Gazebo/Ignition Integration with SlicerROS2

Simon Leonard

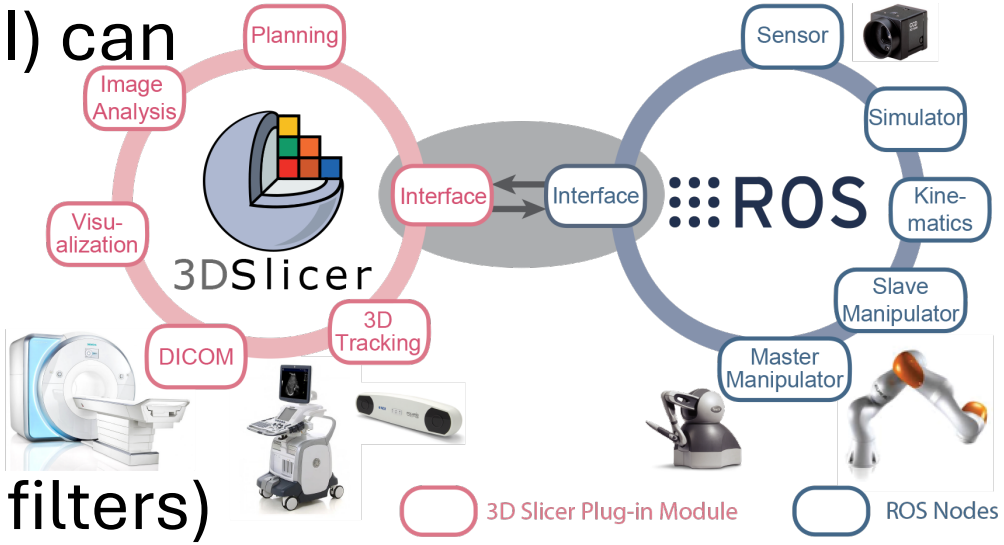
06/03/2024

3D Slicer vs ROS2

- ROS: Open source middleware used in robotics. Development for engineers working on
  - Industrial, field and mobile (U?V) robots.
    - Supports several sensors and hardware used in these applications.
  - Medical? With a few exceptions (i.e. dVRK)…not so much.
- 3D Slicer: Open source software used for medical imaging analysis and visualization
  - Supports multiple algorithms for registration, segmentation, vizualization
  - Image-guided robotics intervention? With few exception (OpenIGTLink)…not so much.

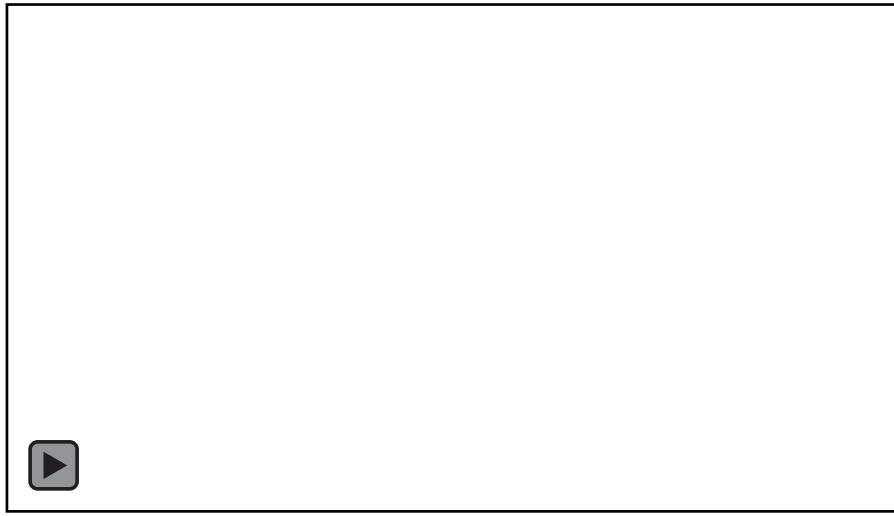
SlicerROS2

- Converts 3D Slicer into a ROS node
- Image-guided robotics interventions (IGRI) can access the ROS goodies from Slicer
  - Topics (supports “stock” messages)
  - TF
  - Parameters
  - TODO: services and actions
- Launch ROS processes (robots, sensors, filters) and access them from Slicer
  - Subscribe/publish data
  - Path planning and control algorithms



Open Source Dynamic Simulation

- ROS has an unfair advantage from dynamic simulation
  - Interchangeable with real hardware
  - Simulates rigid body dynamics: forces, friction, impulse
  - Simulates sensors: camera (thermal, RGB, RGBD), LiDAR, sonar, GPS, IMU, compass, altimeter, etc.
- Dynamic simulation has been a key ingredient in the progress of robotics research:
  - Cheap (decent computer)
  - Accelerates the development cycle (no need for calibration)
  - Test workspace, joint limits, synchronization
  - Excellent for testing prototypes (perfect calibration)

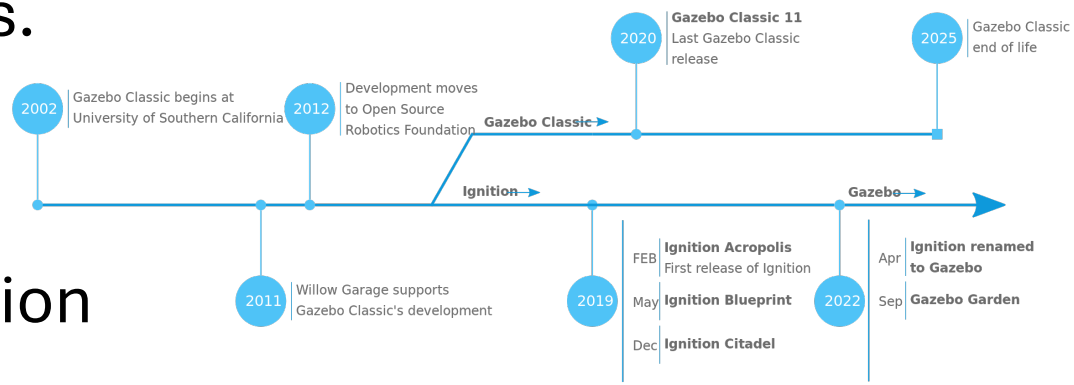


Simulation of IGRI

- Like all robotics applications, IGRI experiments require extensive preparation
  - Limited access to hardware.
  - Often run on multiple workstations running different OS.
  - Maintenance nightmare.
- Preliminary experiments should not require elaborate calibration
  - Test if the control algorithm works in reasonable situation
  - Does a path planner work at all?
- Unfortunately, there is little/no support for sensors commonly used for IGRI: ultrasound, OCT, MRI, CT, X-ray, tracking.

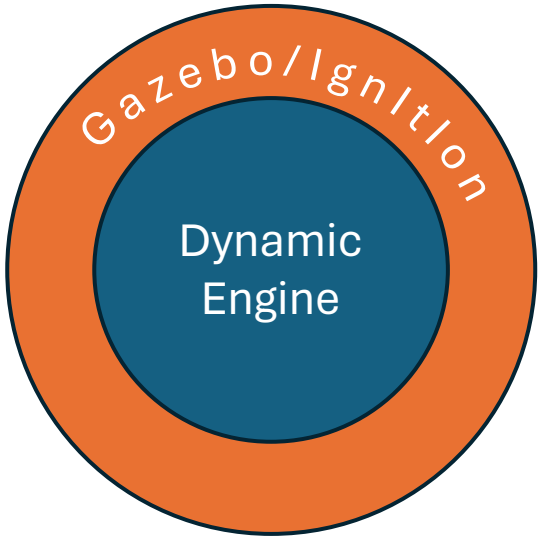
Gazebo History

- Initially part of Player/Stage/Gazebo (late 90s)
- ROS/OSRF revived Gazebo in late 2000s.
- Lessons learned from numerous challenges and years of usage lead to Ignition as the successor of Gazebo
- Following trademark infringement, Ignition was renamed Gazebo and what was formally known as Gazebo was renamed Gazebo Classic
- As of today, many packages still bear the “ignition” name but they are being gradually renamed.



Gazebo Overview

- Gazebo/Ignition does not implement dynamic simulation
- Instead, it wraps code around dynamic simulation code (3rd party) to make it easier to use and expand
- Examples of dynamic engines supported:
  - Open Dynamic Engine
  - Bullet 3d
  - Dart



Extending Gazebo

- Like many related software, Gazebo is designed to extend the capabilities of simulation
  - MoveIt!: add new path planning
  - ROS2 Control: add new control algorithms
  - Rviz: add new widgets
  - Slicer: add new modules
- Gazebo uses plugins to add new sensors, controllers, visualization
  - Virtually all Gazebo sensors are plugins.
  - Sensor plugins are not that difficult to implement.
- No plugins for medical devices typically used in IGRI.

Gazebo Plugin

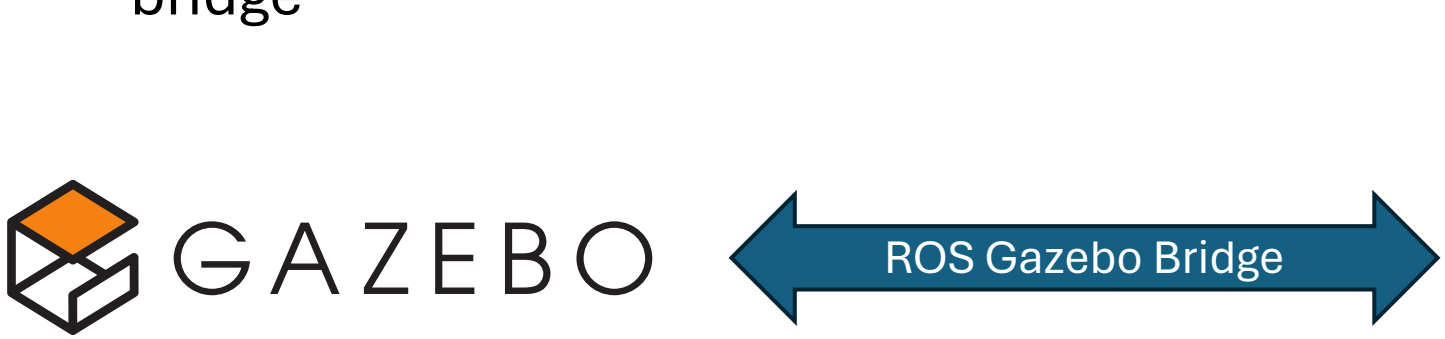
- Piece of C++ code build into a library that is “thrown into” the simulation at runtime with the expectation that Gazebo will execute the code at each iteration.
- Very little limits to what you can do (whatever you can do with C++ can go inside a plugin).
- If you have a mathematical model of something you want to simulate, all you need to do is to “shape it” as a Gazebo plugin and register it (it’s not that bad!)
- These are built and distributed as ROS packages

```
// Code
class custom_plugin:
  public ignition::gazebo::System,
  public ignition::gazebo::ISystemConfigure,
  public ignition::gazebo::ISystemPreUpdate,
  public ignition::gazebo::ISystemUpdate,
  public ignition::gazebo::ISystemPostUpdate
{
};


<!– URDF -->
 <plugin filename=“custom_plugin_library
name=“custom_plugin"/>
```

Gazebo vs ROS

- Information exchange between ROS and Gazebo
  - Gazebo classic: Plugin implemented a ROS node from which topics could be published/subscribed
  - Gazebo: Implements its own node, its own publisher/subscriber and its own messages. Such that messages between ROS and Gazebo must be “bridge”



Preview of Gazebo Plugin for IGRI

- Toy example of a plugin
  - Simple components
  - Move a 1DoF robot
- Create an ultrasound sensor
  - Use code from PlusTookit (https://github.com/PlusToolkit/)
  - Wrap the US simulation in a Gazebo plugin
  - Add the US sensor to a UR5 (with FT sensor)
  - Create a small world with a block of gel and vessel
  - Run the simulation and teleoperate the robot
  - Display the robot and simulated US images in 3D Slicer

What’s next?

- X-rays, OCT, etc. Any 2D data should be relatively “easy” to wrap into Gazebo (as long as there is code for it).
- MRI, CT. A bit more complex because there is currently no 3D data type that is compatible with this data.
  - Maybe hack 3D point cloud
  - Define custom data type (not that of a bit deal) in both Gazebo and ROS and a bridge to convert the data

- If you need a Linux Docker image follow these instructions during the break
  - It has PlusToolkit, Slicer and SlicerROS2 installed
  - https://rosmed.github.io/ismr2024

- Code for the plugins will be taken from
  - https://github.com/rosmed/ismr24_plugin (toy plugin)
  - https://github.com/rosmed/ismr24 (US sensor plugin, transducer URDF, resolved rate motion controller for the UR5)

- If using the Docker image, you can download and run these commands
  - wget www.cs.jhu.edu/~sleonard/ismr24.sh
  - chmod 755 ismr24.sh
  - ./ismr24.sh