Rcss3d Nao

ijnek

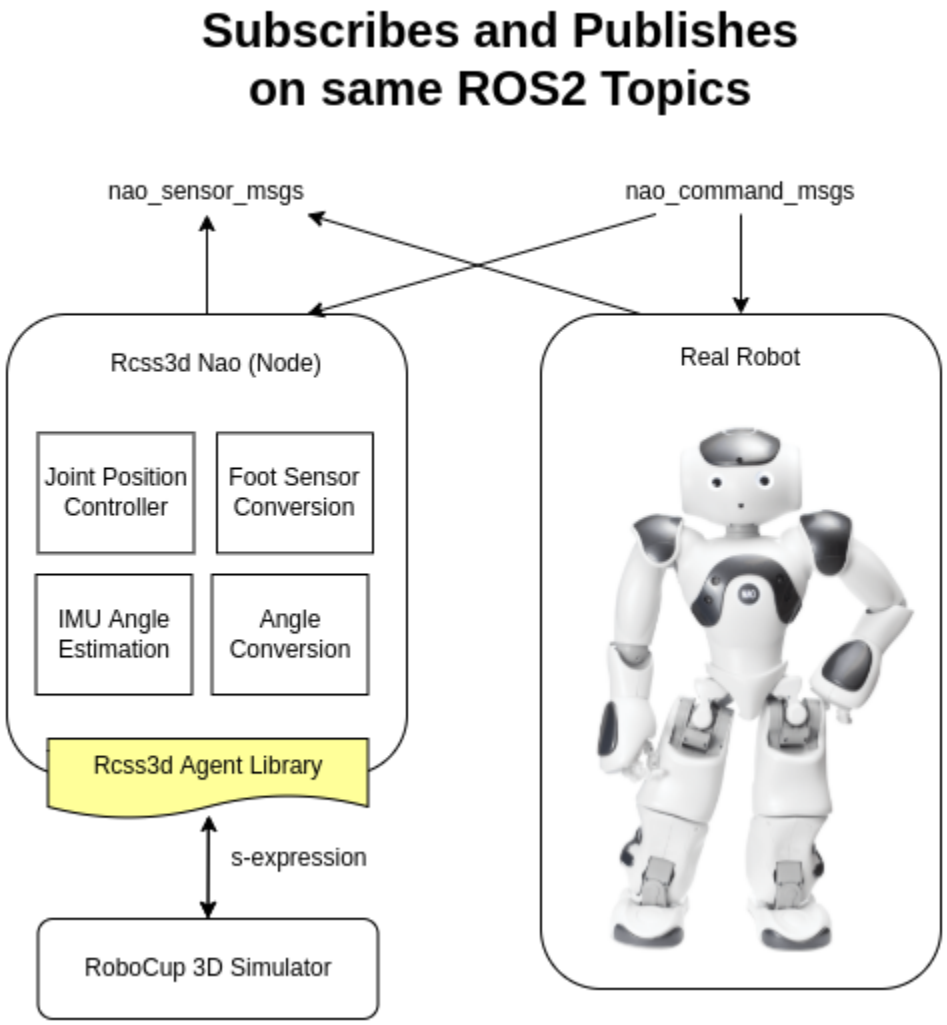
Jul 21, 2022

CONTENTS

Rcss3d Nao is a ROS2 package that provides an interface to the RoboCup 3D Simulator SimSpark that closely matches a Softbank Nao robot interface.

The package aims to provide an interface that closely matches that of the the physical Softbank NAO so it can be used in the RoboCup Standard Platform League.

Publishing / Subscription is performed using interfaces specific to the Nao (nao_command_msgs and nao_sensor_msgs), such that this simulated Nao robot can be substituted out for a real robot seamlessly.



The project is hosted on Github by ROS Sports.

CHAPTER

ONE

INSTALLATION

To install the packages, do one of the following:

- a *Binary Installation*
- a *Source Installation*

1.1 Binary Installation

Binary installation is available for ROS 2 Rolling only. Please follow source installation instructions for older distros.

Rcss3d Nao is released to the ROS ecosystem. You can install the debian packages using apt as following:

```
sudo apt update
sudo apt install ros-${ROS_DISTRO}-rcss3d-nao
```

If this method does not work for your platform, perform the *Source Installation* instead.

1.2 Source Installation

Source installation should work for almost any platform that has ROS2 installed on it. ROS2 Galactic onwards is supported. There is no alternative for ROS1 distros or older ROS2 distros.

From within a ROS2 workspace, run the following code that clones Rcss3d Nao into your src directory, and builds your workspace:

```
git clone https://github.com/ros-sports/rcss3d_nao.git src/rcss3d_nao --branch ${ROS_
↪DISTRO}
rosdep install --from-paths src --ignore-src
colcon build
```

1.3 Confirming Installation

To confirm that the packages have installed correctly, source either your:

- ros workspace if you did the binary installation (ie. `source /opt/ros/${ROS_DISTRO}/setup.bash`)
- current workspace if you did the source installation (ie. `source install/local_setup.bash`)

Check that your ROS2 package is installed correctly by asking for its path:

```
ros2 pkg prefix rcss3d_nao
```

CHAPTER

TWO

GETTING STARTED

This is a tutorial on how to launch a simulated NAO robot in the SimSpark simulator using ROS2.

Attention: This tutorial assumes that you have set up rcssserver3d on your computer. If you haven't done so already, go to SimSpark's Gitlab and follow the installation instructions.

2.1 Starting the Simulator

In a terminal, start the simulator by running:

```
rcsoccersim3d
```

Tip: Simulator tends to crash sometimes when connecting / disconnecting agents, which leaves unwanted server processes lingering. To kill this process, run `pkill -9 rcssserver3d` before restarting the simulation server.

2.2 Launching a Player

To run rcss3d_nao as a standalone application, in a new terminal, run:

```
ros2 run rcss3d_nao rcss3d_nao
```

Moving around the simulator camera with the WASD keys and the mouse, you should see your robot at the corner of the field, as below:

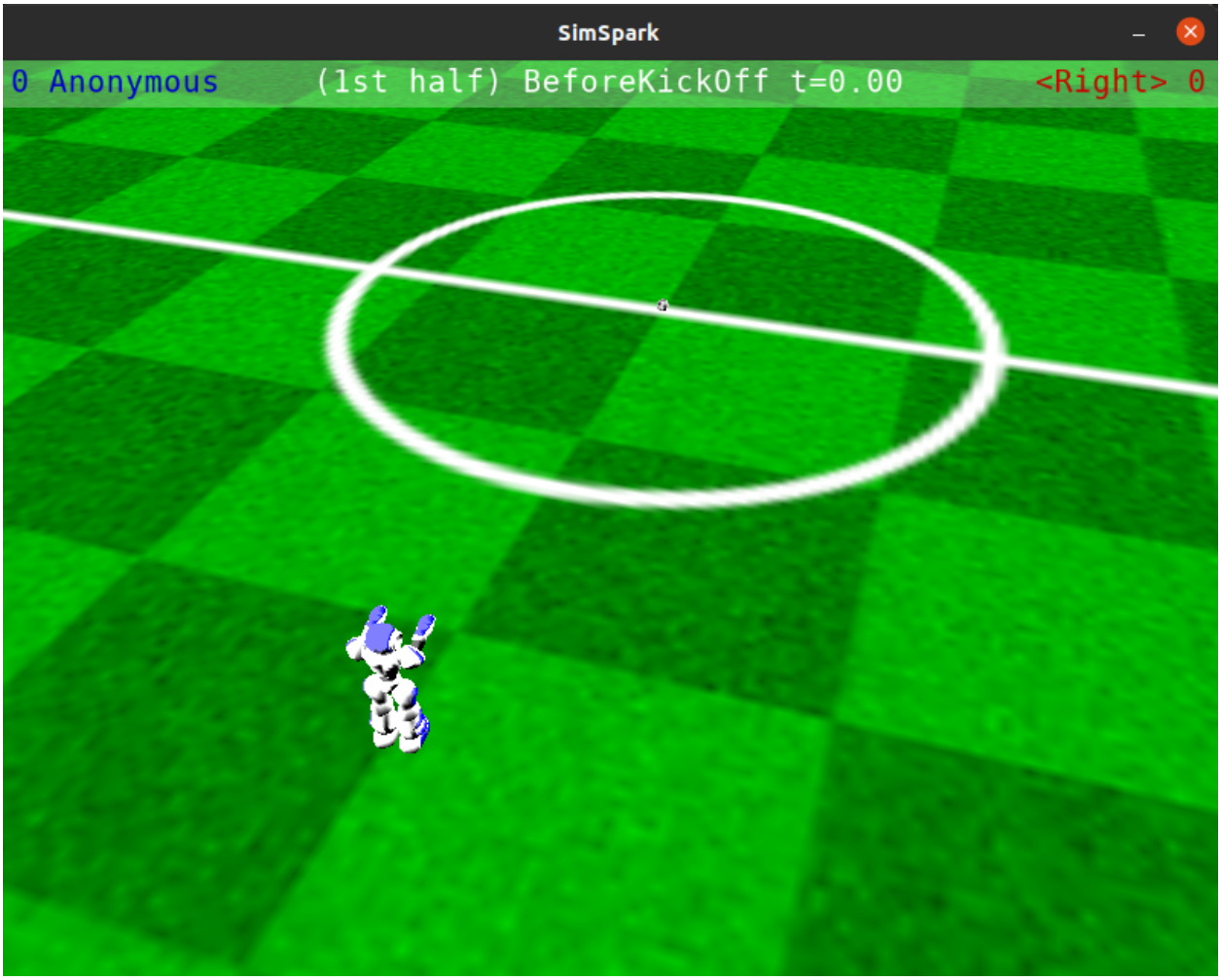
2.3 Beaming the Robot

The Beam Effector allows a player to position itself on the field before the start of each half. In this example, we will move the robot to four metres behind the centre circle, facing it. The coordinate of the robot after beamed will be (-4.0, 0.0, 0.0).

In a new terminal, run:

```
ros2 topic pub --once effectors/beam rcss3d_agent_msgs/msg/Beam "
x: -4.0
y: 0.0
rot: 0.0
"
```

In the simulator, you should see the robot has moved to the requested pose as below:



2.4 Moving the Robot's Joints

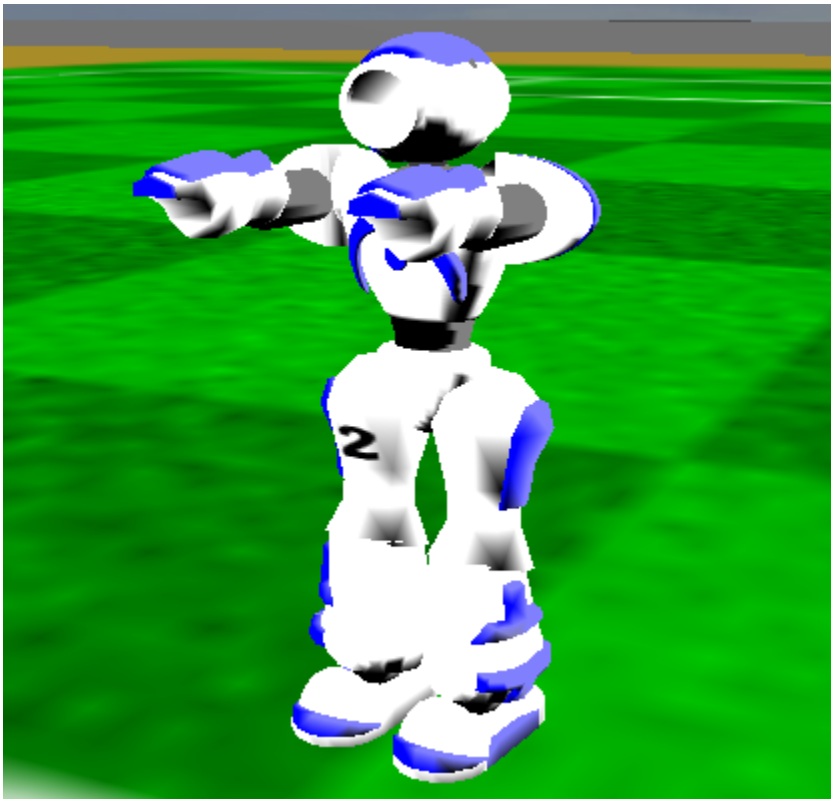
To send a joint command to the simulated robot, you must publish nao_command_msgs/msg/JointPositions data to the `effectors/joint_positions` topic:

Let's try and rotate the HeadYaw joint so that the robot faces 90 degrees to the left. To do so, we publish an array of joint angles, and specify 1.57 radians (90 degrees) for the HeadYaw joint.

In a new terminal, run:

```
ros2 topic pub --once effectors/joint_positions nao_command_msgs/msg/JointPositions '
↪{indexes:{0}, positions:{1.57}}'
```

In the simulation monitor, you should see the robot with its head twisted.



See also:

See joint_indexes to see which joint corresponds to each index of the float array published in the previous message.

Tip: Spend a bit of time playing around with the angles for each joint if you are not familiar with the NAO's joints!

2.5 Print out Published Topics

Let's try and print out the visual ball information received by our simulated robot. In a new terminal, run:

```
ros2 topic echo soccer_vision_3d/balls
```

The terminal will start echoing information the agent is publishing to the `ssoccer_vision_3d/balls` topic, like below:

```
header:
  stamp:
      sec: 0
      nanosec: 0
  frame_id: CameraTop_frame
  balls:
  - center:
      x: 14.210767712208556
      y: -9.207436569105516
      z: -0.4907249223563084
```

(continues on next page)

(continued from previous page)

```
  confidence:
      confidence: -1.0
---
```

Other topics the agent is publishing to are listed in *Topics*. By writing a node that subscribes to these topics, you can access this information in your own package.

2.6 Summary

That's it! You should by now know how to

- start up a simulated robot
- beam the robot
- send joint position commands
- access sensor information

CHAPTER

THREE

TOPICS

3.1 Published Topics

Warning: `vision/` topics were added after ROS 2 Humble, and aren't available in Humble and previous distros.

- **sensors/accelerometer** (nao_sensor_msgs/msg/Accelerometer)
- **sensors/angle** (nao_sensor_msgs/msg/Angle)
- **sensors/fsr** (nao_sensor_msgs/msg/FSR)
- **sensors/gyroscope** (nao_sensor_msgs/msg/Gyroscope)
- **sensors/joint_positions** (nao_sensor_msgs/msg/JointPositions)
- **soccer_vision_3d/balls** (soccer_vision_3d_msgs/msg/BallArray)
- **soccer_vision_3d/goalposts** (soccer_vision_3d_msgs/msg/GoalpostArray)
- **soccer_vision_3d/markings** (soccer_vision_3d_msgs/msg/MarkingArray)
- **soccer_vision_3d/robots** (soccer_vision_3d_msgs/msg/RobotArray)

3.2 Subscribed Topics

- **effectors/joint_positions** (nao_command_msgs/msg/JointPositions)

CHAPTER

FOUR

PARAMETERS

- **rcss3d/host** (*string*, default="127.0.0.1")

  Host IP Address that simulation server is running on

- **rcss3d/port** (*int*, default=3100)

  Port number that simulation server is communicating on

- **team** (*string*, default="Anonymous")

  Team name of robot, to be sent to simulation server

- **unum** (*int*, default=0)

  Player number of robot, to be sent to simulation server

CHAPTER

FIVE

DIFFERENCES WITH REAL ROBOT

The following data is available on a real Nao, but not available in the simulated robot:

- Sonar Sensors
- Touch Sensors
- Battery Status

The simulation server **does not provide camera images**. Instead, **simulated vision data** including balls, robots, and field lines are published. The simulated vision in the simulator does not report field features (Corners, Circle, T-Junctions, X-Junctions, etc.)