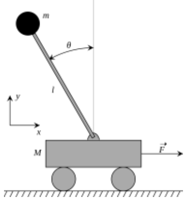
Final Project Overview

Carmelo Scribano

carmelo.scribano@unimore.it



UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

High Performance Real Time Lab

Recap

- Standard project (these slides):
  - Complete **two** assigments using ROS2 and the other technologies presented in class.

- Advanced project:
  - To be agreed individually with the professor.
  - May involve advanced topics not covered in class

Standard Project Assignment

Overview

› **TWO** assigments to be completed:

1. Control of the Cart-Pole system

- Completing the first assigment is **always required.**
- A correct solution entitles you to pass the exam (accordingly to the written/oral part).

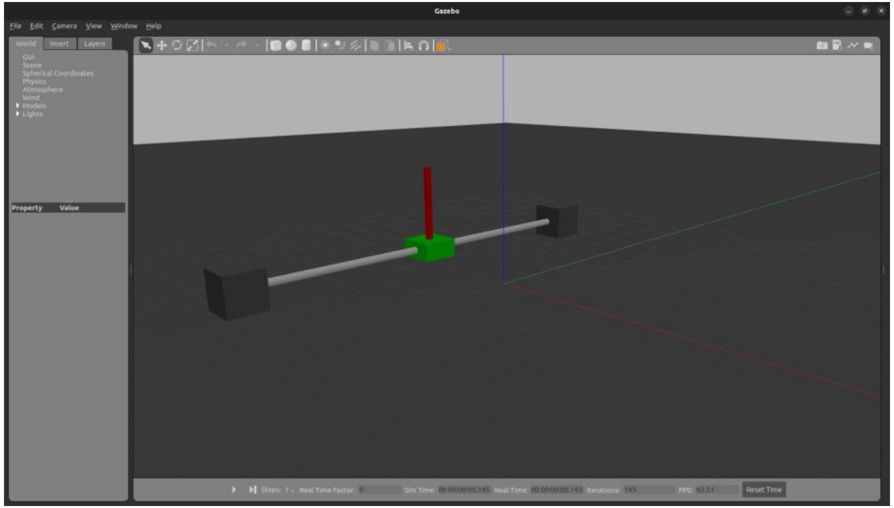
2. Autonomous navigation

- Only after completing the first assignment
- Allows you to obtain a higher score (if done properly..)

Part 1 – Cart-Pole Control

Resources

- The simulation environment is provided as a ROS2 Package: https://github.com/cscribano/gazebo_polecart_ros
- Clone and build the simulation package, then launch it!
  - More details about the simulation in slides 05.



Simulation Environment

- Subscribed: /cart/force
  - Type: geometry_msgs/Wrench
  - Used to apply a force (N) to the cart. Note: the cart can only move on the **y** axis!!
- Published: /cart/pole_state
  - Type: sensor_msgs/JointState message
  - Used to publish the pole-cart angle (in radians). The desired setpoint is 0.0 degrees (pole standing up)

Suggested Approach

- Write a PID Controller to stabilize the system.
  - You can find a lot of PID implementations to take inspiration

IDEA (control Loop):

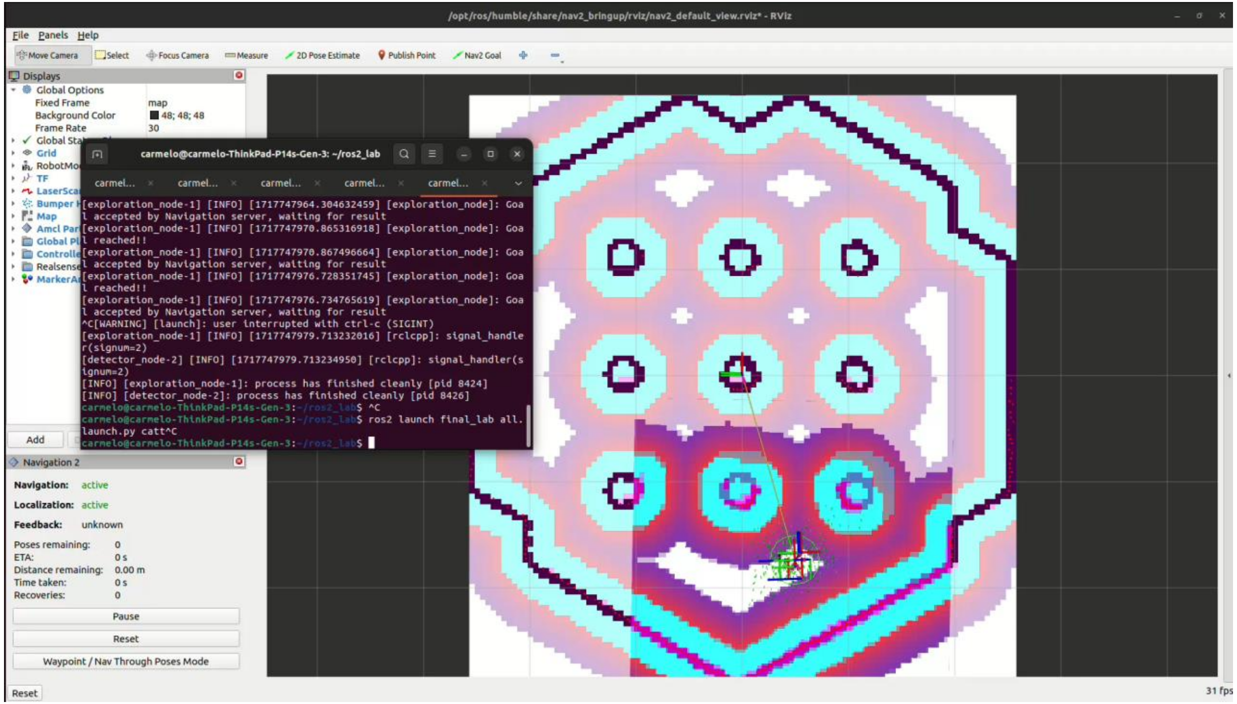
1. Read the pole state, use the reading to compute the error (wrt. 0.0rad setpoint)
2. Update the PID control to obtain a control value
3. Publish the control signal to the control topic.

TIP: zero-out the force and the simulator before starting the control loop.

Part 2 – Autonomus Navigation

Example assigment

› Autonomusly navigate the map to find an ArUco Marker
  – (Demo video available on Moodle)



Requirements

- The EXAMPLE assigment use the **Turtlebot3** simulation environment and the **Nav2** Navigation stack.
- The provided «**starter pack**» folder (on Moodle) contains:
  - Gazebo Model for the Cubic ArUco Marker (aruco_box.zip)
  - Example OpenCV code for ArUco based pose estimation (without ROS) (cv_aruco_pose.zip)
- Install the marker in gazebo:
  - Extract the «aruco_box.zip» archive, then copy the extracted folder to Gazebol model path (by default /home/<your_name>/.gazebo/models)

Nav2 Actions

- By default Nav2 expose ROS2 **Actions**.

```
$ ros2 action list -t
/assisted_teleop [nav2_msgs/action/AssistedTeleop]
…
/navigate_to_pose [nav2_msgs/action/NavigateToPose]
```

- **navigate_to_pose** can be used to set the navigation goal and check the outcome.
- Show the interface:

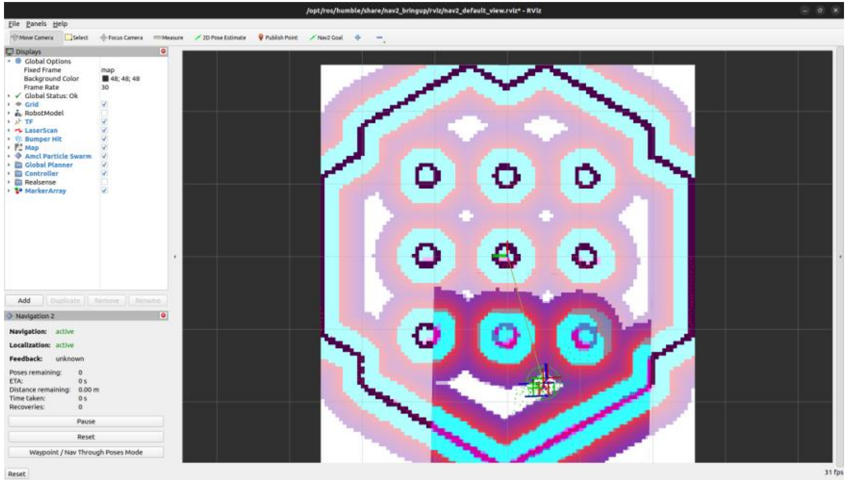
```
$ ros2 interface show nav2_msgs/action/NavigateToPose
```

- Send example goal (from CLI)

```
$ ros2 action send_goal /navigate_to_pose nav2_msgs/action/NavigateToPose "{pose: {header: {stamp: {sec: 0, nanosec: 0}, frame_id: 'map'}, pose: {position: {x: 0.5, y: 0.5, z: 0.0}, orientation: {x: 0.0, y: 0.0, z: 0.0, w: 1.0}}}}"
```

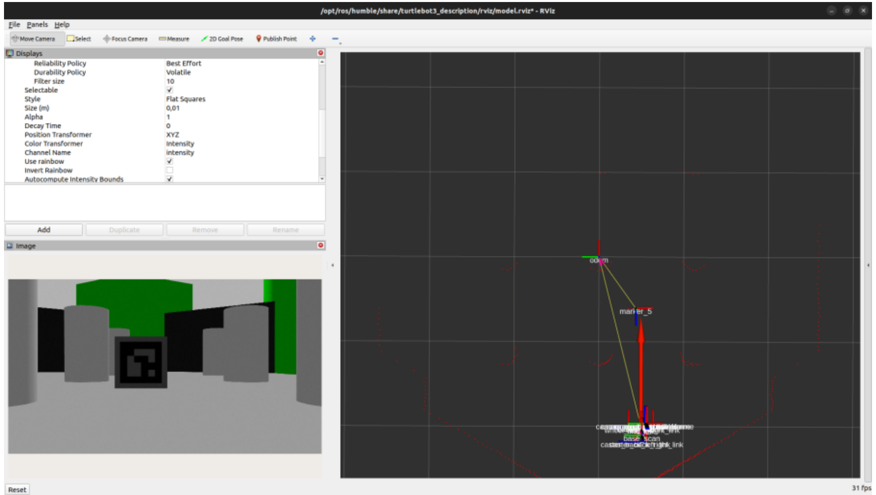
Exploration Node

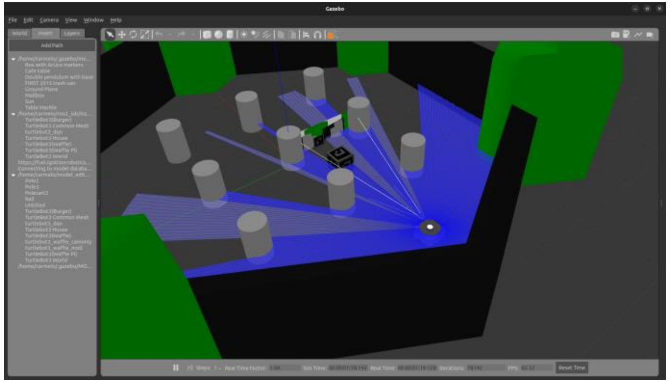
- The Exploration Node interacts with the Nav2 stack to navigate to autonomously navigate the (previously mapped) environment.
- You can define a predetermined set of navigation goals, or compute at runtime new target locations (i.e, using the Laser sensor)



Detection Node

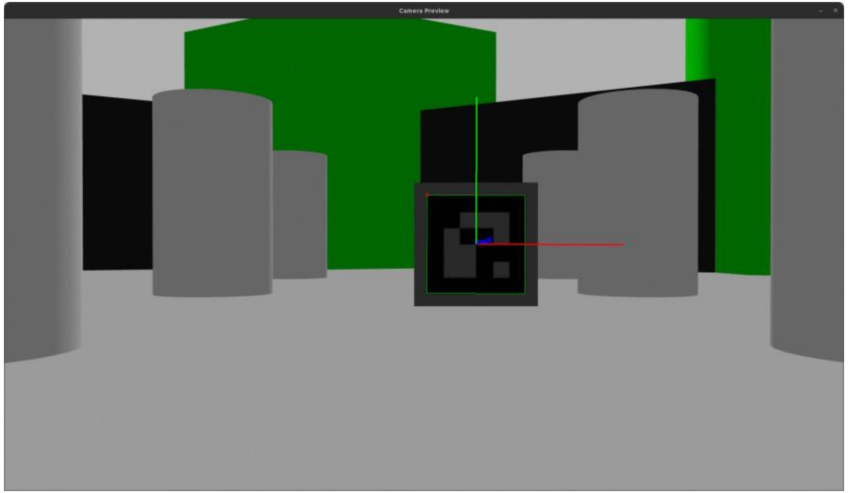
- The Exploration Node analyze the camera feed by acquiring messages from the camera topic
  - Once a Marker is detected, the pose (camera->marker) is computed using the Opencv function
  - The marker is referenced in map frame (or odom if map is not published?) and the map->marker transform is boradcasted.





Sample Pose Estimation Code

- The provided C++ code implement Pose estimation using Opencv.
  - Build and run the code using Cmake (like we did in slides 03..). An example image is also provided for testing.
  - You might need to wrap this code in ROS2...



TIPS

- Can use TF Lookup and TF boradcasting to retrieve and manage transforms (slides 05, or docs https://docs.ros.org/en/humble/Tutorials/Intermediate/Tf2/Tf2-Main.html).
- For exploration, you can define a next pose to reach (statically or dinamically) and use the navigate_to_pose action to navigate to it.
- Writing an action client is required (https://docs.ros.org/en/humble/Tutorials/Intermediate/Writing-an-Action-Server-Client/Cpp.html#writing-an-action-client).
- Properly configure your CMakelist.txt (the sample code from previous labs can help!)