ROS 2 for RoboCup

Marcus M. Scheunemann and Sander G. van Dijk

University of Hertfordshire, AL10 9AB, UK

There has always been much motivation for sharing code and solutions among teams in the RoboCup community. Yet the transfer of code between teams was usually complicated due to a huge variety of used frameworks and their differences in processing sensory information. The RoboCup@Home league has tackled this by transitioning to ROS as a common framework. In contrast, other leagues, such as those using humanoid robots, are reluctant to use ROS, as in those leagues real-time processing and low-computational complexity is crucial. However, ROS 2 now offers built-in support for real-time processing and promises to be suitable for embedded systems and multi-robot systems. It also offers the possibility to compose a set of nodes needed to run a robot into a single process. This, as we will show, reduces communication overhead and allows to have one single binary, which is pertinent to competitions such as the 3D-Simulation League. Although ROS 2 has not yet been announced to be production ready, we started the process to develop ROS 2 packages for using it with humanoid robots (real and simulated). The strong support from large entities from the industry, such as Intel and Amazon, and the benchmarks shown here, indicate that ROS 2 is a promising candidate for a common framework.

Benefits over ROS 1

- Built-in support for real-time systems, as it sits on top of the Data Distribution Service (DDS) standard.
- Support for defining the 'Quality of Service' of topics. This allows one to make a range of trade-offs between strong reliability and 'best effort' policies, to deal with lossy communication.
- Nodes can be run in individual executables, or composed, using a variety of executors. In ROS 1, one has to maintain 'nodelet' versions of all nodes to make this possible.
- No need to run the ROS 1 `roscore` instance and maintain environment variables to make it and nodes reachable; with DDS, nodes discover each other through a network automatically.
- Communication between nodes can be strictly restricted by placing them in different 'domains'.

ROS 2 and RoboCup Contributions

- We have developed a range of general and RoboCup specific packages for ROS 2, including:
  - USB (V4L2) camera driver
  - Driver for ROBOTIS CM-730 sub-controller and Dynamixel motors
  - IMU fusion filter
  - Humanoid league Game Controller interface
  - 3D simulation league suite
- **Open sourced at:** `https://gitlab.com/boldhearts`

Real Humanoid Robot

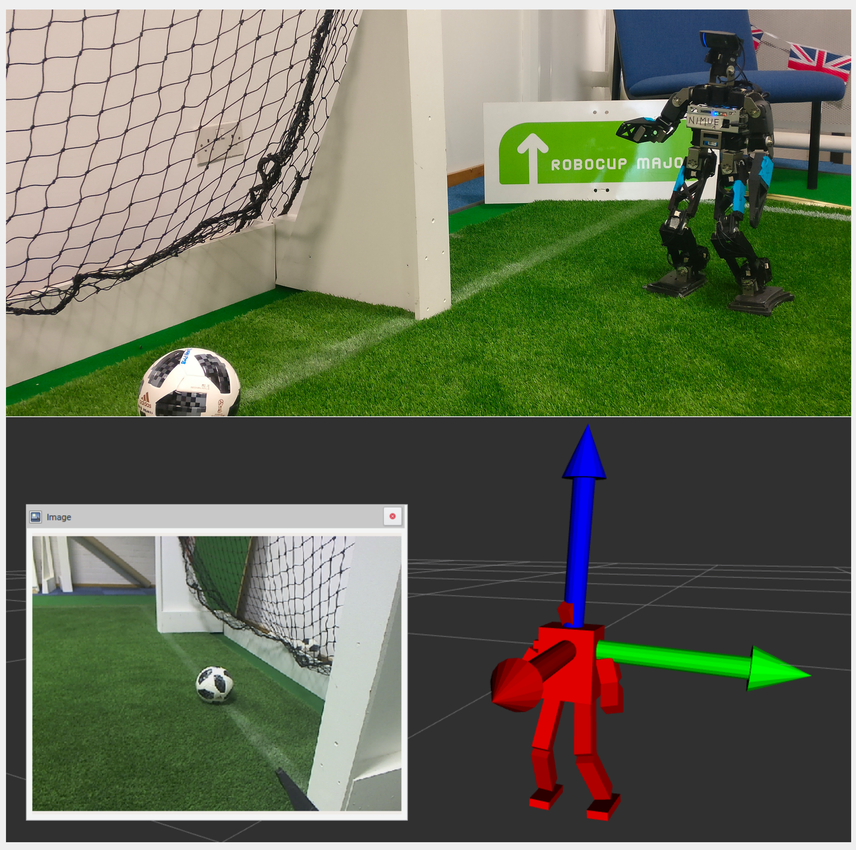

Figure: A scene with a robot looking at a ball (top). The lower image depicts a screenshot of RViz2. It shows the camera image feed retrieved with our USB camera driver (left). The CM-730 package publishes joint states, accelerometer and gyro information for building the robot model and compute its orientation with our IMU fusion package (both right). The robot model is built from a URDF model and standard ROS 2 joint state messages.

Simulated Humanoid Robot

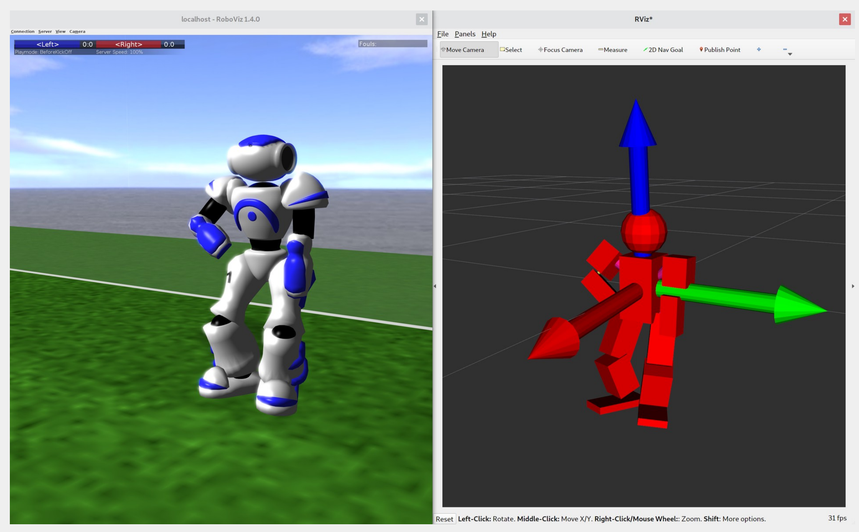

Figure: Depicted is a scene from RCSSServer3D used in the 3D Simulation League (left, using RoboViz). Our package translates the servo information into standard ROS messages and publishes the topic `/joint_state`. Also, the simulated gyroscope and accelerometer information are published. Our IMU fusion package subscribes to the messages and computes the robot's orientation, exactly as for the real robot above. The interface package allows for using ROS 2 within the context of the 3D Simulation League.

Benchmark Stand-Alone versus Composed Nodes

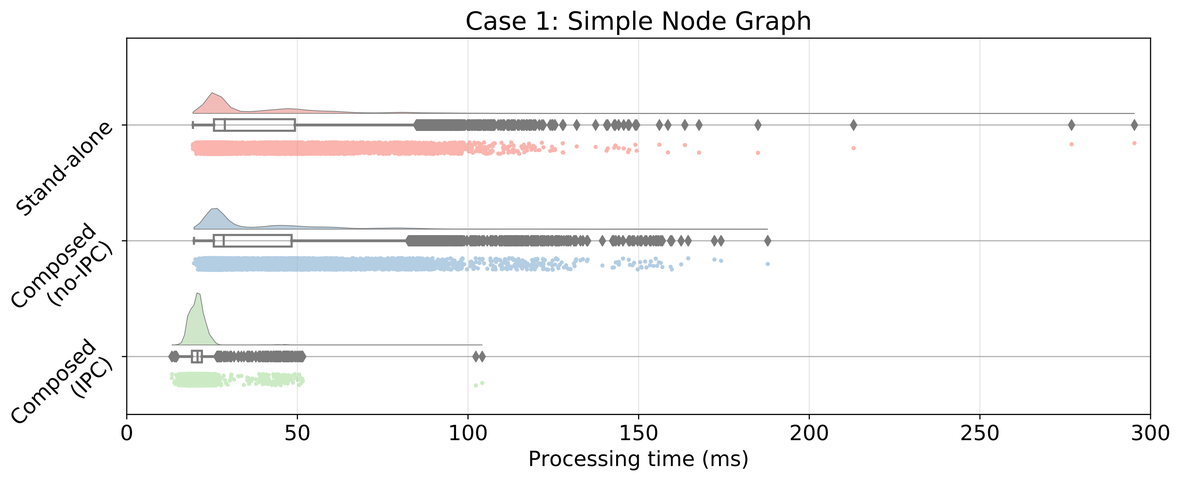

Figure: Distributions of time measured from image capture by camera node until end of processing a Sobel on the full image by processing node. Both nodes either run as stand-alone executables or run composed in a single executable, the latter with Intra-Process Communication (IPC) disabled and enabled. Each plot shows the density (top), a boxplot (middle), and individual data points (bottom). 10.000 samples are measured in each case.

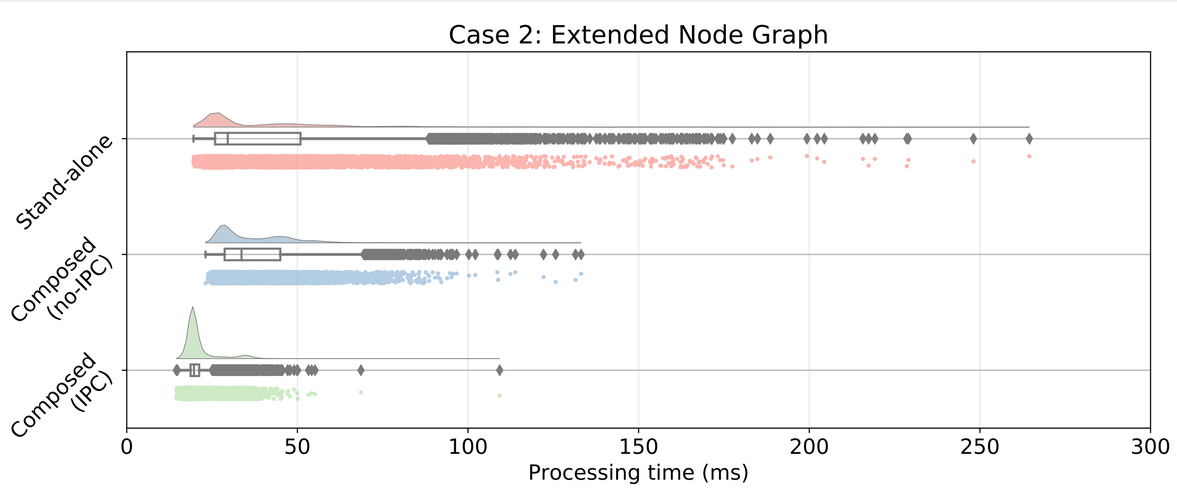

Figure: Image processing times as above, but with the robot system extended with other necessary nodes and topics like `/joint_states` and `/imu/data`, for a total of 7 nodes running at a time.

marcus@mms.ai / sgvandijk@gmail.com