Contents

1 Useful Resources **2**

2 Useful Tools **2**

3 Installation **3**

4 Package Overview **3**
- 4.1 world . . . 3
- 4.2 urdf . . . 3
  - 4.2.1 Intel Realsense D435 camera . . . 4
  - 4.2.2 cub_full_robosense.urdf.xacro . . . 4
  - 4.2.3 cub_full_urdf.xacro . . . 4
  - 4.2.4 cub_full_shifted_urdf.xacro . . . 4
  - 4.2.5 cub_full_issac_urdf.xacro . . . 4
- 4.3 meshes . . . 4
- 4.4 rviz . . . 4
- 4.5 launch . . . 5

5 Mesh Manipulation **5**
- 5.1 Reducing the quality of a mesh (Mesh Lab) . . . 5
- 5.2 Scaling Down a mesh (Fusion 360) . . . 5
- 5.3 Mesh Origin . . . 6
  - 5.3.1 Centering a mesh . . . 6
  - 5.3.2 Translating & rotating a mesh . . . 6

6 Computing Intertia of meshes **6**
- 6.1 Key Variables . . . 6
- 6.2 Computing Intertia . . . 7
- 6.3 Importing Inertia Values into URDF . . . 8
- 6.4 Verifying Inertia Values . . . 9
- 6.5 Closed Inertia Formulas . . . 10
  - 6.5.1 Solid Sphere . . . 10

7 Gazebo World from 2D Map **10**
- 7.1 Setup & Installation . . . 10
- 7.2 Generation of map.stl file . . . 11
- 7.3 Importing the mesh into Gazebo . . . 11

8 multimaster_fkie **11**
8.1 Networking 12
8.2 Package Installation 12
8.3 Setup & Usage 12
8.4 Testing 13

9 Gzweb **14**
9.1 Setup & Installation 14
9.2 Usage 14

10 Animating Models in Gazebo **15**

11 Converting from urdf.xacro to urdf & sdf **15**

12 RoboSense Helios-5515 3D lidar **15**
12.1 Installation 15
12.2 Usage 16

13 How to add collision checks (bumper sensors) in Gazebo **16**

1 Useful Resources

Robot modelling in urdf is rather tedious. Here are a couple of guides/tips which I have found useful.

- URDF/xacro guide
- Guide for gazebo plugins (used to simulate sensors/actuators in gazebo)
- Sonar Plugin Guide
- This is a folder which contains many gazebo .world files.

```
/usr/share/gazebo-9/worlds
```

2 Useful Tools

- Fusion 360 - A CAD software. Quite useful for measuring the dimensions of mesh files
- Mesh Lab - A piece of software to visualise and manipulate mesh files.

3 Installation

1. Install the joint state publisher:

```
$ sudo apt install ros-melodic-joint-state-publisher-gui
```

2. Next, git clone the **lb_msgs** package into your catkin/src folder.

3. Next, git clone realsense package into your catkin/src folder.

```
$ git clone
    https://github.com/SynapseProgramming/realsense_gazebo_plugin.git
```

4. Next, git clone the map2gazebo package into your catkin/src folder

```
$ git clone
    https://github.com/SynapseProgramming/map2gazebo.git
```

5. Enter the following command into the terminal to setup gazebo environment variables

```
$ echo "source /usr/share/gazebo/setup.sh" >> ~/.bashrc
$ source ~/.bashrc
```

6. Lastly, remember to run the cakin_make command.

4 Package Overview

4.1 world

The world folder is used to store gazebo .world files. .world files are used to describe the surrounding environment in Gazebo.

4.2 urdf

The urdf folder is used to store (.urdf.xacro) and (.gazebo.xacro) files.

- (.urdf.xacro) files describe the transformations between the various links of the robot, along with other variables such as intertia.
- (.gazebo.xacro) files would describe the kinematic properties of the robot such as velocity limits, coefficient of friction etc.

4.2.1 Intel Realsense D435 camera

- The (_d435.gazebo.xacro) file is used to describe the various kinematic properties of the depth camera.
- The (_d435.urdf.xacro) file is used to describe the various frames of the depth camera.

  The (<camera>_bottom_screw_frame) is used as the main reference point for the depth camera. It is centered about the bottom tripod mount of the depth camera. Thus, to define the position of the depth camera on the robot, we would have to provide the transformation from base link to the bottom screw frame link

4.2.2 cub_full_robosense.urdf.xacro

Cub with a robosense Helios-5515 3D lidar.

4.2.3 cub_full_urdf.xacro

Cub complete with all depth cameras, sonar, bumper and lidar sensors.

4.2.4 cub_full_shifted_urdf.xacro

URDF file used to test camera extrinsics calibration.

4.2.5 cub_full_issac_urdf.xacro

URDF file used to test issac sim stuff.

4.3 meshes

The meshes folder is used to store the various mesh files of the robot(.stl .dae). The mesh files are used to visualize various components of the robot(sensors, chassis) and to enable accurate collisions within gazebo.

4.4 rviz

The rviz folder is used to store the various rviz configs.

4.5 launch

- The launch folder is used to store roslaunch files. Launch files titled simulate_<robotname>.launch are used to launch the gazebo simulation environment and rviz.
- Launch files titled view_<robot>.launch are used to view the robots urdf model in rviz.
- Launch files titled <stuff>_robosense.launch are used to launch cub models with a robosense Helios 3D lidar.

5 Mesh Manipulation

5.1 Reducing the quality of a mesh (Mesh Lab)

In most cases, the generated meshes are too large for gazebo to render at a high framerate. Thus, we would have to reduce the resolution of our meshes.

1. Open meshlab and import the mesh (file > import mesh)
2. Click on Filters > Remeshing, Simplification and Reconstruction > Quadratic edge collapse decimation
3. Select a lower number of (target number of faces) and click apply.
4. click on file > export mesh as > (enter mesh name and select *.stl* file type)

5.2 Scaling Down a mesh (Fusion 360)

1. Click on insert > insert mesh
2. Select the correct units for the given mesh and select center position.
3. Click on mesh > modify > scale mesh
4. Click on the mesh and apply a scale factor of 0.1
5. Repeat steps 3 and 4 repeatedly until the desired scale factor is achieved (for example, this process would have to be repeated 3 times to scale from mm to m)
6. Save the project, and export the final mesh as a .stl file.

5.3 Mesh Origin

Each stl file has a origin. In most cases, we would want to modify the origin for various reasons.

5.3.1 Centering a mesh

1. Import a mesh into meshlab.
2. click on render > show axis
3. click on filters > Normals Curvature and Orientation > Transform: Move, Translate, Center
4. Select translate center of bbox to the origin and click apply.

In most cases, the centered origin is used to as the centre of mass. Thus, intertia is computed with respect to the centered origin.

5.3.2 Translating & rotating a mesh

- Under filters > Normals Curvature and Orientation, there are a few transform operations which could carried out on the mesh.

Do take note of the translation between the center origin of the mesh, and the final origin of the mesh as we would have to take that into account when defining the intertia reference frame.

6 Computing Intertia of meshes

In Gazebo, the inertia parameters should be well defined as they are used by the physics engine. Incorrect inertia parameters may result in odd occurences such as sliding on the spot.

6.1 Key Variables

- scaling factor s
  - URDF uses m (metres) to measure distance.
  - Most *.stl* files would be in mm (milimetres)
  - In this example, scaling factor $s = 10^2$ (the .stl file is in cm)

- Mesh Origin

  The center origin of the mesh would be used as the reference frame for intertia.

  Therefore, if the center origin of the mesh is not used as the main origin for the current part, we would have to specify the translation from the main origin to the center origin.

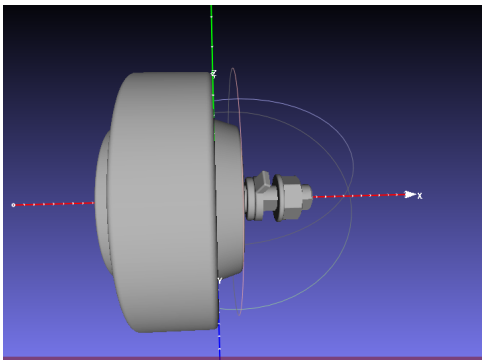

Figure 1: Wheel mesh with centered origin

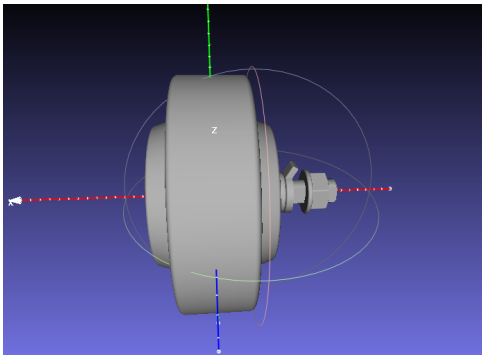

Figure 2: Wheel mesh with off-centered origin

6.2 Computing Intertia

1. Firstly, import the mesh into meshlab.
2. Ensure that the current origin of the mesh is its center.
3. Click on filters > Cleaning and Repairing > Remove Duplicate Faces

4. Click on view > Show Layer Dialog (ensure that a right window pops up)

5. Click on filters > Quality measure and computations > Compute Geometric Measures

6. At the bottom right corner, the following text would be generated.

```
Mesh Bounding Box Size 10.275001 13.000000 13.000000
Mesh Bounding Box Diag 21.061235
Mesh Volume is 575.673584
Mesh Surface is 1340.176147
Thin shell barycenter -0.288410 0.036296 0.007559
Center of Mass is -0.142327 0.002174 0.000383
Inertia Tensor is :
| 10287.553711 2.708627 0.324480 |
| 2.708627 6752.031738 -0.408094 |
| 0.324480 -0.408094 6754.208984 |
Principal axes are :
| 1.000000 -0.000770 0.000046 |
| 0.000766 0.983973 -0.178316 |
| 0.000092 0.178316 0.983973 |
axis momenta are :
| 10287.555664 6751.955566 6754.282715 |
```

- The Intertia Tensor generated by MeshLab would correspond to the following matrix

$$\begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix}$$

- As the mesh used in this example is in cm, the mesh volume would be in cm^3

6.3 Importing Inertia Values into URDF

1. Let v be the volume of the mesh in m^3

   As the generated mesh volume is in cm^3, we would have to multiply the mesh volume by 10^{-6} to scale it to m^3.

   $v = 575.673584 \times 10^{-6}$

2. Let m be the mass of the given object in kg

   In this case, we will assume that the wheel weighs $5kg$

3. Each element in the inertia matrix would have to be scaled down by s^5, where s is the scaling factor. (eg. s=10^2 for a mesh in cm. Thus, each I value would have to be multiplied by 10^{-10})

   Furthermore, each element in the inertia matrix would have to be multiplied by mass m and divided by volume v

$$\frac{m}{vs^5}\begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix}$$

4. Next, we would have to fill up the inertia tag in the URDF file.

   - *origin* tag should be filled with the translation from the main origin to the center origin (if neccessary)
   - *inertia* tag should be filled with the scaled inertia values.
   - *mass* tag should be filled with mass m.

```
<inertial>
  <origin xyz="0 0 0"/>
  <mass value="5"/>
  <inertia ixx="0.00893523" ixy="2.35258e-06"
      ixz="2.81826e-07" iyy="0.00586446" iyz="-3.54454e-07"
      izz="0.00586635"/>
</inertial>
```

5. In the simulation, the center of mass may cause the robot to drift forward. Therefore, the origin inertia tag should be tuned if needed.

6.4 Verifying Inertia Values

If the inertia values were computed correctly, then the mesh should have a pink box surrounding the mesh in Gazebo (view > inertia)

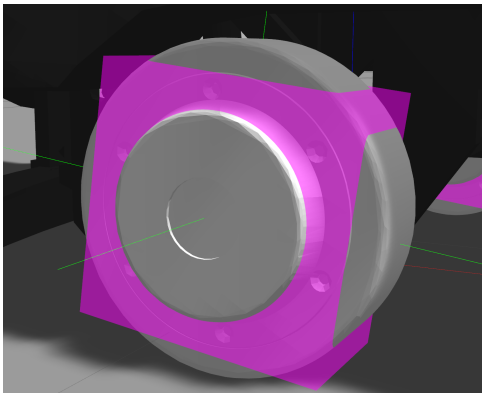

Figure 3: Wheel with inertia bounding box

6.5 Closed Inertia Formulas

6.5.1 Solid Sphere

- m Mass in kg
- r radius of sphere in metres

$$\begin{bmatrix} I_{xx} = \frac{2}{5}mr^2 & I_{xy} = 0 & I_{xz} = 0 \\ I_{xy} = 0 & I_{yy} = \frac{2}{5}mr^2 & I_{yz} = 0 \\ I_{xz} = 0 & I_{yz} = 0 & I_{zz} = \frac{2}{5}mr^2 \end{bmatrix}$$

7 Gazebo World from 2D Map

7.1 Setup & Installation

1. Ensure that the map server is installed on your system.

```
$ sudo apt install ros-melodic-map-server
```

2. Clone the *map2gazebo* repository into your catkin/src folder.

3. Navigate to the *map2gazebo* package in your catkin/src folder and run the following commands.

```
$ pip install --user trimesh
$ pip install --user numpy
$ pip install --user pycollada
```

```
$ pip install --user scipy
$ pip install --user networkx
```

4. Lastly, navigate to your catkin_ws folder and run catkin_make

7.2 Generation of map.stl file

1. launch roscore
2. Navigate to the folder containing the *pgm* and *yaml* file for the map that you would want to convert and run the following command to launch the map server.

```
$ rosrun map_server map_server <map name>.yaml
```

3. Then, run the following command to generate the *map.stl* at a given directory. (replace */home/roald/Desktop/generatedmaps* with your own path)

```
$ roslaunch map2gazebo map2gazebo.launch
    export_dir:=/home/roald/Desktop/generatedmaps
```

7.3 Importing the mesh into Gazebo

1. Firstly, navigate to the following directory

```
catkin_ws/src/map2gazebo/models/map/meshes
```

2. Replace the existing map.stl file with the newly generated map.stl file.
3. Launch the following command to launch the gazebo simulation with the current robot in the new map.

```
$ roslaunch fake_world simulate_cub_lab.launch
```

8 multimaster_fkie

Multimaster_fkie is a collection of packages which allows topics to be passed between separate devices, while running separate masters.

8.1 Networking

- All machines should be connected to the same local network
- DHCP reservation could be used to ensure that the machines have a static IP address

8.2 Package Installation

1. Enter the following lines to install the package dependencies.

```
$ sudo add-apt-repository ppa:roehling/grpc
$ sudo apt update
$ sudo apt install python-grpcio python-grpc-tools
```

2. Next, navigate to your catkin/src folder and run the following command.

```
$ git clone
    https://github.com/fkie/multimaster_fkie.git
```

3. Lastly, run catkin_make to build the package.

8.3 Setup & Usage

In this section, we shall assume that we would want to pass data between two machines, *M1* and *M2*.

1. Ensure that the multimaster_fkie package is installed on both machines.
2. Ensure that only one networking option is enabled (either wifi or ethernet)
3. Next, run the *ifconfig* command on *M1* and *M2* and note down the assigned ip address.
4. Next, navigate to the *etc* directory and run the following command

```
$ sudo gedit hosts
```

5. Add the ip addresses of M1 and M2 to the hosts file (replace the fake ip-addresses with real ones and replace *M1* and *M2* with the names of your machines)

   (example hosts file for *M1*)

```
127.0.0.1 localhost
127.0.1.1 M1

#for ROS multimaster
12.13.14.15 M1
14.13.153.134 M2
```

6. Repeat steps 3 and 4 on the other machine. (eg. *M2*)

7. Ensure that roscore is running on both machines *M1* and *M2*

8. Run the following commands in separate terminals to sync up the two rosmasters on *M1* and *M2*

```
$ rosrun fkie_master_discovery master_discovery
$ roslaunch fkie_master_sync master_sync.launch
```

8.4 Testing

1. Firstly, ensure that roscore, the discovery and the master sync nodes are running on both machines *M1* and *M2*.

2. Next, run the following command to check if both masters are known by the discovery node. Ideally, both masters on the separate machines should be listed here.

```
$ rosservice call /master_discovery/list_masters
```

3. Next, run the following command in *M1* to run the turtlesim node.

```
$ rosrun turtlesim turtlesim_node
```

4. Next, run the following command in *M2* to run the turtlesim teleop node.

```
$ rosrun turtlesim turtle_teleop_key
```

5. Thus, if all went well, pressing the arrow keys on *M2* should move the turtle on *M1*.

9 Gzweb

9.1 Setup & Installation

1. Firstly, open a new terminal and enter the following commands. you may wish to refer to *this gazebo webpage*

```
$ sudo apt install gazebo9 libgazebo9-dev
$ sudo apt install libjansson-dev libboost-dev imagemagick
    libtinyxml-dev mercurial cmake build-essential
$ curl -o-
    https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.3/install.sh
    | bash
$ source ~/.bashrc
$ nvm install 8
$ cd ~; git clone https://github.com/osrf/gzweb
$ cd ~/gzweb
$ git checkout gzweb_1.4.1
$ npm run deploy --- -m
```

2. Next, navigate to the gzweb/http/client/assets directory and copy and paste the fake_world package in the assets directory.

3. Navigate to the gzweb directory and run *npm install*

9.2 Usage

1. Navigate to the gzweb directory and run *npm start*

2. Lastly, ensure that the launch file in the simulation package has the GUI tag disabled, before proceeding to enter the roslaunch command.

```
        <arg name="gui" value="false"/>
```

3. In order to view the gzweb page on another machine, the URL should be <ip address of server>:8080

10 Animating Models in Gazebo

- To animate models in Gazebo, we would have to write a model plugin, and specify that plugin from within the model tag.
- In the world file, the name of the plugin would be lib<name of plugin in Cmake>.so

```
#in CMake
add_library(movebox src/movebox.cc)
#In .world file
<plugin name="push_animate" filename="libmovebox.so"/>
```

11 Converting from urdf.xacro to urdf & sdf

1. To convert a existing urdf.xacro file to a urdf file, please run the following command (and replace bot with the name of your file)

```
$ xacro bot.urdf.xacro > bot.urdf
```

2. To convert a existing urdf file to a sdf file, please run the following command (and replace bot with the name of your file)

```
$ gz sdf -p bot.urdf > bot.sdf
```

12 RoboSense Helios-5515 3D lidar

12.1 Installation

1. Firstly, git clone the **robosense_simulator_Helios_5515** package into your catkin/src folder and run catkin_make
2. **(optional) update gazebo to version 9.4.0++**
   - This step is required if you want the GPU to compute the point-cloud.
   - **If you do not carry out this step, please set *gpu="false"* in the *xacro::RS-5515 tag***

- Navigate to the robosense_simulator_Helios_5515 directory and refer to *gazebo_upgrade.md* for a more detailed explanation on the upgrade process.

12.2 Usage

- After installing the package, please add the following lines to your robots *.urdf.xacro* file.

```
<xacro:include filename="$(find
    robosense_description)/urdf/RS-5515.urdf.xacro"/>

<xacro:RS-5515 parent="base_link" name="helios" hz="10"
    samples="1800" gpu="true" noise="0.002">
  <origin xyz="0 0 1.35" rpy="0 0 0"/>
</xacro:RS-5515>
```

- You may wish to refer to the *example.urdf.xacro* file which can be found in the *urdf* directory of the Robosense simulator package.

13 How to add collision checks (bumper sensors) in Gazebo

1. The first step would be to add the following sensor tag to a gazebo reference (link).

   Do change the tags accordingly.

```
<gazebo reference="left_bumper">
  <sensor type="contact" name="left_bumper_contact_sensor">
    <selfCollide>true</selfCollide>
    <update_rate>10.0</update_rate>
    <always_on>1</always_on>
    <contact>
      <collision>
        base_footprint_fixed_joint_lump__left_bumper_collision_8
      </collision>
    </contact>
    <plugin name="gazebo_ros_bumper_controller"
        filename="libgazebo_ros_bumper.so">
      <bumperTopicName>/left_bumper</bumperTopicName>
      <update_rate>10.0</update_rate>
```

```
      <always_on>1</always_on>
      <frameName>base_link</frameName>
    </plugin>
  </sensor>
</gazebo>
```

2. The collision tag is slightly problematic. To fill up the collision tag properly, please generate the sdf file of your robot model by following the steps in **this** section.

3. Next, open up the generated *.sdf* file and ctrl+f the name of your link. Copy the name of your link which ends with *_collision_X* and paste it in the collisions tag.