Param Bridge

Release 1.1.1

Aravind

Nov 04, 2022

CONTENTS:

This module is designed to enable direct communication between ROS 2 and 3D Slicer. You can find more details regarding this module in Bridging 3D Slicer and ROS2 for Image-Guided Robotic Interventions .

This is an early prototype and we hope to add more features soon.

CHAPTER

ONE

FEATURES

The main feature currently supported is the ability to visualize a robot within Slicer. The default approach is similar to RViz, i.e.:

- Load a robot description from ros parameter `robot_description`, create a MRML Slicer node for each link and load meshes
- Update the links positions using tf2, assuming there is a `robot_state_publisher` running

Alternatively, the module can load an URDF file directly and use KDL for the kinematic chain. This feature only works with serial robots (no parallel mechanisms). This is not the recommended approach.

This is an early prototype and we hope to add more features soon.

CHAPTER

TWO

GETTING STARTED

2.1 Pre-requisites

- ROS 2 (tested using ubuntu 20.04 and ROS2 Foxy or Galactic).
- Slicer built from source is required to build an extension, see Slicer build instructions. **Remember the build directory for Slicer, it will be needed to compile the Slicer ROS 2 module.**

Warning: Make sure we use the system/native OpenSSL libraries otherwise you'll get some errors when compiling the Slicer ROS 2 module. After you ran CMake, in the Slicer build directory, set `Slicer_USE_SYSTEM_OpenSLL ON` using `cmake . -DSlicer_USE_SYSTEM_OpenSSL=ON` or `ccmake`.

Note: *Older Slicer* Make sure `CMAKE_CXX_STANDARD` is set to `14` (required to compile Slicer code along ROS 2).

2.2 Compilation

This code should be built with `colcon` as a ROS2 package. For now, we will assume the ROS workspace directory is `~/ros2_ws` and the source code for this module has been cloned under `~/ros2_ws/src/slicer_ros2_module`.

You will first need to source the ROS setup script for ROS 2 (foxy or galactic):

Then build the module using *colcon* while providing the path to your Slicer build directory:

Note: The `-DSlicer_DIR...` option is only needed for the first `colcon build`.

If you prefer to use CMake (`ccmake`) instead of passing the `Slicer_DIR` on the colcon command line, you can run `colcon build` once and then run `ccmake` on the `slicer_ros2_module` build directory.

Warning: You should see the following error messages if the `Slicer_DIR` is not set properly (or if Slicer has not been built from scratch):

```
Could not find a package configuration file provided by "Slicer" with any
of the following names:

SlicerConfig.cmake
slicer-config.cmake
```

If you see this message, run CMake on the build directory for `slicer_ros2_module` using `ccmake ~/ros2_ws/build/slicer_ros2_module`. In CMake, set `Slicer_DIR` to point to your Slicer build directory then hit `c` to configure until you can hit `g` to generate the makefiles. Then try to `colcon build` again (after `cd ~/ros2_ws`).

CHAPTER

THREE

STARTING A ROS ROBOT_STATE_PUBLISHER

The following are examples of robots we've used to test the Slicer ROS2 module.

1. Note that we tried to follow the standard ROS approch, i.e. make the URDF description available as a ROS parameter and then launch the usual ROS `robot_state_publisher` node.
2. The `robot_state_publisher` will use the joint state to compute the forward kinematic and broadcast the 3D position of each link to tf2.
3. The Slicer ROS 2 module will then query the 3D positions to display the robot's links in the correct position.

3.1 Sensable Phantom Omni aka Geomagic/3DS Touch

We created and used the following package for the Omni: Omni Github Link

This package contains the URDF, STL meshes and a launch file for the `robot_state_publisher`. **Make sure you start the Omni nodes (using a couple of terminals) before loading the Slicer ROS 2 module.**:

```
ros2 launch sensable_omni_model omni.launch.py # first terminal
ros2 run sensable_omni_model pretend_omni_joint_state_publisher # second
```

3.2 dVRK PSM

todo

3.3 Cobot

We also tested SlicerROS2 on myCobot by Elephant Robotics, specifically the myCobot 280 M5 Stack. The ROS 2 interface for the device can be found here and drivers can be installed from the Elephant Robotics website.

Assuming the interface (mycobot_ros2) is cloned under the same ros2_ws, the state publisher can be started using the following steps:

```
cd ~/ros2_ws/src/mycobot_ros2/src/mycobot_ros2/mycobot_280/mycobot_280/config
python3 listen_real.py
```

It's possible that you will need to change the port specified on line 14 of `listen_real.py` depending on your device. The `.dae` files in the robot description also need to be converted to STLs (an online converter will work) and the paths in the URDF file should be updated to reflect this change.

Once running - make sure your robot is in *Transponder Mode*. More instructions for basic operation of the myCobot can be found in the Gitbook

CHAPTER

FOUR

USING THE SLICER ROS 2 MODULE

In a terminal navigate to your Slicer inner build directory (`cd`). Then:

```
source ~/ros2_ws/install/setup.bash # or whatever your ROS 2 workspace is
# you can ignore the error message related to slicer_ros2_module/local_setup.bash
./Slicer
```

Note: The first time you run Slicer, you need to add the module directory should be in the application settings so that it can be loaded.

To do so, open Slicer and navigate through the menus:

Edit → *Application Settings* → *Modules* → *Additional module paths* → *Add*:

```
~ros2_ws/build/slicer_ros2_module/lib/Slicer-4.13/qt-loadable-modules
```

At that point, Slicer will offer to restart. Do so and then load the module using the button:

Modules → *Examples* → *Ros2*

The module's interface will appear on the left side of Slicer. You should leave the first two drop-down menus as-is, i.e. `State` should be `tf2` and `Description` should be `parameter`.

The defaults `/robot_state_publisher` and `robot_description` should work for most cases so leave these as-is too.

To activate your selection, just hit "Enter" in the `/robot_state_publisher` box. Please note that we plan to improve this user interface.

At that point, the robot's model should be loaded and displayed in Slicer.

CHAPTER

FIVE

KNOWN LIMITATIONS

- **We only support STL and OBJ meshes for the `visual` defined in the URDF.**
  - If any `visual` is defined using a `geometry` (sphere, box…), it will not be displayed in Slicer
- The current module only supports one robot at a time
- There is no mechanism to synchronize the MRML scene in Slicer with tf2 in ROS 2

CHAPTER

SIX

USEFUL LINKS

- Source code: https://github.com/rosmed/slicer_ros2_module/