2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

ROS4iOS : Native ROS Development on iOS Devices

Ronan Chauvin*, François Ferland*, Dominic Létourneau*, François Michaud*
*IntRoLab – Intelligent, Interactive, Integrated and Interdisciplinary Robotics Lab
3IT - Interdisciplinary Institute for Technological Innovation, Université de Sherbrooke, Sherbrooke (QC) Canada J1K 0A5
Email: {Ronan.Chauvin, Francois.Ferland, Dominic.Letourneau, Francois.Michaud}@USherbrooke.ca

I. Introduction

Smartphones and tablets are now part of our everyday lives, facilitating access to information through intuitive graphical interfaces. They can make great devices to work with robots [1] because they integrate a lot of sensors and reasonable computing power. Using ROS [2] on those devices would facilitate code reuse and integration with existing robotics applications and libraries.

Porting ROS applications to iOS is difficult because there is no native support for ROS on the iOS platform. Often, interaction with ROS is accomplished using a standard web-based approach (e.g., using `rosbridge`), with the web server hosted on the robot's computer, to ensure portability and compatibility with standard mobile web browsers. However, the web-based approach has its limitations and does not offer all the functionalities and computing capabilities a mobile device has to offer. Another possibility is to use a bridge application on the robot's side that can translate messages between ROS and another protocol. The programmer also needs to write an application on the mobile device using the same protocol, which somewhat duplicates the effort. One popular example is the use of the ROSOSC package and the Open Sound Control (OSC) protocol in the `rososc` package [3], used in conjunction with the TouchOSC iOS application to control the robot and monitor its status. Such development is time consuming and requires constant code maintenance since both protocols and messages types are evolving over time. The ROSpod project (http://ros.org/wiki/rospod) demonstrates a proof-of-concept of implementing a ROS port on iOS, but it did not provide a straightforward way to automatically build already existing ROS code for iOS. Futhermore, it has not been updated for recent ROS releases.

Therefore, we decided to concentrate our efforts on making a native port of ROS for iOS. Our port is not a re-implementation of the ROS environment, but a way to build standard ROS packages as iOS frameworks. The only requirement is to have a ROS Master already running on a separate computer. Our design is based on the following guidelines:

- Start from ROS sources hosted on `GitHub` and apply minimal patches for iOS compilation.
- Concentrate on the C++ portion of ROS' core libraries
- Automate the creation of iOS frameworks from ROS packages for use with Apple's Xcode development tool.
- Reuse all the communication protocols, messages and services available from ROS.
- Create complete ROS nodes running on the mobile device by reusing the same C++ code.
- Design an architecture to interface ROS data structures with iOS user interface elements.

The video attachment shows how the iOS frameworks are generated along with four use case scenarios demonstrating useful ROS4iOS functionalities and tools to create complete applications. These functionalities are:

- Display of maps, robot models and trajectories.
- Send/Receive a video stream using ROS image transport protocol and H.264 codec.
- Display of a point cloud from color and depth data.
- Send/Receive a sound stream.
- Remote control using joystick or accelerometer data.

The most CPU intensive scenario is the real time display of the point cloud, which requires the decoding of two compessed video feeds (H.264 and image depth compression) and displays OpenGL elements. The application uses 65% of the dual core CPU of an iPhone 5 (A6 chip). The procedure and documentation required to create the ROS4iOS frameworks are available at `https://github.com/introlab/ros_for_ios`.

II. Conclusion and future work

With ROS4iOS, the programmer can concentrate on the application instead of porting code from ROS or implementing bridging protocols and applications, which saves a considerable amount of time. Future work consists of developing different HRI scenarios using ROS4iOS.

Acknowledgments

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada.

References

[1] S. O. Adalgeirsson and C. Breazeal, "Mebot: A robotic platform for socially embodied telepresence," in *Proceedings of the 5th ACM/IEEE international conference on Human-robot interaction*, 2010, p. 15–22.
[2] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.
[3] D. Wessel and M. Wright, "Problems and prospects for intimate musical control of computers," *Computer Music Journal*, vol. 26, no. 3, p. 11–22, 2002.