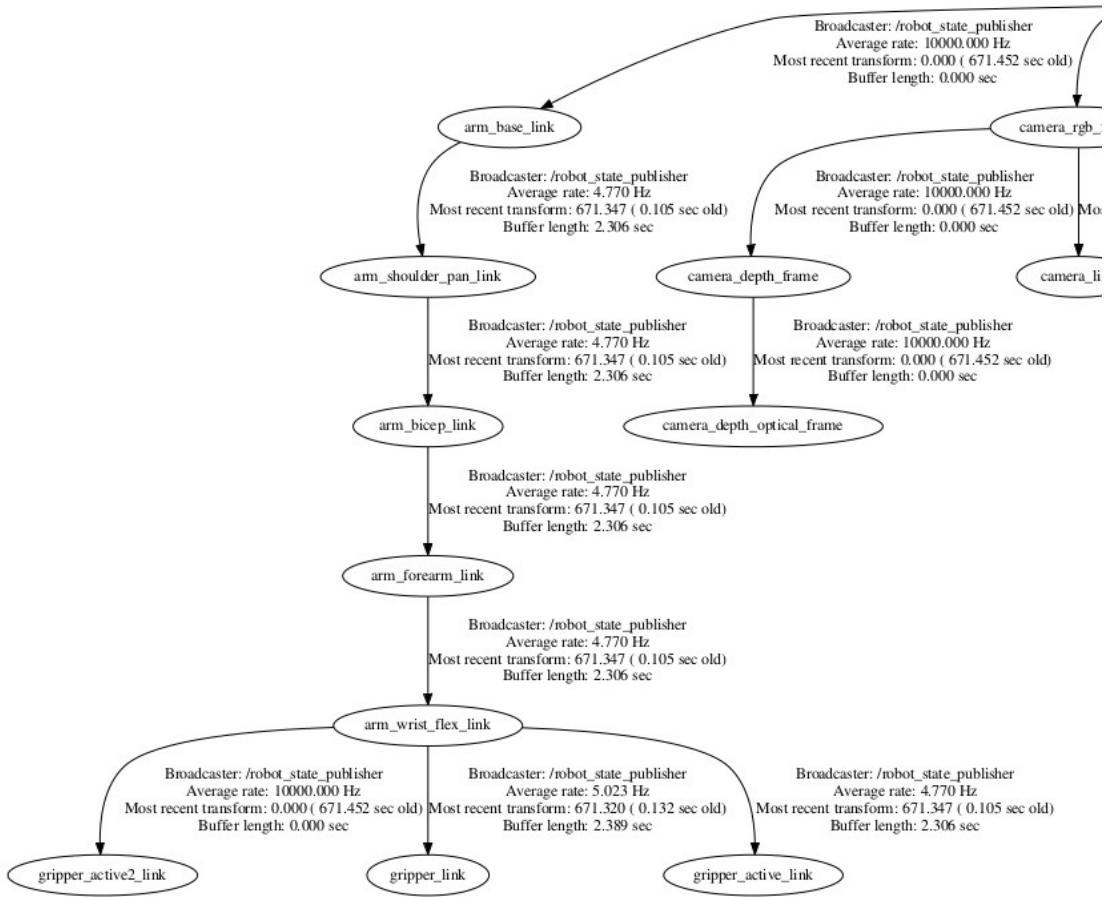
Screenshot of Rviz. There is no transform from any frame missing. rtabmapviz is still not showing image, which is reasonable



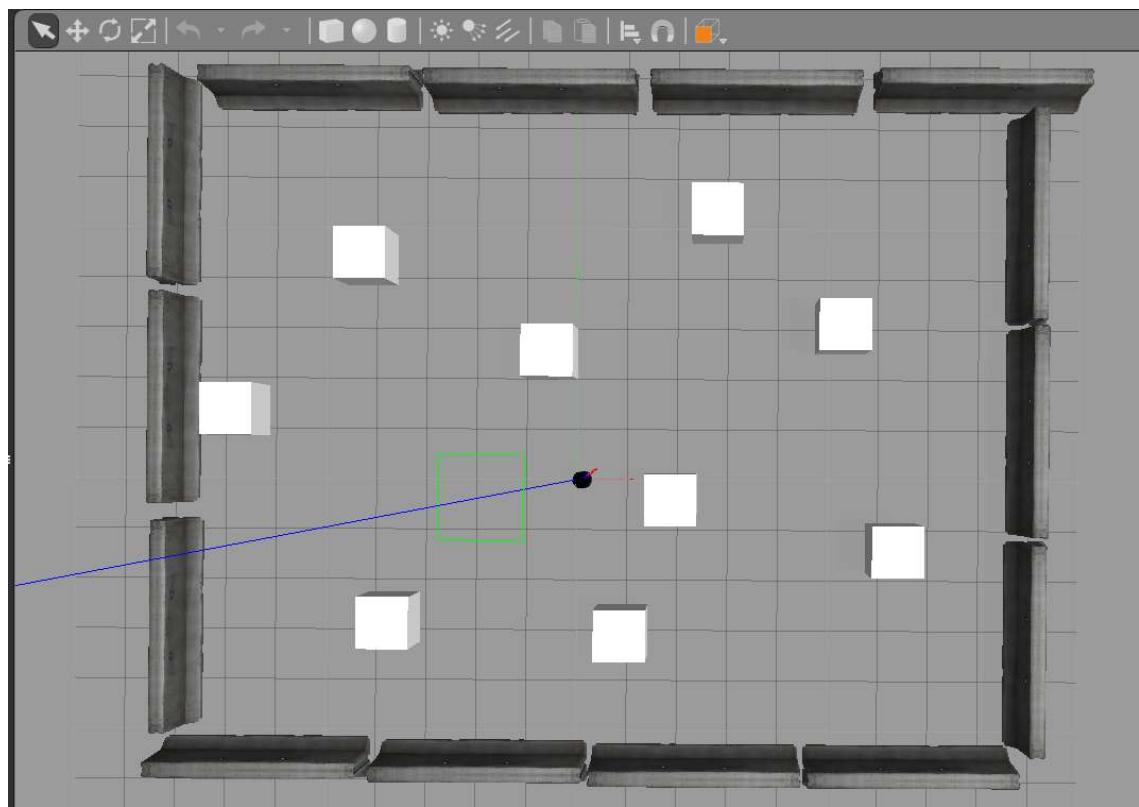


Screenshot of TF frames, every thing is connected





Robot is able to react to keyboard teleop



rostopic list

```
turtlebot2i-simulation@ubuntu:~$ rostopic list
/arm_bicep_joint/command
/arm_controller/command
/arm_controller/follow_joint_trajectory/cancel
/arm_controller/follow_joint_trajectory/feedback
/arm_controller/follow_joint_trajectory/goal
/arm_controller/follow_joint_trajectory/result
/arm_controller/follow_joint_trajectory/status
/arm_forearm_joint/command
/arm_shoulder_pan_joint/command
/arm_wrist_flex_joint/command
/attached_collision_object
/block_controls/feedback
/block_controls/update
/block_detection_action_server/block_output
/camera/depth/camera_info
/camera/depth/image
/camera/depth/image/compressed
/camera/depth/image/compressed/parameter_descriptions
/camera/depth/image/compressed/parameter_updates
/camera/depth/image/compressedDepth
/camera/depth/image/compressedDepth/parameter_descriptions
/camera/depth/image/compressedDepth/parameter_updates
/camera/depth/image_raw
/camera/depth/image_rect
/camera/depth/image_rect/compressed
/camera/depth/image_rect/compressed/parameter_descriptions
/camera/depth/image_rect/compressed/parameter_updates
/camera/depth/image_rect/compressedDepth
/camera/depth/image_rect/compressedDepth/parameter_descriptions
/camera/depth/image_rect/compressedDepth/parameter_updates
/camera/depth/points
/camera/depth_registered/image_raw
/camera/ir/image_rect_ir
/camera/ir/image_rect_ir/compressed
/camera/ir/image_rect_ir/compressed/parameter_descriptions
/camera/ir/image_rect_ir/compressed/parameter_updates
/camera/ir/image_rect_ir/compressedDepth
/camera/ir/image_rect_ir/compressedDepth/parameter_descriptions
/camera/ir/image_rect_ir/compressedDepth/parameter_updates
/camera/ir_rectify_ir/parameter_descriptions
/camera/ir_rectify_ir/parameter_updates
/camera/parameter_descriptions
/camera/parameter_updates
```

```
/camera/parameter_descriptions
/camera/parameter_updates
/camera/rgb/camera_info
/camera/rgb/image_raw
/camera/rgb/image_raw/compressed
/camera/rgb/image_raw/compressed/parameter_descriptions
/camera/rgb/image_raw/compressed/parameter_updates
/camera/rgb/image_raw/compressedDepth
/camera/rgb/image_raw/compressedDepth/parameter_descriptions
/camera/rgb/image_raw/compressedDepth/parameter_updates
/camera/rgb/image_rect_color
/camera_sr300/depth/camera_info
/camera_sr300/depth/image_raw
/camera_sr300/depth/points
/camera_sr300/parameter_descriptions
/camera_sr300/parameter_updates
/camera_sr300/rgb/camera_info
/camera_sr300/rgb/image_raw
/camera_sr300/rgb/image_raw/compressed
/camera_sr300/rgb/image_raw/compressed/parameter_descriptions
/camera_sr300/rgb/image_raw/compressed/parameter_updates
/camera_sr300/rgb/image_raw/compressedDepth
/camera_sr300/rgb/image_raw/compressedDepth/parameter_descriptions
/camera_sr300/rgb/image_raw/compressedDepth/parameter_updates
/camera_sr300/rgb/image_rect_color
/clock
/cmd_vel_mux/active
/cmd_vel_mux/input/navi
/cmd_vel_mux/input/safety_controller
/cmd_vel_mux/input/switch
/cmd_vel_mux/input/teleop
/cmd_vel_mux/parameter_descriptions
/cmd_vel_mux/parameter_updates
/collision_object
/diagnostics
/diagnostics_agg
/diagnostics_toplevel_state
/dock_drive/debug/feedback
/dock_drive/debug	mode_shift
/dock_drive_action/cancel
/dock_drive_action/feedback
```

```
/collision_object
/diagnostics
/diagnostics_agg
/diagnostics_toplevel_state
/dock_drive/debug/feedback
/dock_drive/debug/mode_shift
/dock_drive_action/cancel
/dock_drive_action/feedback
/dock_drive_action/goal
/dock_drive_action/result
/dock_drive_action/status
/dock_drive_manager/bond
/execute_trajectory/cancel
/execute_trajectory/feedback
/execute_trajectory/goal
/execute_trajectory/result
/execute_trajectory/status
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/set_link_state
/gazebo/set_model_state
/gazebo_gui/parameter_descriptions
/gazebo_gui/parameter_updates
/gripper_controller/gripper_action/cancel
/gripper_controller/gripper_action/feedback
/gripper_controller/gripper_action/goal
/gripper_controller/gripper_action/result
/gripper_controller/gripper_action/status
/gripper_joint/command
/gripper_pose
/joint_states
/kobuki_safety_controller/disable
/kobuki_safety_controller/enable
/kobuki_safety_controller/reset
/map
/map_updates
/mobile_base/commands/controller_info
/mobile_base/commands/digital_output
/mobile_base/commands/external_power
/mobile_base/commands/led1
/mobile_base/commands/led2
/mobile_base/commands/motor_power
/mobile_base/commands/reset_odometry
/mobile_base/commands/sound
/mobile_base/commands/velocity
/mobile_base/controller_info
```

03/10

I found that there is another issue I need to fix, "no matching device found"

```
[INFO] [1583848755.239016730, 409.0510000000]: No matching device found... waiting for devices. Reason: std::string astral_wrapper::AstralDriver::resolveDeviceURI(const string) @ /home/turtlebot2i-simulation/turtlebot2i/src/ros_astar_camera/src/astar_driver.cpp @ 709 : Invalid device number 1, there are 0 devices connected.
[WARN] [1583848757.886675960, 410.3540000000]: /rtabmap/rtabmapviz: Did not receive data since 5 seconds! Make sure the input topics are published ("$ rostopic my_topic") and the timestamps in their header are set. If topics are coming from different computers, make sure the clocks of the computers are synchronized ("ntpdate"). If topics are not published at the same rate, you could increase "queue_size" parameter (current=10).
[INFO] [1583848757.886675960, 410.3540000000]: /rtabmap/rtabmapviz subscribed to (approx sync):
/rtabmap/odom,
/camera/gb/image_rect_color,
/camera/depth_registered/image_raw,
/camera/gb/camera_info,
/scan
```

I think now I need to disable the calling for physical device and check whether these topics exists and have valid messages published to them.

First find where the physical camera is called and comment it.

Wait a second, now I have the whole simulated robot setted up. What's the point of running `turtlebot2i_mapping.launch`, which contains some contents for real robot?

What I need is just to create a map, why not running `rtabmap.launch` solely?

It seems that it doesn't work, errors as follow, the same as previous ones

```
[ INFO] [1583850300.961124359, 429.152000000]: No matching device found.... waiting for devices. Reason: std::cxx11::string astra_wrapper::AstraDriver::resolveDeviceURI(const string&)
[ INFO] @ /home/turtlebot2i-simulation/turtlebot2i/src/ros_astra_camera/src/astra_driver.cpp @ 709 : Invalid device number 1, there are 0 devices connected.
[ WARN] [1583850301.91150195, 429.845000000]: /rtabmap/rtabmap: Did not receive data since 5 seconds! Make sure the input topics are published ("rostopic hz my_topic") and the timestamps in their header are set. If topics are coming from different computers, make sure the clocks of the computers are synchronized ("ntpdate"). If topics are not published at the same rate, you could increase "queue_size" parameter (current=10).
/rtabmap/rtabmap subscribed to (approx sync):
/camera/gb/image_raw,
/camera/depth_registered/image_raw,
/camera/gb/camera_info,
/scan
[ INFO] [1583850303.961802783, 431.264000000]: No matching device found.... waiting for devices. Reason: std::cxx11::string astra_wrapper::AstraDriver::resolveDeviceURI(const string&)
[ INFO] @ /home/turtlebot2i-simulation/turtlebot2i/src/ros_astra_camera/src/astra_driver.cpp @ 709 : Invalid device number 1, there are 0 devices connected.
[ INFO] [1583850306.962062421, 433.428000000]: No matching device found.... waiting for devices. Reason: std::cxx11::string astra_wrapper::AstraDriver::resolveDeviceURI(const string&)
[ WARN] [1583850306.962062421, 433.428000000]: /rtabmap/rtabmapvizi: Did not receive data since 5 seconds! Make sure the input topics are published ("rostopic hz my_topic") and the timestamps in their header are set. If topics are coming from different computers, make sure the clocks of the computers are synchronized ("ntpdate"). If topics are not published at the same rate, you could increase "queue_size" parameter (current=10).
/rtabmap/rtabmapvizi subscribed to (approx sync):
/rtabmap/odom,
/camera/gb/image_rect_color,
/camera/depth_registered/image_raw,
/camera/gb/camera_info,
/scan
```

I found something new when referring to tutorials for rtabmap on ROS Wiki.

[http://wiki.ros.org/rtabmap\\_ros/Tutorials/SetupOnYourRobot#Kinect .2B- Odometry .2B-Fake 2D laser from Kinect](http://wiki.ros.org/rtabmap_ros/Tutorials/SetupOnYourRobot#Kinect .2B- Odometry .2B-Fake 2D laser from Kinect)

[http://wiki.ros.org/rtabmap\\_ros/Tutorials/MappingAndNavigationOnTurtlebot](http://wiki.ros.org/rtabmap_ros/Tutorials/MappingAndNavigationOnTurtlebot)

It seems that I need to run a separate node to generate messages published to /scan topic

Check whether there are messages published to subscribed topics again,

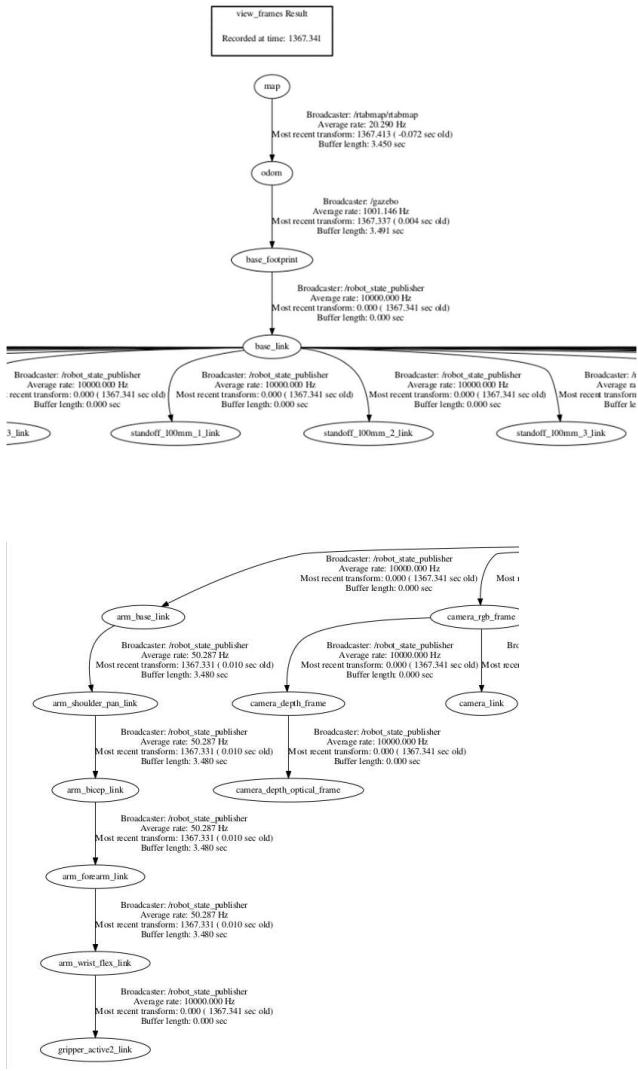
```
[ INFO] [1583851010.836888918, 965.474000000]: No matching device found.... waiting for devices. Reason: std::cxx11::string astra_wrapper::AstraDriver::resolveDeviceURI(const string&)
[ INFO] @ /home/turtlebot2i-simulation/turtlebot2i/src/ros_astra_camera/src/astra_driver.cpp @ 709 : Invalid device number 1, there are 0 devices connected.
[ INFO] [1583851013.837396950, 967.617000000]: No matching device found.... waiting for devices. Reason: std::cxx11::string astra_wrapper::AstraDriver::resolveDeviceURI(const string&)
[ WARN] [1583851013.837396950, 967.617000000]: /rtabmap/rtabmap: Did not receive data since 5 seconds! Make sure the input topics are published ("rostopic hz my_topic") and the timestamps in their header are set. If topics are coming from different computers, make sure the clocks of the computers are synchronized ("ntpdate"). If topics are not published at the same rate, you could increase "queue_size" parameter (current=10).
/rtabmap/rtabmapvizi subscribed to (approx sync):
/rtabmap/odom,
/camera/gb/image_rect_color,
/camera/depth_registered/image_raw,
/camera/gb/camera_info,
/scan
[ WARN] [1583851016.668521645, 969.678000000]: /rtabmap/rtabmap: Did not receive data since 5 seconds! Make sure the input topics are published ("rostopic hz my_topic") and the timestamps in their header are set. If topics are coming from different computers, make sure the clocks of the computers are synchronized ("ntpdate"). If topics are not published at the same rate, you could increase "queue_size" parameter (current=10).
/rtabmap/rtabmapvizi subscribed to (approx sync):
/camera/gb/image_raw,
/camera/depth_registered/image_raw,
/camera/gb/camera_info,
/scan
```

Topics in red rectangles don't have messages published to them.

I think /rtanmap/odom could be visual odometry staff, because I didn't enable the following tag

```
rtabmap.launch (~/turtlebot2i/src/turtlebot2i_launch) - gedit
Open ▾ Save
standalone.launch | turtlebot2i_controlLaunch | mobile_base.launch.xml | rtabmap.launch >
<?xml version="1.0"?>
<launch>
  <arg name="database_path" default="rtabmap.db"/>
  <arg name="localization" default="false"/>
  <arg name="rgbd_odometry" default="false"/>
  <arg name="args" default="">
  <arg name="rtabmapviz" default="false"/>
  <arg name="wait_for_transform" default="0.2"/>
```

I noticed the TF error didn't show up just now, so I check the TF tree, and TF is in good shape. I think it may because I didn't run turtlebot2i\_mapping.launch, so the TF is not redefined.



Now I think I'm going to do some modification to `rtabmap.launch`, adding the `/scan` part and remap `/camera/depth_registered/image_raw`

Accidentally see there is Simulation part in this mapping and navigation tutorial for rtabmap, [http://wiki.ros.org/rtabmap\\_ros/Tutorials/MappingAndNavigationOnTurtlebot](http://wiki.ros.org/rtabmap_ros/Tutorials/MappingAndNavigationOnTurtlebot), so I decided to add the `/scan` part in `turtlebot2i_world.launch`

```

<!-- ros_control turtlebot launch file -->
<include file="$(find turtlebot2i_control)/launch/turtlebot2i_control.launch" />

<!-- Velocity muxer -->
<node pkg="nodelet" type="nodelet" name="mobile_base_nodelet_manager" args="manager"/>
<node pkg="nodelet" type="nodelet" name="cmd_vel_mux"
    args="load yocs_cmd_vel_mux/CmdvelMuxNodelet mobile_base_nodelet_manager">
    <param name="yaml_cfg_file" value="$(find turtlebot2i_bringup)/param/mux.yaml" />
    <remap from="cmd_vel_mux/output" to="mobile_base/commands/velocity"/>
</node>

<!-- Fake laser -->
<node pkg="depthimage_to_laserscan" type="depthimage_to_laserscan"
name="depthimage_to_laserscan">
    <param name="scan_height" value="10"/>
    <param name="output_frame_id" value="camera_depth_frame"/>
    <param name="range_min" value="0.45"/>
    <remap from="image" to="/camera/depth/image_raw"/>
    <remap from="scan" to="/scan"/>
</node>

</launch>

```

After running turtlebot2i\_world.launch(self created), the /scan doesn't have message published to it. When I restart(maybe) and check things for a while, /scan works well.

Now I started turtlebot2i\_bringup/rtabmap.launch, and it still shows no matching device found and the orange no message published to topic error. I think maybe I run rtabmap\_ros/rtabmap.launch will solve this issue? And I check that file, but is doesn't show much difference.

```

[ INFO] [1583852927.815610610, 948.649688000]: No matching device found... waiting for devices. Reason: std::cxx11::string astra_wrapper::AstraDriver::resolveDeviceURI(const string&)
[ INFO] [1583852930.816216993, 950.314080000]: No matching device found... waiting for devices. Reason: std::cxx11::string astra_wrapper::AstraDriver::resolveDeviceURI(const string&)
[ INFO] [1583852932.729782833, 951.796000000]: /rtabmap/rtabmap: Did not receive data since 5 seconds! Make sure the input topics are published ("5 rostopic hz my_topic") and the timestamp in their header are set. If topics are coming from different computers, make sure the clocks of the computers are synchronized ("ntpdate"). If topics are not published at the same rate, you could increase "queue_size" parameter (current=10).
/rtabmap/rtabmap subscribed to (approx sync):
/camera/rgb/image_raw,
/camera/depth_registered/image_raw,
/camera/rgb/camera_info,
/scan
[ INFO] [1583852933.816461096, 952.561080000]: No matching device found... waiting for devices. Reason: std::cxx11::string astra_wrapper::AstraDriver::resolveDeviceURI(const string&)
[ INFO] [1583852936.816216993, 953.847080000]: No matching device found... waiting for devices. Reason: std::cxx11::string astra_wrapper::AstraDriver::resolveDeviceURI(const string&)
[ INFO] [1583852938.761780574, 955.796000000]: /rtabmap/rtabmap: Did not receive data since 5 seconds! Make sure the input topics are published ("5 rostopic hz my_topic") and the timestamp in their header are set. If topics are coming from different computers, make sure the clocks of the computers are synchronized ("ntpdate"). If topics are not published at the same rate, you could increase "queue_size" parameter (current=10).
/rtabmap/rtabmap subscribed to (approx sync):
/camera/rgb/image_raw,
/camera/depth_registered/image_raw,
/camera/rgb/camera_info,
/scan

```

So, now I think I should continue with turtlebot2i\_bringup/rtabmap.launch and finish this direction of fixing.

I commented 2 lines and add 1 line. I think the launch file is designed for zr300 camera instead of sr300 camera, and sr300 is publishing the topic in the rectangle, besides these are inputs of rtabmap(black box), so I made modifications as shown in the screenshot.

```

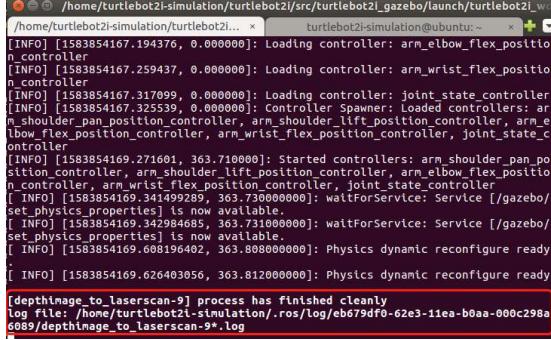
<!-- inputs -->
<remap from="scan" to="/scan"/>
<!-- TODO: zr300: remap from="rgb/image" to="/camera/rgb/
image_rect_color"/>-->
<remap if="$(eval $arg('3d_sensor') != 'zr300')" from="depth/image" to="/camera/
depth_registered/image_raw"/>-->
<remap if="$(eval $arg('3d_sensor') == 'zr300')" from="depth/image" to="/camera/
depth/image_raw"/>-->
<remap if="true" from="depth/image" to="/camera/depth/image_raw"/>
<remap from="rgb/camera_info" to="/camera/rgb/camera_info"/>

<!-- output -->
<remap from="grid_map" to="/map"/>

<!-- RTAB-MAP's parameters: do "rosmaster rtabmap rtabmap (double-dash)params" to see the list of available parameters. -->
<param name="RGBD/ProximityBy5Space" type="string" value="true"/>
<!-- Local loop closure detection (using estimated position) with locations in WM -->
<param name="RGBD/optimizeFromGraphEnd" type="string" value="false"/>

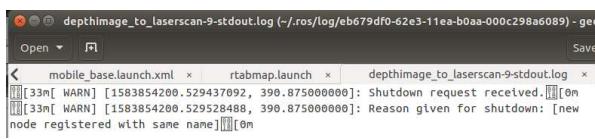
```

I found `depthimage_to_laserscan` will die after I ran `rtabmap.launch`. So I'm going to check the log file.



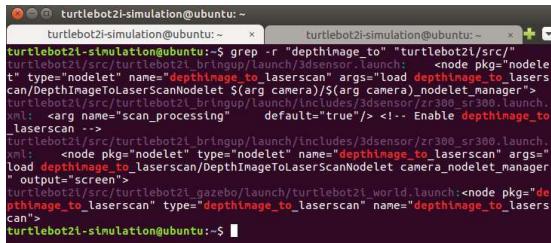
```
[INFO] [1583854167.194376, 0.000000]: Loading controller: arm_elbow_flex_position_controller
[INFO] [1583854167.259437, 0.000000]: Loading controller: arm_wrist_flex_position_controller
[INFO] [1583854167.317099, 0.000000]: Loading controller: joint_state_controller
[INFO] [1583854167.325539, 0.000000]: Controller Spawner: Loaded controllers: arm_shoulder_pan_flex_position_controller, arm_shoulder_lift_position_controller, arm_elbow_flex_position_controller, arm_wrist_flex_position_controller, joint_state_controller
[INFO] [1583854169.271601, 363.710000]: Started controllers: arm_shoulder_pan_position_controller, arm_shoulder_lift_position_controller, arm_elbow_flex_position_controller, arm_wrist_flex_position_controller, joint_state_controller
[INFO] [1583854169.341499289, 363.730000000]: waitForService: Service [/gazebo/set_physics_properties] is now available.
[INFO] [1583854169.342984685, 363.731000000]: waitForService: Service [/gazebo/set_physics_properties] is now available.
[INFO] [1583854169.608196402, 363.808000000]: Physics dynamic reconfigure ready
[INFO] [1583854169.626403056, 363.812000000]: Physics dynamic reconfigure ready
[depthimage_to_laserscan-9] process has finished cleanly
log file: /home/turtlebot2i-simulation/.ros/log/eb679df0-62e3-11ea-b0aa-000c298a6089/depthimage_to_laserscan-9*.log
```

After checking the log file, it turns out that the node is shut down because there is a node with the same name created.



```
[33m[ WARN] [1583854200.529437092, 390.875000000]: Shutdown request received.
[33m[ WARN] [1583854200.529528488, 390.875000000]: Reason given for shutdown: [new node registered with same name]
```

So I checked where the second time this node is created, these are the result(may not complete).



```
turtlebot2i-simulation@ubuntu:~$ grep -r "depthimage_to" "turtlebot2i/src/"
turtlebot2i/src/turtlebot2i_bringup/launch/3dsensor.launch: <node pkg="nodelet" type="nodelet" name="depthimage_to_laserscan" args="load #depthimage_to_laserscanNodelet $(arg camera)/$(arg camera)_nodelet_manager" launch_xml_file="$(find turtlebot2i_bringup/launch/3dsensor.launch.xml)">
<arg name="scan_processing" default="true"/> <!-- Enable depthimage_to_laserscan -->
turtlebot2i/src/turtlebot2i_bringup/launch/dual_3dsensor.launch:
<node pkg="nodelet" type="nodelet" name="depthimage_to_laserscan" args="load depthimage_to_laserscan/DepthImageToLaserScanNodelet camera_nodelet_manager" output="screen">
<node pkg="turtlebot2i_gazebo" type="depthimage_to_laserscan" name="depthimage_to_laserscan" args="load depthimage_to_laserscan/DepthImageToLaserScanNodelet camera_nodelet_manager" output="screen">
```

So I think `turtlebot2i_bringup/rtabmap.launch` may include some of files listed above.

So I checked it, and find it called `dual_3dsensor.launch`. So I opened it.

```

<arg name="3d_sensor" default="$(env TURTLEBOT_3D_SENSOR)" />
<arg name="3d_sensor2" default="$(optenv TURTLEBOT_3D_SENSOR2 None)" />

<!-- 3D Sensor(s) -->
<include if="$(eval arg('3d_sensor') != 'zr300')" file="$(find turtlebot2i_bringup)/
launch/3dsensor.launch"/>
<include if="$(eval arg('3d_sensor') == 'zr300' and arg('3d_sensor2') == 'sr300')"
file="$(find turtlebot2i_bringup)/launch/includes/3dsensor/zr300_sr300.launch.xml"/>
<include if="$(eval arg('3d_sensor') != 'zr300' and arg('3d_sensor2') != 'None')"
file="$(find turtlebot2i_bringup)/launch/includes/3dsensor/$(arg
3d_sensor).launch.xml"/>
<arg name="launch_camera" value="False" />
<arg name="manager" value="/camera/camera_nodelet_manager" />
</include>

</launch>

```

After looking for cascaded files for a while, I found the `depthimage_to_laserscan` node is created here.

```

<arg name="depth_processing" value="$(arg depth_processing)"/>
<arg name="depth_registered_processing" value="$(arg
depth_registered_processing)"/>
<arg name="disparity_processing" value="$(arg disparity_processing)"/>
<arg name="disparity_registered_processing" value="$(arg
disparity_registered_processing)"/>
</include>

<!--
      Laserscan
      This uses lazy subscribing, so will not activate until scan is requested.
-->
<group if="$(arg scan_processing)">
<node pkg="nodelet" type="nodelet" name="depthimage_to_laserscan" args="load
depthimage_to_laserscan/DepthImageToLaserScanNodelet $(arg camera)/$(arg
camera)_nodelet_manager">
    <!-- Pixel rows to use to generate the laserscan. For each column, the scan will
        return the minimum value for those pixels centered vertically in the image.
-->
    <param name="scan_height" value="10"/>
    <param name="output_frame_id" value="$(arg camera)_depth_frame"/>
    <param name="range_min" value="0.45"/>
    <remap from="image" to="$(arg camera)/$(arg depth)/image_raw"/>
    <remap from="scan" to="$(arg scan_topic)"/>
    <!-- Somehow topics here get prefixed by "$(arg camera)" when not inside an app
namespace,
        so in this case "$(arg scan_topic)" must provide an absolute topic name
-->
</node>
</group>

```

So I will comment the one I added in `turtlebot2i_world.launch`, and try again.

/scan still have no message published to it. I found something novel in the

### 3dsensor.launch

```

<!--
      Laserscan
      This uses lazy subscribing, so will not activate until scan is requested.
-->
<group if="$(arg scan_processing)">
<node pkg="nodelet" type="nodelet" name="depthimage_to_laserscan" args="load
depthimage_to_laserscan/DepthImageToLaserScanNodelet $(arg camera)/$(arg
camera)_nodelet_manager">
    <!-- Pixel rows to use to generate the laserscan. For each column, the scan will
        return the minimum value for those pixels centered vertically in the image.
-->
    <param name="scan_height" value="10"/>
    <param name="output_frame_id" value="$(arg camera)_depth_frame"/>
    <param name="range_min" value="0.45"/>
    <remap from="image" to="$(arg camera)/$(arg depth)/image_raw"/>
    <remap from="scan" to="$(arg scan_topic)"/>
    <!-- Somehow topics here get prefixed by "$(arg camera)" when not inside an app
namespace,
        so in this case "$(arg scan_topic)" must provide an absolute topic name
(issue #88) Probably a bug in the nodelet manager: https://github.com/ros/
nodelet_core/issues/7 -->
    <remap from="$(arg camera)/image" to="$(arg camera)/$(arg depth)/image_raw"/>
    <remap from="$(arg camera)/scan" to="$(arg scan_topic)"/>
</node>
</group>
</launch>

```

After googling for minutes, I almost find nothing related to this, so I decide to comment this part and uncomment the part in turtlebot2i\_world.launch, as it works well before.

Now able to see 2D map, I modified this to see what rtabmapviz can show me.

```
<?xml version="1.0"?>
<launch>
  <arg name="database_path" default="rtabmap.db"/>
  <arg name="localization" default="false"/>
  <arg name="rgbd_odometry" default="false"/>
  <arg name="args" default="" />
  <arg name="rtabmapviz" default="true"/>
```

In order to keep Rviz configuration, I added 1 line at the end of

turtlebot2i\_world.launch

```
<!-- ros_control turtlebot launch file -->
<include file="$(find turtlebot2i_control)/launch/turtlebot2i_control.launch" />

<!-- Velocity muxer -->
<node pkg="nodelet" type="nodelet" name="mobile_base_nodelet_manager" args="manager"/>
<node pkg="nodelet" type="nodelet" name="cmd_vel_mux"
  args="Load yocs_cmd_vel_mux/CmdvelMuxNodelet mobile_base_nodelet_manager">
  <param name="yaml_cfg_file" value="$(find turtlebot2i_bringup)/param/mux.yaml" />
  <remap from="cmd_vel_mux/output" to="mobile_base/commands/velocity"/>
</node>

<!-- Fake laser -->
<node pkg="depthimage_to_laserscan" type="depthimage_to_laserscan"
  name="depthimage_to_laserscan">
  <param name="scan_height" value="10"/>
  <param name="output_frame_id" value="camera_depth_frame"/>
  <param name="range_min" value="0.45"/>
  <remap from="image" to="/camera/depth/image_raw"/>
  <remap from="scan" to="/scan"/>
</node>

<!-- Rviz Configuration -->
<include if="$(arg rviz)" file="$(find turtlebot2i_bringup)/launch/rviz.launch"/>
```

In order to view things correctly in rtabmapviz, I modified rtabmapviz part of rtabmap.launch.

```

<!-- visualization with rtabmapviz -->
<node name="rtabmapviz" pkg="rtabmap_ros" type="rtabmapviz" name="rtabmapviz"
args="-d $(find rtabmap_ros)/Launch/config/rgbd_gui.ini" output="screen">
  <param name="subscribe_depth" type="bool" value="true"/>
  <param name="subscribe_scan" type="bool" value="true"/>
  <param name="frame_id" type="string" value="base_footprint"/>
  <param name="wait_for_transform_duration" type="double" value="${arg
wait_for_transform}"/>
  <remap from="rgb/image" to="/camera/rgb/image_rect_color"/>
<!--
  <remap if="$(eval $arg('3d_sensor')) != 'sr300'" from="depth/image" to="/camera/
depth_registered/image_raw"/>
  <remap if="$(eval $arg('3d_sensor')) == 'sr300'" from="depth/image" to="/camera/
depth/image_raw"/>-->
  <remap if="true" from="depth/image" to="/camera/depth/image_raw"/>
  <remap from="rgb/camera_info" to="/camera/rgb/camera_info"/>
  <remap from="scan" to="/scan"/>
</node>

```

Now make a github copy for current environment, and next step is to change the camera to camera\_sr300, as the astra camera is partially block by the arm.

Reboot and run turtlebot2i\_gazebo.launch again, error prompts "arg rviz should be provided"

After inspection of that file, I made modification as follows:

```

<!-- Load the URDF into the ROS Parameter Server -->
<arg name="urdf_file" default="$(find xacro)/xacro --inorder '$(find turtlebot2i_description)/
robots/$(arg base)_$(arg stacks)_$(arg 3d_sensor).urdf.xacro' />
<param name="robot_description" command="$(arg urdf_file)" />

<!-- Run a python script to send a service call to gazebo_ros to spawn a URDF robot -->
<node pkg="urdf_spawner" pkg="gazebo_ros" type="spawn_model" respawn="false" output="screen"
args="--urdf -model turtlebot -param robot_description"/>

<!-- ros_control turtlebot launch file -->
<include file="$(find turtlebot2i_control)/launch/turtlebot2i_control.launch" />

<!-- Velocity muxer -->
<node pkg="nodelet" type="nodelet" name="mobile_base_nodelet_manager" args="manager"/>
<node pkg="nodelet" type="nodelet" name="cmd_vel_mux">
  <param name="Load_yocs_cmd_vel_mux_CmdvelMuxNodelet mobile_base_nodelet_manager">
    <param name="yaml_cfg_file" value="$(find turtlebot2i_bringup)/param/mux.yaml" />
  <remap from="cmd_vel_mux/output" to="mobile_base/commands/velocity"/>
</node>

<!-- Fake laser -->
<node pkg="depthimage_to_laserscan" type="depthimage_to_laserscan" name="depthimage_to_laserscan">
  <param name="scan_height" value="1000"/>
  <param name="output_frame_id" value="camera_depth_frame"/>
  <param name="range_min" value="0.45"/>
  <remap from="image" to="/camera/depth/image_raw"/>
  <remap from="scan" to="/scan"/>
</node>

<!-- Rviz Configuration -->
<include if="true" file="$(find turtlebot2i_bringup)/launch/rviz.launch"/>

```

Now restart it again. I found although Rviz subscribes some topics, Rviz received no message from those topics. So I think this may be caused by Gazebo being paused at the beginning. So I made modification to turtlebot2i\_world.launch

```

rtabmap.launch x turtlebot2i_world.launch x rtabmap.launch x turtlebot2i_mapping.launch x
<launch>
  ... these are the arguments you can pass this launch file, for example paused:=true -->
  <arg name="paused" default="false"/>
  <arg name="use_sim_time" default="true"/>
  <arg name="gui" default="true"/>
  <arg name="headless" default="false"/>
  <arg name="debug" default="false"/>

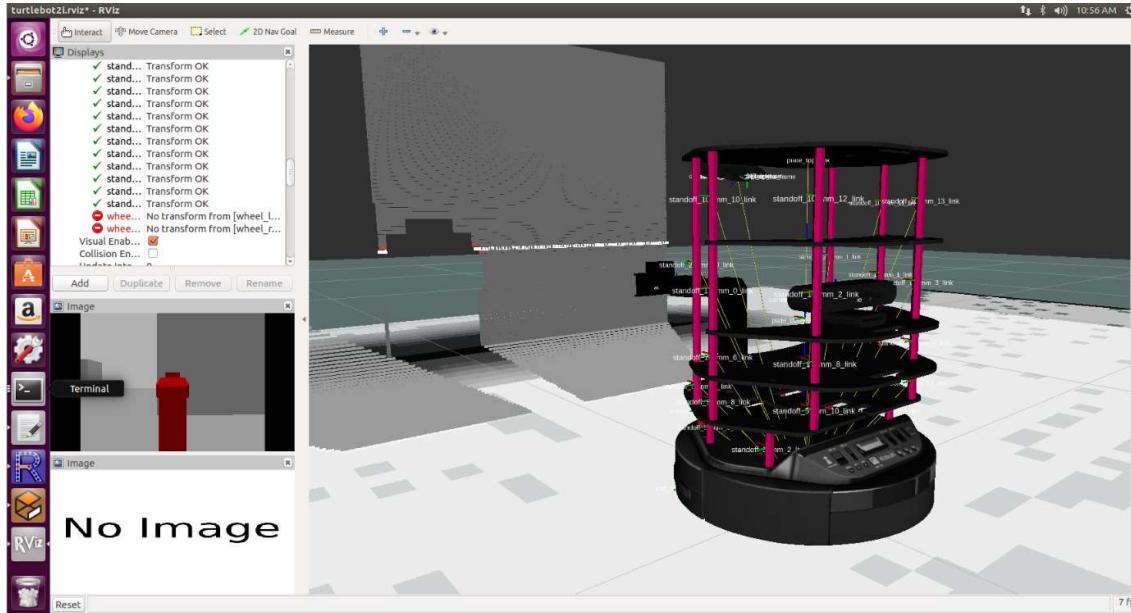
```

Run the two launch files again, rtabmap.launch shows this error. It seems that the program is trying to find the database, which is the previous map. I remember there is a flag in the launch file that can disable this behavior and start generating a new map. And wheel frames and gripper frames are missing again.

```

[ INFO] [1583862525.500070072, 375.441000000]: RTAB-Map detection rate = 1.00000
[ INFO] [1583862525.500335338, 375.445000000]: rtabmap: Using database from "/home/turtlebot2i-simulation/.ros/rtabmap.db" (206 MB).
[ERROR] [2020-03-10 10:48:45.880] VWDictionary.cpp:672::addWordRef() Not Found = ord 443 (dict size=431)
[ERROR] [2020-03-10 10:48:45.880] VWDictionary.cpp:672::addWordRef() Not Found = ord 940 (dict size=431)
[ERROR] [2020-03-10 10:48:45.880] VWDictionary.cpp:672::addWordRef() Not Found = ord 945 (dict size=431)

```



```

<?xml version="1.0"?>
<launch>
    <arg name="new_rtabmap" default="true"/>
    <arg name="localization" default="false"/>
    <arg name="rviz" default="true"/>
    <arg name="rtabmapviz" default="true"/>

    <include file="$(find turtlebot2i_bringup)/launch/minimal.launch"/>
    <include file="$(find turtlebot2i_bringup)/launch/arms/phantomx_pincher_arm.launch"/>
    <include file="$(find turtlebot2i_arm_director)/launch/named_pose_director.launch">
        <arg name="load_movel" value="true"/>
    </include>

    <include file="$(find turtlebot2i_bringup)/launch/rtabmap.launch">
        <arg if="$(arg new_rtabmap)" name="args" value="--delete_db_on_start"/>
        <arg name="localization" value="$(arg args)"/>
        <arg name="rtabmapviz" value="$(arg rtabmapviz)"/>
    </include>

```

So I think maybe I can do this kind of cascaded calling of launch files and parameter passing as well, so I modified my turtlebot2i\_world.launch to launch Rviz in it.

I met some strange error I didn't notice before in turtlebot2i\_world.launch, which says **robot\_semantic\_description not found**. I have never seen it before, maybe I didn't notice the existence of it at upper part after running `roslaunch turtlebot2i_gazebo turtlebot2i_world.launch`. But when I run it again, it disappears.

I don't know why I don't see these at the bottom of turtlebot2i\_world.launch window, I guess they appear previously after I resume Gazebo after launching. So I set the pause flag to "true" and do it again, and it turns out my inference is right. So now I change it back to "false", and I don't think it will cause problems at this point of time.

```

[turtlebot2i-simul... x turtlebot2i-simul... x turtlebot2i-simul... x turtlebot2i-simul... x +]
[INFO] [1583864161.305491, 0.000000]: Loading controller: arm_elbow_flex_position_controller
[INFO] [1583864161.362202, 0.000000]: Loading controller: arm_wrist_flex_position_controller
[INFO] [1583864161.428079, 0.000000]: Loading controller: joint_state_controller
[INFO] [1583864161.438248, 0.000000]: Controller Spawner: Loaded controllers: arm_shoulder_pan_position_controller, arm_shoulder_lift_position_controller, arm_elbow_flex_position_controller, arm_wrist_flex_position_controller, joint_state_controller
[urdf_spawner-4] process has finished cleanly
log file: /home/turtlebot2i-simulation/.ros/log/30596c10-62fb-11ea-8e7b-000c298a0000/urdf_spawner-4*.log
[INFO] [1583864169.363.710000]: Started controllers: arm_shoulder_pan_position_controller, arm_shoulder_lift_position_controller, arm_elbow_flex_position_controller, arm_wrist_flex_position_controller, joint_state_controller
[INFO] [1583864169.069494268, 363.732000000]: waitForService: Service [/gazebo/set_physics_properties] is now available.
[INFO] [1583864169.070485045, 363.732000000]: waitForService: Service [/gazebo/set_physics_properties] is now available.
[INFO] [1583864169.324267262, 363.802000000]: Physics dynamic reconfigure ready
[INFO] [1583864169.326130375, 363.803000000]: Physics dynamic reconfigure ready

```

I launched turtlebot2i\_world.launch solely, TF error still exists. I think they may not be well defined. I will leave it here at this point of time.

Now resume modifying turtlebot2i\_world.launch to make it launch rtabmap.launch

```

<!-- these are the arguments you can pass this launch file, for example paused:=true -->
<arg name="paused" default="false"/>
<arg name="use_sim_time" default="true"/>
<arg name="gui" default="true"/>
<arg name="headless" default="false"/>
<arg name="debug" default="false"/>
<arg name="new_rtabmap" default="true"/>
<arg name="localization" default="false"/>
<arg name="rviz" default="true"/>
<arg name="rtabmapviz" default="true"/>

```

```

<!-- Run a python script to send a service call to gazebo_ros to spawn a URDF robot -->
<node name="urdf_spawner" pkg="gazebo_ros" type="spawn_model" respawn="false" output="screen"
      args="--urdf -model turtlebot -param robot_description"/>

<!-- ros_control turtlebot launch file -->
<include file="$(find turtlebot2i_control)/launch/turtlebot2t_control.launch" />

<!-- Velocity muxer -->
<node pkg="nodelet" type="nodelet" name="mobile_base_nodelet_manager" args="manager"/>
<node pkg="nodelet" type="nodelet" name="cmd_vel_mux"
      args="Load yocs_cmd_vel_mux/CmdvelMuxNodelet mobile_base nodelet_manager">
    <param name="yaml_cfg_file" value="$(find turtlebot2i_bringup)/param/mux.yaml" />
    <remap from="cmd_vel_mux/output" to="mobile_base/commands/velocity"/>
</node>

<!-- Fake laser -->
<node pkg="depthimage_to_laserscan" type="depthimage_to_laserscan" name="depthimage_to_laserscan">
  <param name="scan_height" value="10"/>
  <param name="output_frame_id" value="/camera_depth_frame"/>
  <param name="range_min" value="0.45"/>
  <remap from="image" to="/camera/depth/image_raw"/>
  <remap from="scan" to="/scan"/>
</node>

<!-- launch rtabmap -->
<include file="$(find turtlebot2i_bringup)/launch/rtabmap.launch">
  <arg if="$(arg new_rtabmap)" name="args" value="--delete_db_on_start"/>
  <arg name="localization" value="$(arg localization)"/>
  <arg name="rtabmapviz" value="$(arg rtabmapviz)"/>
</include>

<!-- Rviz Configuration -->
<include if="true" file="$(find turtlebot2i_bringup)/launch/rviz.launch"/>

```

The XML tab is selected at the bottom.

Now run turtlebot2i\_world.launch again

TF error still exists, and Rviz will die at the beginning, but will recover later. I guess this is because of the speed of computation.

I found there is an area of blank in front of robot, which is the same shape as the arm. So I think that's because rtabmap is taking astra camera data, which is partially blocked by the arm, as input.

Now I want to remap that to sr300 camera data.

After modification:

```

<!-- inputs -->
<remap from="scan" to="/scan"/>
<!-- TODO: sr300: remap from="rgb/image" to="/camera/rgb/image_rect_color"/> -->
<remap from="rgb/image" to="camera_sr300/rgb/image_rect_color"/><!--
<remap if="$(eval $'3d_sensor') != 'sr300'" from="depth/image" to="/camera/
depth_registered/image_rect_color"/>
<remap if="$(eval $'3d_sensor') == 'sr300'" from="depth/image" to="/camera/depth/
image_rect_color"/>
<remap if="true" from="depth/image" to="camera_sr300/depth/image_rect_color"/>
<remap from="rgb/camera_info" to="camera_sr300/rgb/camera_info"/>

```

```

rtabmap.launch (~/turtlebot2i/src/turtlebot2i_bringup/launch) - gedit
rtabmap.launch x turtlebot2i_world.launch x turtlebot2i_mapping.launch x
Save
rtabmap.launch x turtlebot2i_world.launch x turtlebot2i_mapping.launch x

<!-- Odometry : ONLY for testing without the actual robot! /odom TF should not be already published. -->
<node if="$(arg rgbd_odometry)" pkg="rtabmap_ros" type="rgbd_odometry" name="rgbd_odometry" output="screen">
<param name="frame_id" type="string" value="base_footprint"/>
<param name="wait_for_transform_duration" type="double" value="$(arg wait_for_transform)"/>
<param name="RegForce3Dof" type="string" value="true"/>
<param name="Vls/InlierDistance" type="string" value="0.05"/>

<renap from="rgb/image" to="camera_sr300/rgb/image_rect_color"/>
<renap from="depth/image" to="camera_sr300/depth/image_raw"/>
<renap from="rgb/camera_info" to="camera_sr300/rgb/camera_info"/>

</node>

<!-- visualization with rtabmapviz -->
<node if="$(arg rtabmapviz)" pkg="rtabmap_ros" type="rtabmapviz" name="rtabmapviz" args="-d 0">
<find rtabmap_ros>/launch/config/rgbd_gui.lnf" output="screen">
<param name="subscribe_depth" type="bool" value="true"/>
<param name="subscribe_scan" type="bool" value="true"/>
<param name="frame_id" type="string" value="base_footprint"/>
<param name="wait_for_transform_duration" type="double" value="$(arg wait_for_transform)"/>

<renap from="rgb/image" to="camera_sr300/rgb/image_rect_color"/>
<!--
<renap if="$(eval $(3d_sensor)) != 'zr300'" from="depth/image" to="/camera/
depth_registered/image_raw"/>
<renap if="$(eval $(3d_sensor)) == 'zr300'" from="depth/image" to="/camera/depth/
image_raw"/> -->
<renap if="true" from="depth/image" to="camera_sr300/depth/image_raw"/>
<renap from="rgb/camera_info" to="camera_sr300/rgb/camera_info"/>
<renap from="scan" to="/scan"/>

</node>

</group>
</launch>

```

XML Tab Width: 8 ▾ Ln 81, Col 67 ▾ INS

As it always shows this warning, and I think it may be related to the following flag, so I set it to true.

```

[ INFO] [1583866884.906319246, 846.928000000]: rtabmap (440): Rate=1.00s, Limit=0.7005, RTAB-Map=0.0546s, Maps update=0.0027s pub=0.0000s (local map=1, WM=1)
[ INFO] [1583866886.768318732, 847.941000000]: rtabmap (441): Rate=1.00s, Limit=0.7005, RTAB-Map=0.0476s, Maps update=0.0032s pub=0.0000s (local map=1, WM=1)
[ INFO] [1583866888.812682556, 849.081000000]: rtabmap (442): Rate=1.00s, Limit=0.7005, RTAB-Map=0.0435s, Maps update=0.0041s pub=0.0000s (local map=1, WM=1)
[ WARN] [1583866889.066966820, 849.241000000]: /rtabmap/rtabmapviz: Did not receive data since 0.7005 seconds! Make sure the input topics are published ('$ rostopic hz topic') and the timestamps in their header are set. If topics are coming from different computers, make sure the clocks of the computers are synchronized ("ntpdate"). If topics are not published at the same rate, you could increase "queue_size" parameter (current=16).
/rtabmap/rtabmapviz subscribed to (approx sync):
/rtabmap/odom,
/camera/rgb/image_rect_color,
/camera/depth/image_raw,
/camera/rgb/camera_info,
/scan

```

```

rtabmap.launch (~/turtlebot2i/src/turtlebot2i_bringup/launch) - ge
rtabmap.launch x turtlebot2i_world.launch x
Open ▾
<?xml version="1.0"?>
<launch>
<arg name="database_path" default="rtabmap.db"/>
<arg name="localization" default="false"/>
<arg name="rgbd_odometry" default="false"/>
<arg name="args" default=""/>
<arg name="rtabmapviz" default="true"/>
<arg name="wait_for_transform" default="0.2"/>

```

```

rtabmap.launch (~/turtlebot2i/src/turtlebot2i_bringup/la
rtabmap.launch x turtlebot2i_world.launch x
Open ▾
<?xml version="1.0"?>
<launch>
<arg name="database_path" default="rtabmap.db"/>
<arg name="localization" default="false"/>
<arg name="rgbd_odometry" default="true"/>
<arg name="args" default=""/>
<arg name="rtabmapviz" default="true"/>
<arg name="wait_for_transform" default="0.2"/>

```

Now test it. It shows many errors like this, but things seem to work in the right way. I guess it is because of the speed of computation, and after failure, the program is looping to try again. The result shows the following 2 topics doesn't have message published to them

```
[ INFO] [1583867166.363984819, 377.900000000]: rtabmap (11): Rate=1.00s, Limit=0
.700s, RTAB-Map=0.0470s, Maps update=0.0014s pub=0.0000s (local map=1, WM=1)
[ INFO] [1583867166.363984819, 377.900000000]: rtabmap (11): rtabmap (11) sub=0.0000s (local map=1, WM=1)
[ INFO] [1583867166.363984819, 377.900000000]: rtabmap (11): Maps update=0.0027s pub=0.0000s (local map=1, WM=1)
[ WARN] [1583867169.023170438, 379.285000000]: /rtabmap/rtabmapviz: Did not receive data since 5 seconds! Make sure the input topics are published ("$ rostopic hz my_topic") and the timestamps in their header are set. If topics are coming from different computers, make sure the clocks of the computers are synchronized ("ntpdate"). If topics are not published at the same rate, you could increase "queue_size" parameter (current=10).
/rtabmap/rtabmapviz subscribed to (approx sync):
  /rtabmap/odom
  /camera_sr300/rgb/image_rect_color
  /camera_sr300/depth/image_raw,
  /camera_sr300/rgb/camera_info,
  /scan
```

I find something new

```
/home/turtlebot2i-simul... × turtlebot2i-simulation@... × turtlebot2i-simulation@... × + ✓
[ INFO] [1583867176.920493170, 383.220000000]: rtabmap (16): Rate=1.00s, Limit=0
.700s, RTAB-Map=0.0534s, Maps update=0.0020s pub=0.0000s (local map=1, WM=1)
[ INFO] [1583867178.975265859, 384.245000000]: rtabmap (17): Rate=1.00s, Limit=0
.700s, RTAB-Map=0.0536s, Maps update=0.0009s pub=0.0001s (local map=1, WM=1)
[ WARN] [1583867179.046547153, 384.285000000]: /rtabmap/rtabmapviz: Did not receive data since 5 seconds! Make sure the input topics are published ("$ rostopic hz my_topic") and the timestamps in their header are set. If topics are coming from different computers, make sure the clocks of the computers are synchronized ("ntpdate"). If topics are not published at the same rate, you could increase "queue_size" parameter (current=10).
/rtabmap/rtabmapviz subscribed to (approx sync):
  /rtabmap/odom
  /camera_sr300/rgb/image_rect_color,
  /camera_sr300/depth/image_raw,
  /camera_sr300/rgb/camera_info,
  /scan
[ WARN] [1583867179.744564045, 384.647000000]: /rtabmap/rbgd_odometry: Did not receive data since 5 seconds! Make sure the input topics are published ("$ rostopic hz my_topic") and the timestamps in their header are set.
/rtabmap/rbgd_odometry subscribed to (approx sync):
  /camera_sr300/rgb/image_rect_color,
  /camera_sr300/depth/image_raw,
  /camera_sr300/rgb/camera_info
[ INFO] [1583867181.225495021, 385.332000000]: rtabmap (18): Rate=1.00s, Limit=0
```

After checking rostopic info, I find rgbd\_odometry node subscribes to image\_rect\_color. So I think I should fix image\_rect\_color first.

```
turtlebot2i-simulation@ubuntu:~/turtlebot2i/src/turtlebot2i_gazebo/launch$ rostopic info /camera_sr300/rgb/image_rect_color
Type: sensor_msgs/Image
Publishers: None
Subscribers:
* /rtabmap/rtabmapviz (http://ubuntu:42295/)
* /rtabmap/rbgd_odometry (http://ubuntu:35409/)
```

After checking how the image\_rect\_color topic is published, I find it seems to be a default name of this package

```
turtlebot2i-simulation@ubuntu:~/turtlebot2i/src/turtlebot2i_gazebo/launch$ grep -r '/camera/rgb/image_rect_color' /opt/ros/kinetic/share/
/opt/ros/kinetic/share/realsense_camera/rviz/realSenseRvizConfiguration3.rviz:
  Image Topic: /camera/rgb/image_rect_color
  Image Topic: /camera/rgb/image_rect_color
  /opt/ros/kinetic/share/realsense_camera/rviz/realSenseRvizConfiguration2.rviz:
    Image Topic: /camera/rgb/image_rect_color
    Image Topic: /camera/rgb/image_rect_color
    /opt/ros/kinetic/share/rtabmap_ros/launch/rtabMapping.launch: <arg name="rgb_topic"
      default="/camera/rgb/image_rect_color" />
    /opt/ros/kinetic/share/rtabmap_ros/launch/rtabMapping.launch: <arg name="rgb_topic"
      default="/camera/rgb/image_rect_color" />
    /opt/ros/kinetic/share/rtabmap_ros/launch/config/test_odometry.rviz: Image
Topic: /camera/rgb/image_rect_color
    /opt/ros/kinetic/share/rtabmap_ros/launch/config/rgbd.rviz: Image Topic: /camera/rgb/image_rect_color
    /opt/ros/kinetic/share/rtabmap_ros/launch/config/rgbdSlam_datasets.rviz: Image
Topic: /camera/rgb/image_rect_color
    /opt/ros/kinetic/share/rtabmap_ros/launch/demo/demodata_recorder.launch: <node
      name="republish_rgbd" type="republish" pkg="image_transport" args="compressed
      in:=data_throttled_image raw_out:=/camera/rgb/image_rect_color" />
    /opt/ros/kinetic/share/rtabmap_ros/launch/demo/turtlebot_mapping.launch: <arg
unless="$arg simulation" name="rgb_topic" default="/camera/rgb/image_rect_color"/>
    /opt/ros/kinetic/share/turtlebot_rviz_launchers/rviz/robot.rviz: Color Image
Topic: /camera/rgb/image_rect_color
```

I also find the topic with new camera name has no publisher at all

```
turtlebot2i-simulation@ubuntu:~/turtlebot2i/src/turtlebot2i_gazebo/launch$ rostopic info /camera_sr300/rgb/image_rect_color
Type: sensor_msgs/Image
Publishers: None
Subscribers:
* /rtabmap/rtabmapviz (http://ubuntu:42295/)
* /rtabmap/rbgd_odometry (http://ubuntu:35409/)
```

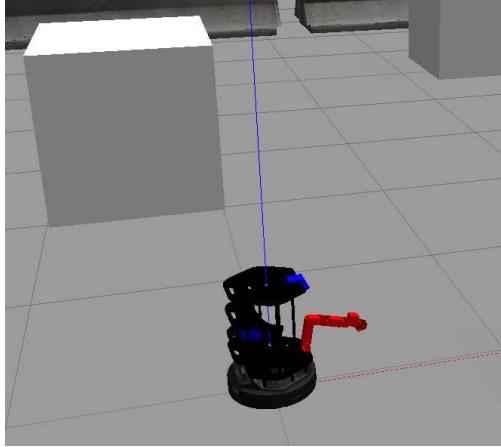
But what is weird is camera/rgb/image\_rect\_color is not published either.

I decided to change some camera\_sr300/rgb/image\_rect\_color back to camera/rgb/image\_rect\_color in /turtlerebot2i Bringup/rtabmap.launch

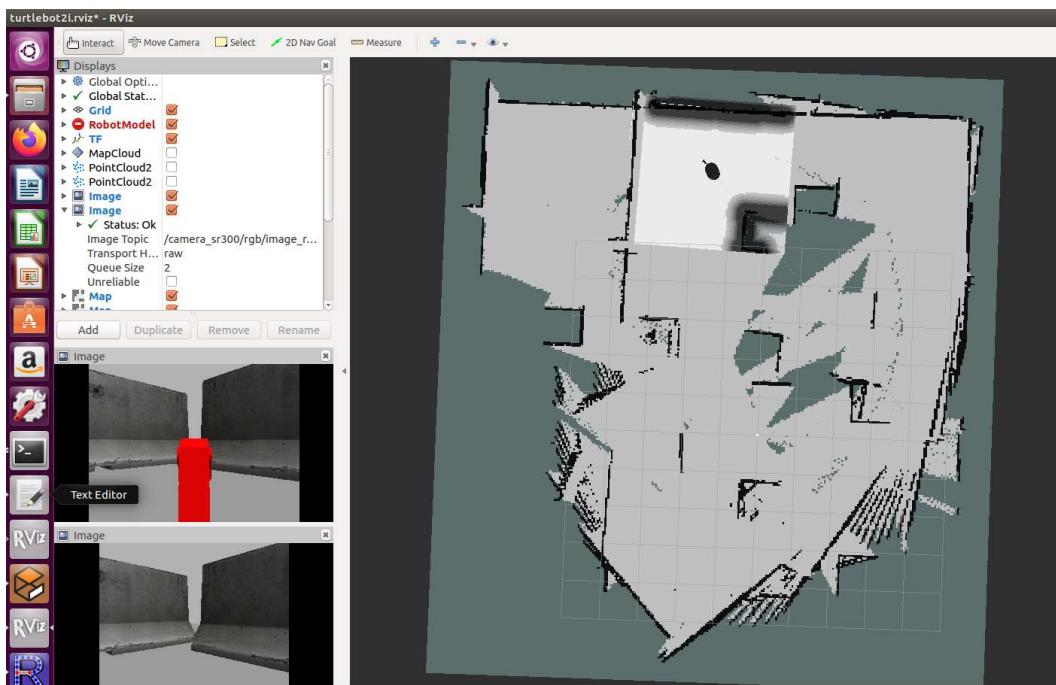
```
<!-- Odometry : ONLY for testing without the actual robot! /odom TF should not be already  
published. -->  
<node if="$(arg rgbd_odometry)" pkg="rtabmap_ros" type="rgbd_odometry" name="rgbd_odometry"  
output="screen">  
  <param name="frame_id" type="string" value="base_footprint"/>  
  <param name="wait_for_transform_duration" type="double" value="$(arg wait_for_transform)"/>  
  <param name="Reg/Force3DoF" type="string" value="true"/>  
  <param name="VLs/InlierDistance" type="string" value="0.05"/>  
  
  <remap from="rgb/image" to="/camera/rgb/image_rect_color"/>  
  <remap from="depth/image" to="/camera_sr300/depth/image_raw"/>  
  <remap from="rgb/camera_linfo" to="/camera_sr300/rgb/camera_linfo"/>  
</node>  
  
<!-- visualization with rtabmapviz -->  
<node if="$(arg rtabmapviz)" pkg="rtabmap_ros" type="rtabmapviz" name="rtabmapviz" args="-d  
$(find rtabmap_ros)/launch/config/rgbd_gui.ini" output="screen">  
  <param name="subscribe_depth" type="bool" value="true"/>  
  <param name="subscribe_scan" type="bool" value="true"/>  
  <param name="frame_id" type="string" value="base_footprint"/>  
  <param name="wait_for_transform_duration" type="double" value="$(arg wait_for_transform)"/>  
  
  <remap from="rgb/image" to="/camera/rgb/image_rect_color"/>
```

I don't know whether it will work, maybe the "camera" is only inherited from default="camera"?  
Anyway, try current modification.

Try to make a map, and I find the arm moves a little. I infer this may be because the controller is not ready as soon as Gazebo is ready, namely computation speed limitation, so it moves a little during that period of time.



Jumping issue again, but this time the parameter set is much more stable than that of Turtlebot. But this jumping is kind of reasonable, as when I turn the robot around, the obstacles matches the previously mapped obstacles

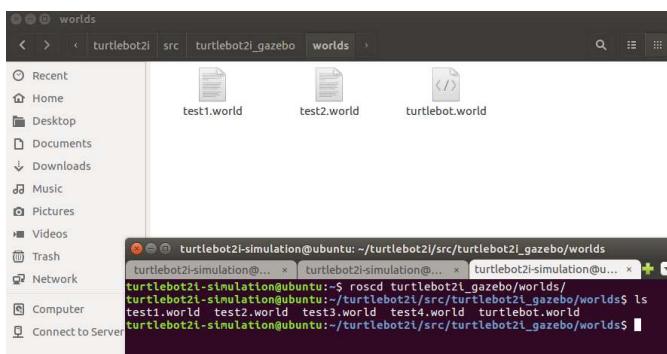
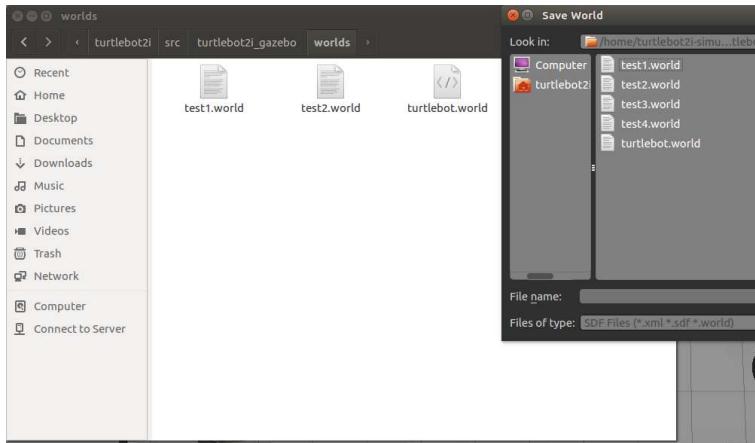


Try again, jump again. I think this is because the environment I created is quite symmetric, so I will modify it a little bit. Maybe use the café model?

For the wheel, when adding café in Gazebo, the robot is lifted while the wheels stay on the same altitude. I launch `turtlebot2i_world.launch` and `turtlebot2i_mapping.launch`. The TF tree is complete. Amazing, maybe the gripper and wheel definitions are only called in `turtlebot2i_mapping.launch`??

Created a new Gazebo world with less similar features

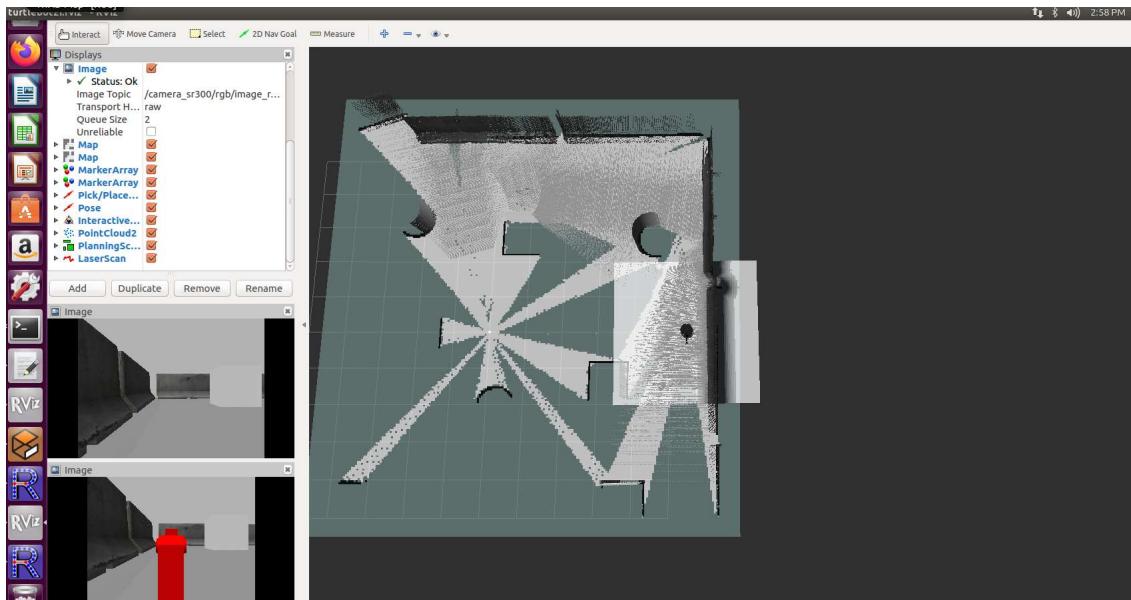
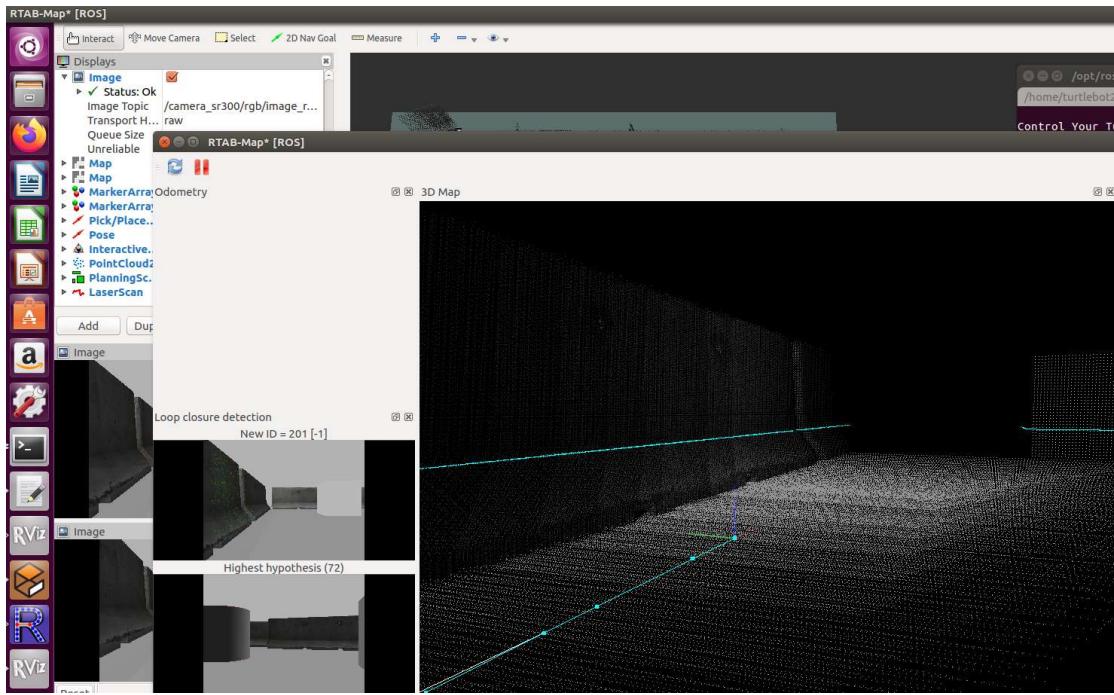
Interesting, I created a new world and use terminal to double check it. It's there. But when I call it, it's not found. I open GUI to find it, and it's not there. Now again.



Reboot, and it gets normal. World files appear.

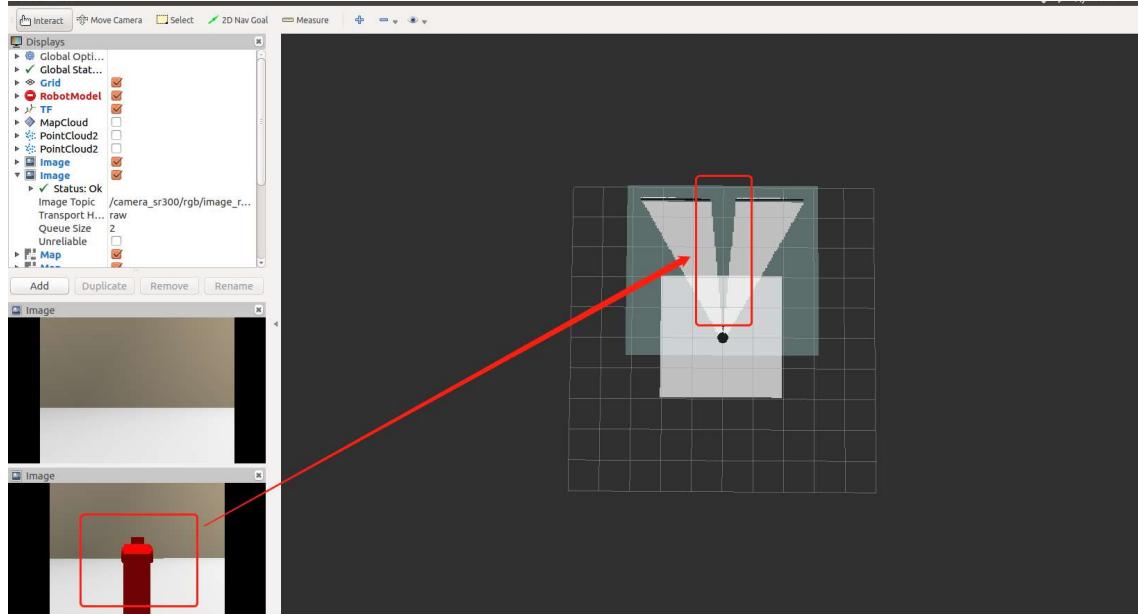
I don't like the slight move of arm, so I set the pause flag to "true", and will resume Gazebo manually.

I just find things work kind of so nice and the map in Rviz seems to be a 3D one. Well no, it turns out the 3D illusion in Rviz is just because I failed to disable the display of map cloud.



After a while, it jumps again, and the surrounding obstacles still matches previously mapped obstacles well. Now I use the café, but sink it a little to make the robot not lifted by the café floor.

The laserscan data is kind of weird. I infer that because `depthimage_to_laserscan` is not taking the demanded input image.



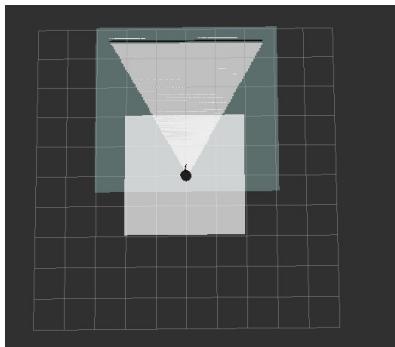
After checking, I changed these to `camera_sr300` related topics

```
turtlebot2i_world.launch ~/turtlebot2i/src/turtlebot2i_gazebo/launch - gedit
Open ▾ Save
<param name="robot_description" command="$(catkin_find urdf_file)" />
<!-- Run a python script to send a service call to gazebo_ros to spawn a URDF robot -->
<node pkg="gazebo_spawner" type="gazebo_ros" name="spawn_model" respawn="false" output="screen"
      args="--urdf -model turtlebot -param robot_description"/>
<!-- ros_control turtlebot launch file -->
<include file="$(find turtlebot2i)/Launch/turtlebot2i_control.launch" />
<!-- Velocity muxer -->
<node pkg="nodelet" type="nodelet" name="mobile_base_nodelet_manager" args="manager"/>
<node pkg="nodelet" type="nodelet" name="cmd_vel_mux"
      args="Load yocs_cmd_vel_mux/CmdvelMuxNodelet mobile_base_nodelet_manager">
    <param name="yaml_cfg_file" value="$(find turtlebot2i_bringup)/param/mux.yaml" />
    <remap from="cmd_vel_mux/output" to="mobile_base/commands/velocity"/>
</node>
<!-- Fake laser -->
<node pkg="depthimage_to_laserscan" type="depthimage_to_laserscan" name="depthimage_to_laserscan">
  <param name="scan_height" value="10"/>
  <param name="output_frame_id" value="/camera_depth_frame"/>
  <param name="range_min" value="0.45"/>
  <remap from="image" to="/camera/depth/image_raw"/>
  <remap from="scan" to="/scan"/>
</node>
```

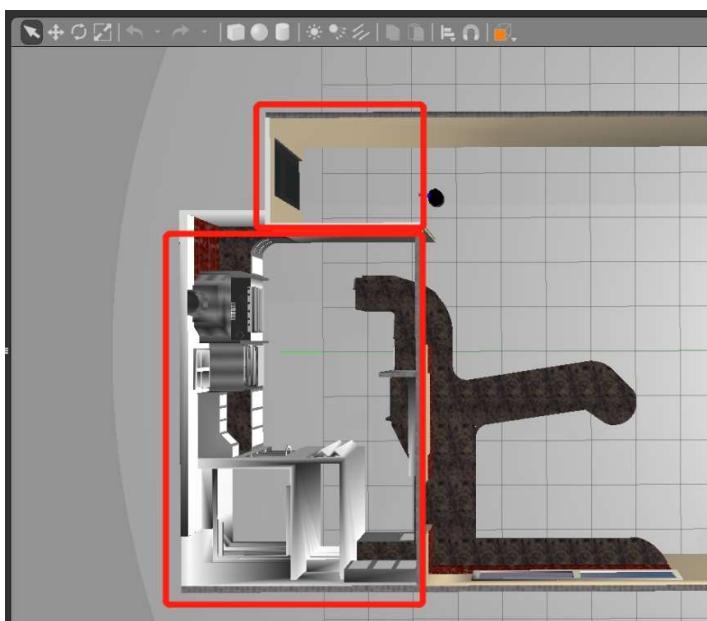
I find another parameter `scan_height`, as it can be defined arbitrarily, I think it should be defined according to the configuration of robot

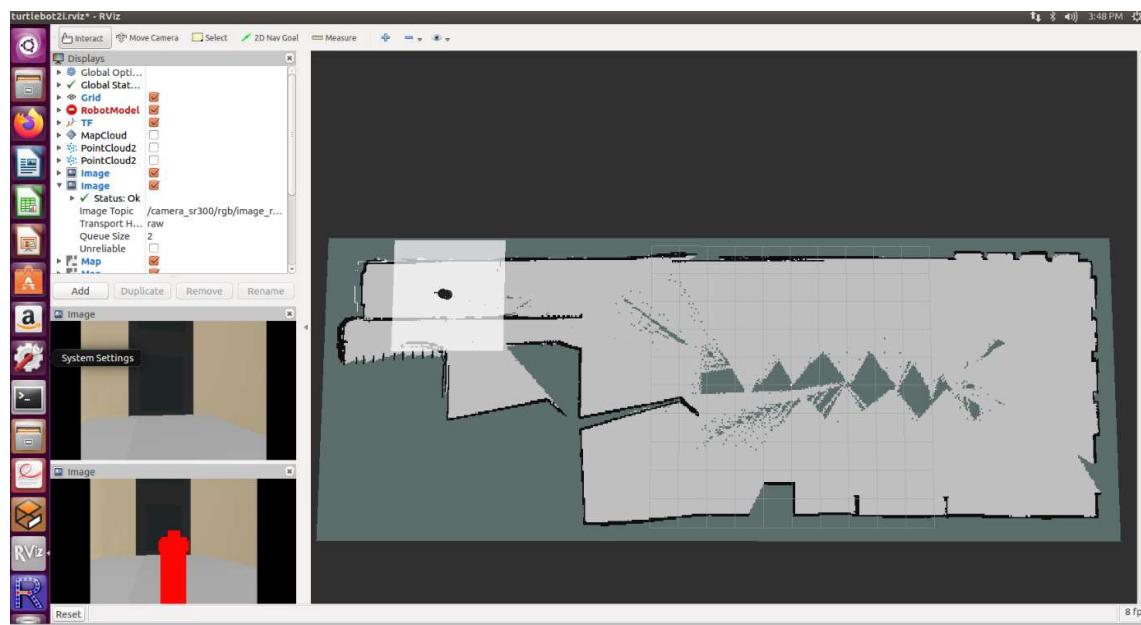
```
<!-- Fake laser -->
<node pkg="depthimage_to_laserscan" type="depthimage_to_laserscan" name="depthimage_to_laserscan">
  <param name="scan_height" value="10"/>
  <param name="output_frame_id" value="/camera_sr300_depth_frame"/>
  <param name="range_min" value="0.45"/>
  <remap from="image" to="/camera_sr300/depth/image_raw"/>
  <remap from="scan" to="/scan"/>
</node>
```

Nice, things are working in the right way



Things work almost in the right way. What is interesting is that these 2 parts look accessible but actually not.



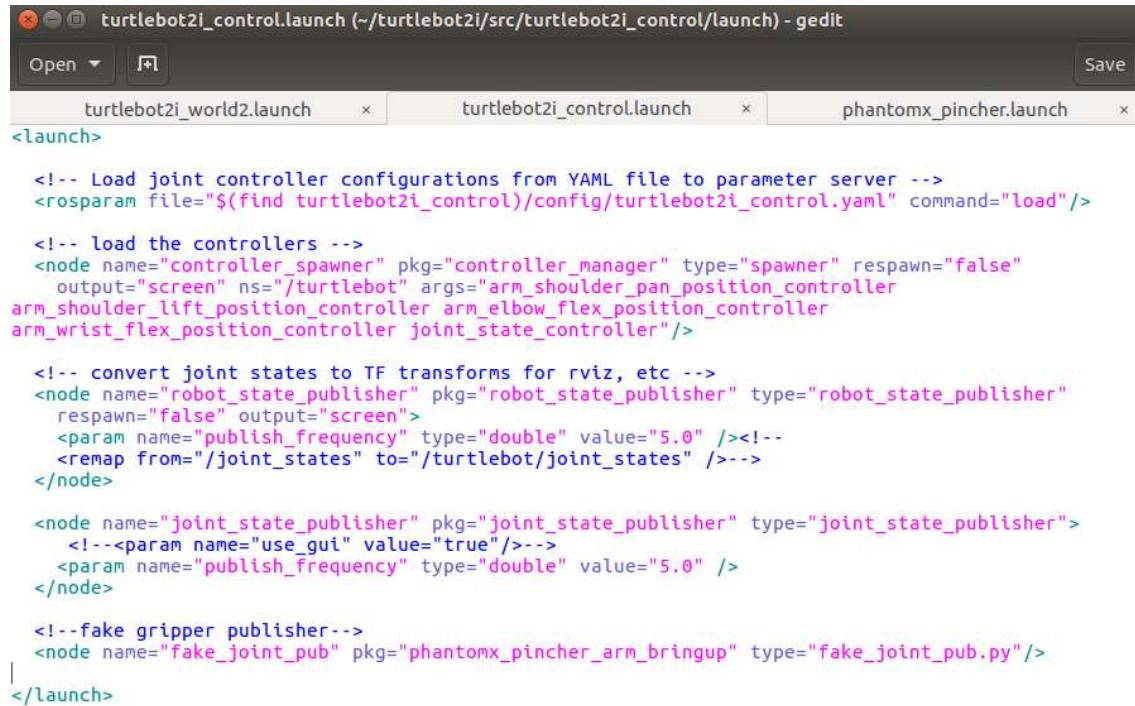


Okay, make a git copy.

04/03/2020 update

The problem now is transform of movable links, i.e. wheel links and arm links, are published to different topics, i.e. /joint\_states and /turtlebot/joint\_states.

I thought that joint\_state\_publiser will publish all movable joint information to /joint\_states. But it was wrong, it is just a GUI to manually change moveable joint angles and publish transformation to /tf. The fake gripper publisher published a joint of gripper, which helps to complete transform tree as well. **But transform of all links is complete.**



```
<!-- Load joint controller configurations from YAML file to parameter server -->
<rosparam file="$(find turtlebot2i_control)/config/turtlebot2i_control.yaml" command="load"/>

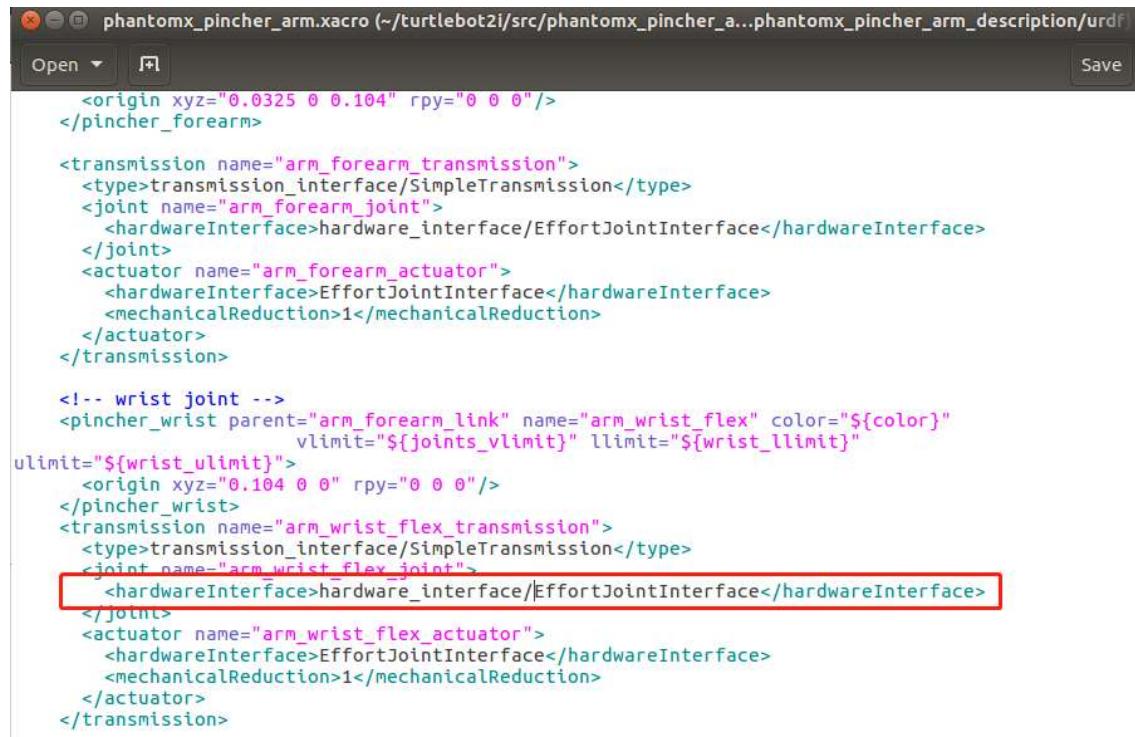
<!-- load the controllers -->
<node name="controller_spawner" pkg="controller_manager" type="spawner" respawn="false"
      output="screen" ns="/turtlebot" args="arm_shoulder_pan_position_controller
arm_shoulder_lift_position_controller arm_elbow_flex_position_controller
arm_wrist_flex_position_controller joint_state_controller"/>

<!-- convert joint states to TF transforms for rviz, etc -->
<node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher"
      respawn="false" output="screen">
    <param name="publish_frequency" type="double" value="5.0" /><!--
    <remap from="/joint_states" to="/turtlebot/joint_states" /-->
</node>

<node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher">
    <!--<param name="use_gui" value="true"/><!--
    &lt;param name="publish_frequency" type="double" value="5.0" /&gt;
&lt;/node&gt;

&lt;!--fake gripper publisher--&gt;
&lt;node name="fake_joint_pub" pkg="phantomx_pincher_arm_bringup" type="fake_joint_pub.py"/&gt;
|&lt;/launch&gt;</pre>
```

Prepended "hardware\_interface/" for 4 joints



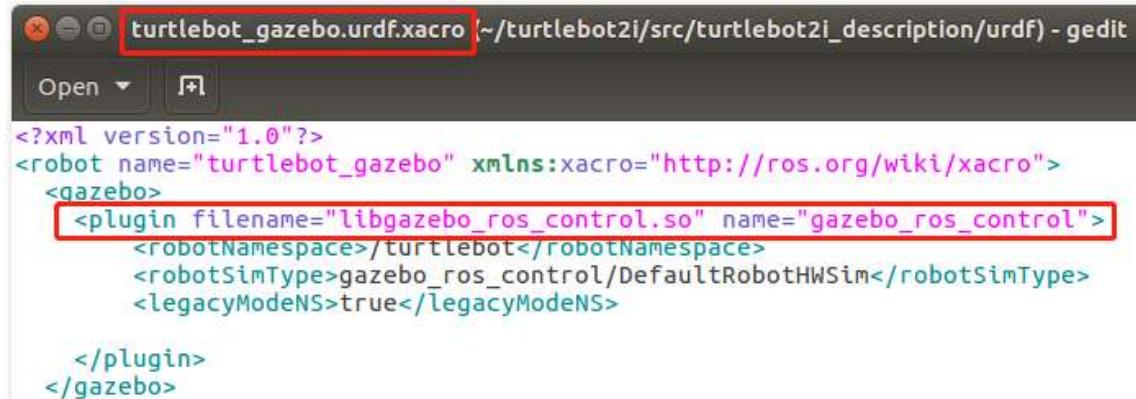
```
<origin xyz="0.0325 0 0.104" rpy="0 0 0"/>
</pincher_forearm>

<transmission name="arm_forearm_transmission">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="arm_forearm_joint">
    <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
  </joint>
  <actuator name="arm_forearm_actuator">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>

<!-- wrist joint -->
<pincher_wrist parent="arm_forearm_link" name="arm_wrist_flex" color="${color}">
  vlimit="${joints_vlimit}" llimit="${wrist_llimit}"
  ulimit="${wrist_ulimit}">
  <origin xyz="0.104 0 0" rpy="0 0 0"/>
</pincher_wrist>
<transmission name="arm_wrist_flex_transmission">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="arm_wrist_flex_joint">
    <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
  </joint>
  <actuator name="arm_wrist_flex_actuator">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>
```

The file hierarchy is kind of massed up originally, this shouldn't be invoked in a camera description file. More specifically, this gazebo\_control plugin.

```
turtlebot2i-simulation@ubuntu:~/turtlebot2i/src/turtlebot2i_description/urdf/arm
s$ grep -r "turtlebot_gazebo.urdf.xacro" "/home/turtlebot2i-simulation/turtlebot
2i"
/home/turtlebot2i-simulation/turtlebot2i/src/turtlebot2i_description/urdf/sensor
s/sr300.urdf.xacro: <xacro:include filename="$(find turtlebot2i_description)/ur
df/turtlebot_gazebo.urdf.xacro"/>
/home/turtlebot2i-simulation/turtlebot2i/src/turtlebot2i_description/urdf/sensor
s/astra.urdf.xacro: <xacro:include filename="$(find turtlebot2i_description)/ur
df/turtlebot_gazebo.urdf.xacro"/>
/home/turtlebot2i-simulation/turtlebot2i/src/turtlebot2i_description/urdf/sensor
s/zr300.urdf.xacro: <xacro:include filename="$(find turtlebot2i_description)/ur
df/turtlebot_gazebo.urdf.xacro"/>
Binary file /home/turtlebot2i-simulation/turtlebot2i/src/.git/index matches
turtlebot2i-simulation@ubuntu:~/turtlebot2i/src/turtlebot2i_description/urdf/arm
```



```
<?xml version="1.0"?>
<robot name="turtlebot_gazebo" xmlns:xacro="http://ros.org/wiki/xacro">
  <gazebo>
    <plugin filename="libgazebo_ros_control.so" name="gazebo_ros_control">
      <robotNamespace>/turtlebot</robotNamespace>
      <robotSimType>gazebo_ros_control/DefaultRobotHWSim</robotSimType>
      <legacyModeNS>true</legacyModeNS>
    </plugin>
  </gazebo>
```

By the way, <robotNamespace>/turtlebot</robotNamespace> above determines the prefix of topics related to controller



```
/turtlebot/arm_elbow_flex_position_controller/command
/turtlebot/arm_elbow_flex_position_controller/pid/parameter_descriptions
/turtlebot/arm_elbow_flex_position_controller/pid/parameter_updates
/turtlebot/arm_elbow_flex_position_controller/state
/turtlebot/arm_shoulder_lift_position_controller/command
/turtlebot/arm_shoulder_lift_position_controller/pid/parameter_descriptions
/turtlebot/arm_shoulder_lift_position_controller/pid/parameter_updates
/turtlebot/arm_shoulder_lift_position_controller/state
/turtlebot/arm_shoulder_pan_position_controller/command
/turtlebot/arm_shoulder_pan_position_controller/pid/parameter_descriptions
/turtlebot/arm_shoulder_pan_position_controller/pid/parameter_updates
/turtlebot/arm_shoulder_pan_position_controller/state
/turtlebot/arm_wrist_flex_position_controller/command
/turtlebot/arm_wrist_flex_position_controller/pid/parameter_descriptions
/turtlebot/arm_wrist_flex_position_controller/pid/parameter_updates
/turtlebot/arm_wrist_flex_position_controller/state
/turtlebot/joint_states
```

```
rostopic pub /turtlebot/gripper1_position_controller/command std_msgs/Float64 -- 0.02
```

## Summary of modification:

You can choose to add publish\_frequency or not, or change its value.



```

<launch>

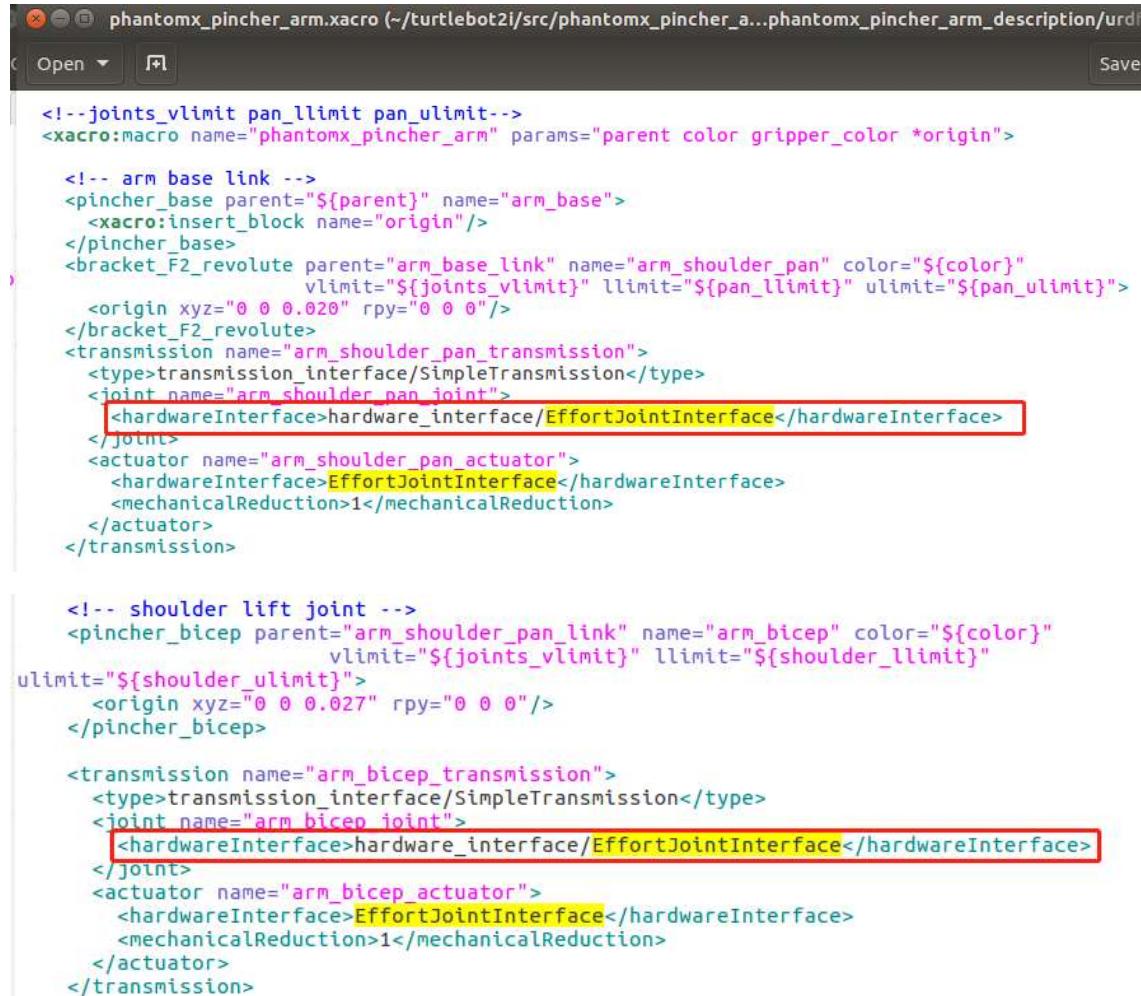
    <!-- Load joint controller configurations from YAML file to parameter server -->
    <rosparam file="$(find turtlebot2i_control)/config/turtlebot2i_control.yaml" command="load"/>

    <!-- load the controllers -->
    <node name="controller_spawner" pkg="controller_manager" type="spawner" respawn="false"
        output="screen" ns="/turtlebot" args="arm_shoulder_pan_position_controller
        arm_shoulder_lift_position_controller arm_elbow_flex_position_controller
        arm_wrist_flex_position_controller joint_state_controller gripper1_position_controller
        gripper2_position_controller"/>

    <!-- convert joint states to TF transforms for rviz, etc -->
    <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher"
        respawn="false" output="screen">
        <param name="publish_frequency" type="double" value="5.0" />
        <remap from="/joint_states" to="/turtlebot/joint_states" />
    </node>

```

Change <hardwareInterface> tag of 4 arm joints to “hardware\_interface/EffortJointInterface”



```

<!--joints_vlimit pan_llimit pan_ulimit-->
<xacro:macro name="phantomx_pincher_arm" params="parent color gripper_color *origin">

    <!-- arm base link -->
    <pincher_base parent="${parent}" name="arm_base">
        <xacro:insert_block name="origin"/>
    </pincher_base>
    <bracket_F2_revolute parent="arm_base_link" name="arm_shoulder_pan" color="${color}"
        vlimit="${joints_vlimit}" llimit="${pan_llimit}" ulimit="${pan_ulimit}">
        <origin xyz="0 0 0.020" rpy="0 0 0"/>
    </bracket_F2_revolute>
    <transmission name="arm_shoulder_pan_transmission">
        <type>transmission_interface/SimpleTransmission</type>
        <joint name="arm_shoulder_pan_joint">
            <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
        </joint>
        <actuator name="arm_shoulder_pan_actuator">
            <hardwareInterface>EffortJointInterface</hardwareInterface>
            <mechanicalReduction>1</mechanicalReduction>
        </actuator>
    </transmission>

    <!-- shoulder lift joint -->
    <pincher_bicep parent="arm_shoulder_pan_link" name="arm_bicep" color="${color}"
        vlimit="${joints_vlimit}" llimit="${shoulder_llimit}"
        ulimit="${shoulder_ulimit}">
        <origin xyz="0 0 0.027" rpy="0 0 0"/>
    </pincher_bicep>

    <transmission name="arm_bicep_transmission">
        <type>transmission_interface/SimpleTransmission</type>
        <joint name="arm_bicep_joint">
            <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
        </joint>
        <actuator name="arm_bicep_actuator">
            <hardwareInterface>EffortJointInterface</hardwareInterface>
            <mechanicalReduction>1</mechanicalReduction>
        </actuator>
    </transmission>

```

```
<!-- elbow joint -->
<pincher_forearm parent="arm_bicep_link" name="arm_forearm" color="${color}"
    vlimit="${joints_vlimit}" llimit="${elbow_llimit}"
    ulimit="${elbow_ulimit}">
    <origin xyz="0.0325 0 0.104" rpy="0 0 0"/>
</pincher_forearm>

<transmission name="arm_forearm_transmission">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="arm_forearm_joint">
        <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
    </joint>
    <actuator name="arm_forearm_actuator">
        <hardwareInterface>EffortJointInterface</hardwareInterface>
        <mechanicalReduction>1</mechanicalReduction>
    </actuator>
</transmission>

<!-- wrist joint -->
<pincher_wrist parent="arm_forearm_link" name="arm_wrist_flex" color="${color}"
    vlimit="${joints_vlimit}" llimit="${wrist_llimit}"
    ulimit="${wrist_ulimit}">
    <origin xyz="0.104 0 0" rpy="0 0 0"/>
</pincher_wrist>
<transmission name="arm_wrist_flex_transmission">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="arm_wrist_flex_joint">
        <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
    </joint>
    <actuator name="arm_wrist_flex_actuator">
        <hardwareInterface>EffortJointInterface</hardwareInterface>
        <mechanicalReduction>1</mechanicalReduction>
    </actuator>
</transmission>
```

Comment all the contents at the end of phantomx\_pincher\_arm.xacro, apart from those in red rectangles. It seems to me that this link is useless and I met problem when using the gazebo part below that.



```

phantomx_pincher_arm.xacro (~/turtlebot2i/src/phantomx_pincher_a...phantomx_pincher_arm_desc
Open ▾ + ↻
</transmission>

<!-- gripper -->
<xacro:include filename="$(find phantomx_pincher_arm_description)/urdf/
phantomx_pincher_gripper.xacro"/>

<!-- gripper_link -->
<!--link name="gripper_link"/>
<joint name="gripper_link_joint" type="revolute">
  <origin xyz="0.110 0 0" rpy="0 0 0"/>
  <parent link="arm_wrist_flex_link"/>
  <child link="gripper_link"/>
  <limit effort="6" velocity="1" lower="-3.14" upper="3.14"/>
  <axis xyz="1 0 0"/>
</joint>

<transmission name="gripper_link_transmission">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="gripper_link_joint">
    <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
  </joint>
  <actuator name="gripper_link_actuator">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>

<gazebo reference="gripper_link">
  <material>Gazebo/${color}</material>
  <gravity>true</gravity>
</gazebo-->
<!--gazebo-->
<gripper name="gripper">
  <grasp_check>
    <attach_steps>5</attach_steps>
    <detach_steps>10</detach_steps>
    <min_contact_count>2</min_contact_count>
  </grasp_check>
</gripper>
<gripper_link>gripper_active_link</gripper_link>
<palm_link>gripper_servo_link</palm_link>
</gazebo-->

</xacro:macro>
</robot>

```

XML ▾ Tab Width: 8 ▾ Ln 70, Col 40

In phantomx\_pincher\_gripper.xacro, do the following modification.

```
<?xml version="1.0"?>
<!-- PhantomX Pincher gripper URDF-->
<robot xmlns:xacro="http://ros.org/wiki/xacro">

  <!-- gripper joint -->
  <!-- Finger 1 -->
  <joint name="gripper_joint" type="prismatic">
    <origin xyz="0.070 0 0" rpy="0 0 0"/>
    <axis xyz="0 1 0"/>
    <limit effort="3" velocity="0.05" lower="0.002" upper="0.032"/>
    <parent link="arm_wrist_flex_link"/>
    <child link="gripper_active_link"/>
    <dynamics friction="0.13"/>
  </joint>

  <pincher_finger name="gripper_active" color="${color}" >
    <origin xyz="0 0 0" rpy="0 0 0"/>
  </pincher_finger>

  <transmission name="gripper_joint_transmission">
    <type>transmission_interface/SimpleTransmission</type>
    <joint name="gripper_joint">
      <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
    </joint>
    <actuator name="gripper_joint_actuator">
      <hardwareInterface>EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
    </actuator>
  </transmission>

```

I changed the name of joint, because I think it would be more explicit.

effort is decreased to avoid large repulsion when grippers are in conflict with objects. The original effort value 30 will make the robot to fall down in some cases.

velocity is decreased because the original velocity is too large that the robot will be twisted influenced by the movement of grippers.

A friction parameter is added to avoid fluctuation of grippers after moving them.

Transmission is essential for publishing of gripper link in /tf and its actuation.

```

<!-- Finger 2 -->
<!-- Note: currently static but should be a Mimic of Finger 1 -->
<joint name="gripper2_joint" type="fixed">
  <origin xyz="0.070 -0.016 0" rpy="0 0 0"/>
  <parent link="arm_wrist_flex_link"/>
  <child link="gripper_active2_link"/>
</joint>

<pincher_finger name="gripper_active2" color="${color}" >
  <origin xyz="0 0 0" rpy="${M_PI} 0 0"/>
</pincher_finger>

<!-- Using Mimic -->
<!-- gripper 2 joint -->
<!-- TODO: Test Mimic -->
<joint name="gripper2_joint" type="prismatic">
  <origin xyz="0.070 0 0" rpy="0 0 0"/>
  <axis xyz="0 1 0"/>
  <limit effort="3" velocity="0.05" lower="-0.032" upper="-0.002"/>
  <parent link="arm_wrist_flex_link" />
  <child link="gripper_active2_link" />
  <!--mimic joint="gripper joint" multiplier="-1" offset = "0"-->
  <dynamics friction="0.13" />
</joint>

<transmission name="gripper2_joint_transmission">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="gripper2_joint">
    <hardwareInterface>hardware_interface/EffortJointInterface</hardwareInterface>
  </joint>
  <actuator name="gripper2_joint_actuator">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>

</robot>

```

The lower and upper parameter is modified because they should be identical to those of gripper1\_joint

The mimic will be valid when you use joint\_state\_publisher\_gui or visualize your URDF file in Rviz using

```
roslaunch urdf_tutorial display.launch model:=<your urdf file>
```

reference:

<https://answers.ros.org/question/10401/how-to-convert-xacro-file-to-urdf-file/>

<http://wiki.ros.org/urdf/Tutorials/Building%20a%20Visual%20Robot%20Model%20with%20URDF%20from%20Scratch>

But mimic tag doesn't work for Gazebo, so I abandoned it. After googling for a while, it seems to me that Gazebo doesn't have official mimic functionality. So I decided to use 2 controllers and 2 prismatic joints to actuate the 2 grippers, which is different from what's happening on real robot that grippers are actuated by the same servo.

Now modify the controller:

```
<!-- Load joint controller configurations from YAML file to parameter server -->
<rosparam file="$(find turtlebot2i_control)/config/turtlebot2i_control.yaml" command="load"/>

<!-- load the controllers -->
<node name="controller_spawner" pkg="controller_manager" type="spawner" respawn="false"
      output="screen" ns="/turtlebot" args="arm_shoulder_pan_position_controller
      arm_shoulder_lift_position_controller arm_elbow_flex_position_controller
      arm_wrist_flex_position_controller joint_state_controller gripper1_position_controller
      gripper2_position_controller"/>
```

Append 2 more controllers for 2 grippers respectively. Here is how the controllers are defined:  
Make sure names of controllers and joints are consistent as those you defined elsewhere

```
turtlebot:
  # Publish all joint states -----
  joint_state_controller:
    type: joint_state_controller/JointStateController
    publish_rate: 50

  # Position Controllers -----
  arm_shoulder_pan_position_controller:
    type: effort_controllers/JointPositionController
    joint: arm_shoulder_pan_joint
    pid: {p: 32.0, i: 0.01, d: 0.0}
  arm_shoulder_lift_position_controller:
    type: effort_controllers/JointPositionController
    joint: arm_bicep_joint
    pid: {p: 32.0, i: 0.01, d: 0.0}
  arm_elbow_flex_position_controller:
    type: effort_controllers/JointPositionController
    joint: arm_forearm_joint
    pid: {p: 32.0, i: 0.01, d: 0.0}
  arm_wrist_flex_position_controller:
    type: effort_controllers/JointPositionController
    joint: arm_wrist_flex_joint
    pid: {p: 32.0, i: 0.01, d: 0.0}
  gripper1_position_controller: ←
    type: effort_controllers/JointPositionController
    joint: gripper_joint ←
    pid: {p: 32.0, i: 0.01, d: 0.0}
  gripper2_position_controller: ←
    type: effort_controllers/JointPositionController
    joint: gripper2_joint ←
    pid: {p: 32.0, i: 0.01, d: 0.0}
```

I uploaded a new launch file “turtlebot2i\_world2.launch”, which will start Gazebo while won’t start rtabmap and Rviz.

After running “turtlebot2i\_world2.launch” you will be able to control the arm using command similar to the following in terminal:

```
rostopic pub /turtlebot/gripper2_position_controller/command std_msgs/Float64 -- -0.02
```

To run Rviz:

```
roslaunch turtlebot2i_bringup rviz.launch
```

To build a new map using rtabmap:

```
roslaunch turtlebot2i_bringup rtabmap.launch args:=--delete_db_on_start
```

**I would strongly recommend you look through the launch files to figure out what arguments you should pass in the command.**

The remaining problems now are:

1. Gripper doesn’t work accurately, and the process of grasping, i.e. when in collision with other objects, it still doesn’t work like that the real world. I think it can be optimized by changing the gripper speed, using optimal PID parameters and increasing the friction.
2. It seems to me that the gripper doesn’t have force feedback in gazebo. So when it grasps things, it is likely to be too loose(object falls) or too tight(the robot will be shaking...).
3. New problems are likely to show up during the process of developing tasks.
4. The transform of wheels are still missing. It can be fixed if we can remap message in /joint\_state to /turtlebot\_joint\_state, but I haven’t found how to do that. A fundamental way is to write a subscriber of /joint\_state and publish the message to /turtlebot\_joint state. Anyway, the absence of these 2 links is not a big problem, as far as I can see, it won’t affect the simulation.

Future Tasks:

1. Do path planning or even path planning through way points.
2. rtabmap seems to be an interesting package, and I will learn more about that.
3. Grasping of objects using logical camera.

I modified rtabmap.launch a little bit, basically only use information of sr300 camera, as astra is partially blocked by the arm. But something weird happened. I don’t know why things are turning red:

