Rapid Developing the Simulation and Control Systems for a Multifunctional Autonomous Agricultural Robot with ROS

Zhenyu Wang(✉), Liang Gong, Qianli Chen, Yanming Li, Chengliang Liu, and Yixiang Huang

Institute of Mechatronics and Logistic Equipment, School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai 200240, People's Republic of China
{silent_l80, gongliang_mi, chenqianli, ymli, chlliu, huang.yixiang}@sjtu.edu.cn

Abstract. Building customized control system for specific robot is generally acknowledged as the fundamental section of developing auto robots. To simplify the programming process and increase the reuse of codes, this research develops a general method of developing customized robot simulation and control system software with robot operating system (ROS). First, a 3D visualization model is created in URDF (unified robot description format), and is viewed in Rviz to achieve motion planning with MoveIt! software package. Second, the machine vision provided by camera driver package in ROS enables the use of tools for image process, 3D point cloud analysis to reconstruct the environment to achieve accurate target location. Third, the communication protocols provided by ROS like serial, Modbus support the communication system development. To examine the method, we designed a tomato harvesting dual-arm robot, and conducted farming experiment with it. This work demonstrates the advantages of ROS when applied in robot control system development, and offers a plain method of building such system with ROS.

Keywords: Robot Operating System (ROS) · Dual-arm multifunctional robot · Rapid system development · Autonomous agricultural robot · Rviz

1 Introduction

With the rapid development of modern agriculture of high efficiency, the vital position and the function of automation control technology has been widely acknowledged. Faced with the pressure of agricultural products output and market competition, agriculture tend to develop with higher efficiency and accuracy, combined with automated mechanical equipment [1]. In agricultural aspect, production lines equipped with machines have been widely applied in planting crops of large scale, for instance wheat, cotton, etc. At the same time, it's clear that there are many process of planting crops requiring to be operated with higher accuracy and flexibility, due to its complex environment and changeable conditions, like growing tomatoes. Within automated mechanical equipment, agricultural robots stand out for its high controllability and

N. Kubota et al. (Eds.): ICIRA 2016, Part I, LNAI 9834, pp. 26–39, 2016.
DOI: 10.1007/978-3-319-43506-0_3

flexible kinematic characteristics, which meet the requirements above perfectly. Taking these into consideration, improving accuracy and controllability of agricultural robots with excellent control systems is gaining more and more significance in modern automation researches [2].

Robot Operating System (ROS) is a collection of software frameworks for robot software development, providing operating system-like functionality on a heterogeneous computer cluster. ROS provides standard operating system services such as hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management. Running sets of ROS-based processes are represented in a graph architecture where processing takes place in nodes that may receive, post and multiplex sensor, control, state, planning, actuator and other messages [3].

Designed to increase the reuse of codes, ROS is completely open-sourced and compatible with multi programming languages. Over 2000 existing program packages are available in ROS freely.

The main characteristics of ROS include: Open sourced, multilingual support, library integration, plentiful tools kit, and point to point communication. The computation graph level is shown in Fig. 1.

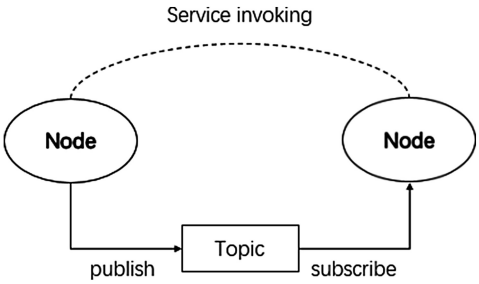

Fig. 1. The computation graph level of ROS

The computation graph reflects the connection way when the processes cooperate to process the statistics (point to point). Concepts concerned include node, service, topic, and messages, etc.

Nodes: A node is an executable that uses ROS to communicate with other nodes.

Messages: ROS data type used when subscribing or publishing to a topic.

Topics: Nodes can publish messages to a topic as well as subscribe to a topic to receive messages.

Master: Name service for ROS (i.e. helps nodes find each other)

Service: The method of communication between nodes, which allows nodes to send the request and answer.

Such point-to-point communication method is playing an important role in multi processes and multi hosts. When the multi hosts connect with different kinds of networks, there might be the risk of data transporting jam in central data server. While for point to point communication, there is no central data server, so it can avoid such problem to ensure the stability of multi processes and hosts.

This paper is organized as follows. Section 2 highlights the key ROS modules for developing an intelligent robot simulation and control systems. Section 3 describes the ROS deployment on a newly developed multifunctional agricultural robot. And conclusion is given in Sect. 4.

2 Customizing Robotics Modules in ROS

In general, there are four essential aspects for developing specific simulation and control systems for an intelligent robot, i.e. the operating system architectural model, the motion planning module, the machine vision module and built-in communication module.

2.1 Building Description Model Based on URDF

Robot visualization models enable the users to be informed of the current situation of the robot, decreasing the workload and the error rate. In ROS, the visualization is achieved with Rviz, and the general format of the robot description model is in URDF (unified robot description format). Based on XML, URDF is a language designed to describe the robot simulation model universally in ROS system, including the shape, size and colour, kinematic and dynamic characteristics of the model [4].

The basic way of building the visualization model in URDF is writing and compiling the URDF file. In URDF grammar, robot structure should be divided into links and joints. The connection relationship between parts is described by <parent> and <child>. To precisely describe the parts with parameters, ROS provides XACRO to allow users to use calculation macro in URDF. The common commands used to define the connection relationships are listed in Table 1.

Table 1. Common commands used in URDF

Commands	Grammar
Name the robot	robot name = "***"
Define the part	link name = "link***"/
Define the connection node	joint name = "joint***" type = "***"
Define the connection relationship	parent link = "link****"/ child link = "link****"/

When the structure of robots is relatively complex, the complexity of writing URDF file increases greatly. To ensure the accuracy of description, the often-used method is using 3D modelling software like Solidworks, Unigraphics NX, and transform the model into URDF by applying the plug-in like "solidworks to urdf". The suggested format of the 3D model is *.gae, while using stl format file is also acceptable in ROS as it contains the main grid statistics.

After writing the URDF file, the check_urdf tool can be used to check the grammar. If there exist no mistakes, this URDF file can be viewed in Rviz to visualize the robot model now.

When the URDF file needs to be applied in robots, users need to publish the robot conditions to tf, robot_state_publisher serves as the basis. The relevant parameters include the urdf xml robot description, and the joints information source with sensor_msgs/JointState format. The process of compiling URDF will generate the *. launch file, as the executive program in ROS.

In short, to realize the visualization, users need to write the urdf description file, and a node to publish and transform the information supported by robot_state_publisher.

2.2 Motion Planning Visualization Based on Rviz and Moveit

Rviz is the built-in visualization tool in ROS, providing the 3D simulation environment. With properly-set URDF file, using Rviz to visualize the robot model will be of no difficulty. When it comes to the kinematic analysis, the precise kinematic definitions needing can be acquired from the URDF file, and the relevant operation is supported by Moveit. The visualization environment provided by Rviz is shown in Fig. 2.

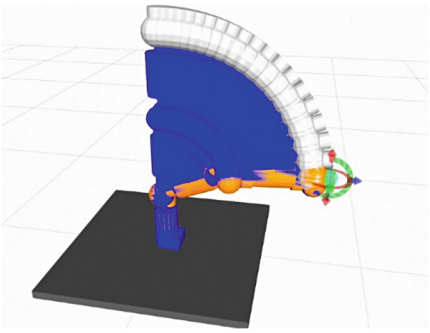

Fig. 2. The visualization environment provided by Rviz

Moveit is a universally-used integrated tool kit in ROS, as the core of motion control system, in charge of the calculation process of the positive and athwart kinematics of the robot's kinematic model. While the motion planning algorithm is based on the third-party library, OMPL (open motion plan library) [5].

To apply Moveit! in robot control system, the most convenient way is using the application assistant, which allows users to finish relevant configuration in steps. The main steps include importing the URDF file, setting the collision matrix, and adding the links and joints to the concerned motion planning group [6].

2.3 Implement Machine Vision with ROS

Machine vision is the technology and methods used to provide imaging-based automatic inspection and analysis for such applications as automatic inspection, process control, and robot guidance in industry. ROS integrates the driver kit OpenNI of Kinect, and can use OpenCV library to operate various image processes.

The common package used in ROS for image operating include camera_calibration, which serves as a calibration tool for the camera. The main algorithm it uses is the calibration method of chess board put forward by Zhengyou Zhang, which is also the general calibration method applied in OpenCV. The package image_view allows users to check the camera photos and give corresponding advice for the robot [7].

Here is the frequently-used usage:

```
  rosrun camera_calibration cameracalibrator.py --size 8x6 --sq
uare 0.108 image:=/my_camera/image camera:=/my_camera
```

Use an 8 × 6 chessboard with 108 mm squares to calibrate the camera.

```
  rosrun image_view stereo_view stereo:=<stereo namespace> imag
e:=<image topic identifier>
```

Apply the image_view package to look over the stereo photo acquired from the camera for further correction.

To achieve better target location effect, binocular vision has been the mainstream in machine vision aspect. With binocular cameras shooting at known positions, the relative position information of the target object can be acquired by analyzing the images of the binocular cameras. The basic binocular vision principle is shown in Fig. 3.

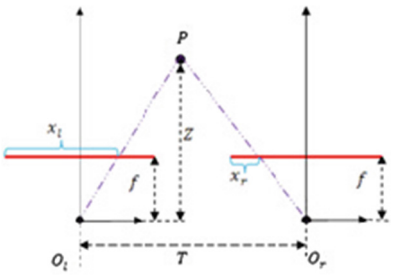

Fig. 3. Binocular vision principle

Suppose that we use two cameras to estimate the position of the target P. The two cameras have parallel optical axis, and the distance between the two cameras is T. The red thick lines stand for the image planes. Both focus of the cameras is f. According to the similar triangle principle, Z can be induced by:

$$\frac{T-(x_l-x_r)}{Z-f}=\frac{T}{Z}\Rightarrow Z=\frac{fT}{x_l-x_r}$$

ROS provides the relevant camera drivers packages, so users can select the corresponding branch to adapt need.

2.4 Devices Communication Principle

The communication within computers using ROS is generally based on TCP/IP protocol, which provides convenience in system building as the communication through internet nodes has been fully developed. With the master process running on the control computer, each node is able to interact with the other node in form of messages [8].

The mechanical arm motion statistics acquired from motion planning is transported with JointTrajectory message, which is a kind of track statistics in PVT (position, velocity, time) format. It stands for the position, the instantaneous velocity at the position, and the time taken to reach this position of each mechanical arm concerned. Through Modbus TCP Protocol, the PVT motion statistics is transported to multi-axis controller. Each axis finishes the planned motion according to the statistics after interpolation operation.

In telecommunications, RS-232 is a standard for serial communication transmission of data. It formally defines the signals connecting between a DTE (data terminal equipment) such as a computer terminal, and a DCE (data circuit-terminating equipment or data communication equipment), such as a modem. The RS-232 standard is commonly used in computer serial ports. In ROS, the package ROS-serial is a protocol for wrapping standard ROS serialized messages and multiplexing multiple topics and services over a character device such as a serial port or network socket. Classified by the different clients, ROS-serial provides various library aimed at Arduino, windows, Linux, etc. Usually, serial package is used to realize the communication between RS-232 serial and the device running Windows or Linux [9].

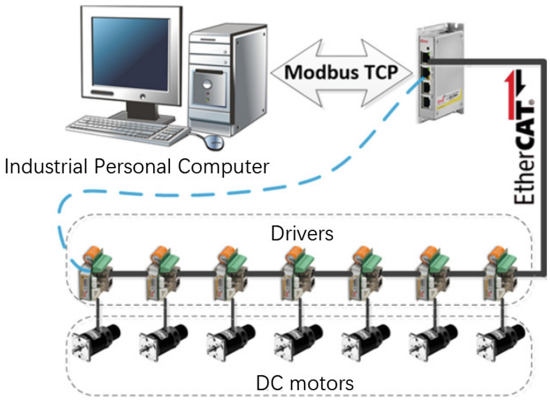

Fig. 4. The Modbus TCP communication base on EtherCAT bus

The Modbus package provides a wrapper from the Modbus TCP communication to standardized ROS messages. Programs of users use API library to run a Modbus server, in which there are the holding register and coil register. The upper machine acts as a client of Modbus, realizing communication and information interaction through reading the value of the register in Modbus server. The corresponding value leads the user's programs to realize motion control. The Modbus TCP communication is shown in Fig. 4.

3 Deployment Instance: Tomato Harvesting Dual-Arm Robot BUGABOO

It is a challenging task to develop an autonomous agricultural robot to fulfil multiple purposes due to the fact that the unstructured environment leads to difficulties for machine vision to identify targets and for intelligent manipulation to avoid obstacles. A humanoid agricultural robot, BUGABOO, is designed at Shanghai Jiao Tong University to perform various tasks such as plant disease monitoring, pesticide spraying and fruit harvesting. In this section the tomato harvesting task of the agricultural robot is selected as a symbolic case to show that the ROS facilitates a rapid development of simulation and control systems for BUGABOO.

3.1 The Mechanical System for BUGABOO

To finish the autonomous tomato harvesting task with accuracy and efficiency, this designed structure is as follows,

BUGABOO has two 3 DOF upper limbs mimicing the human arms, and a rotating platform serving as the waist. The serial arms with same structure are installed on the waist platform symmetrically. The 3 degrees of freedom include: the DOF of the lifting joint vertically, the DOF of the rotating joint of bigger arm, the DOF of the rotating joint of smaller arm. The waist-shape platform can rotate around the axis perpendicular with the ground to change the overall direction of robot. The single arm structure is like that of SCARA robot, in which the lift joint changes the position of the end in vertical direction, and the two rotating joints cooperate to change the end position in horizontal direction. Flange surface is installed at the end of the smaller arm; thus different end actuators can be installed to finish different tasks. The base is fixed on the automated trail car, which could move along the trail in field. When the robot recognizes and locates the ripe fruit, the dual-arm cooperate to harvest the fruit. The structure of the robot main body is shown in Fig. 5.

3.2 The General Control System Structure Design

As shown in Fig. 6, there are multiple nodes under ROS framework running on the monitoring computer and airborne computer at the same time. All the nodes together constitute the upper machine software part of harvest system. According to the function, the structure mainly include the 3D simulation environment based on ROS built-in

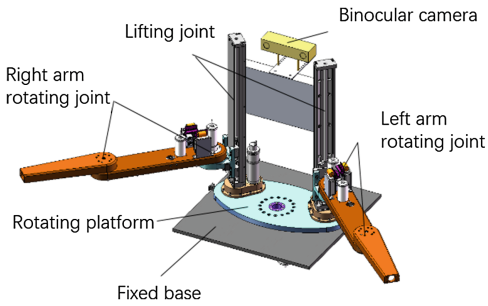

Fig. 5. Structure of the robot main body

visualization tool RViz, mechanical arm motion planning function library set based on MoveIt!, machine vision processing based on OpenCV open-source library, task level state machine programming based on SMACH library, interactive control interface based on the wxPython (encapsulated in Python sizers cross-platform GUI library), etc. Besides, there are also some function modules concerned with the bottom hardware, such as camera driver, and lower machine communication serial port and Modbus TCP procedures, etc. These function modules continuously produce messages when running, and at the same time also have demand for other information or services. ROS framework provides a good message-swapping and service invocation mechanism.

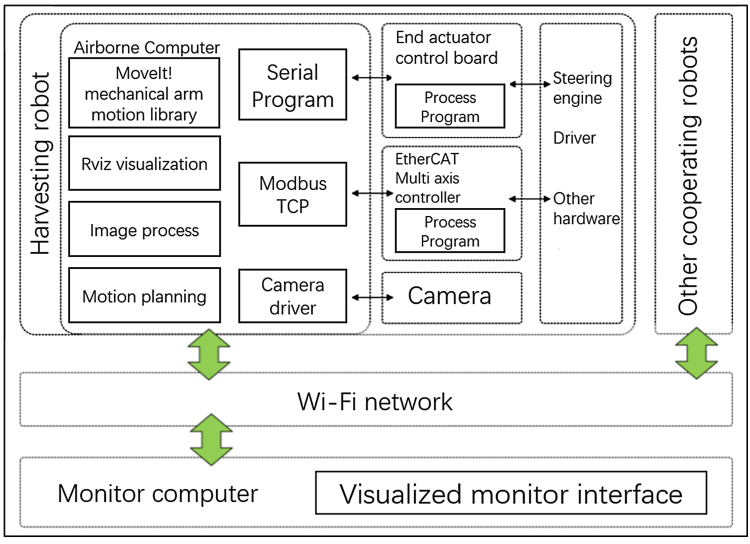

Fig. 6. Whole software system structure

3.3 Software Running Environment

Running ROS is available in Linux, Mac OS X, Android and Microsoft Windows, but users generally choose Ubuntu Linux, because this is the official recommendation to support the best operating system, and the use of Ubuntu is completely free of charge. The existing ROS versions are quite various. Considering the running stability, we choose the first version with 5 years support, indigo running on Ubuntu14.04 as the ROS version.

3.4 Modelling URDF with *.stl File

To ensure the accuracy of the model, we choose to build the 3D model with professional modelling software, Solidworks to get the required *.stl file, and import it into the URDF file as meshes.

With the URDF file finished, we use the tool in Rviz check_urdf to check if there is any grammar mistake. The corresponding output is shown as below:

```
robot name is: lr1_robot
---------- Successfully Parsed XML ------
root Link: support_link has 1 child(ren)
    child(1):  waist_link
        child(1):  bumblebee2_link
            child(1):  left_camera_frame
            child(2):  right_camera_frame
        child(2):  left_lift_link
            child(1):  left_upperarm_link
                child(1):  left_forearm_link
                    child(1):  left_arm_end_frame
        child(3):  right_lift_link
            child(1):  right_upperarm_link
                child(1):  right_forearm_link
                    child(1):  sawwrist_link
                        child(1):  saw_link
                          child(2):  right_arm_end_frame
```

To view the model more intuitively, we can use the tool urdf_to_graphiz to show the tree structure of the robot model, as Fig. 7 shows.

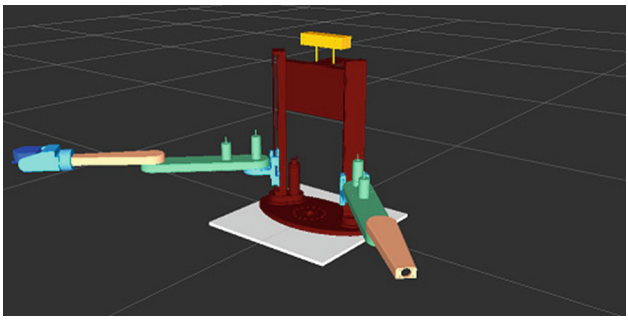

Fig. 7. The visualization model of the robot

3.5 Visualization and Motion Planning

With Rviz and MoveIt!, we can easily get the visualization model. Given the start point and end point, OMPL library support to finish the motion planning, "lr1" being the name of the robot. The simulation interface is shown in Fig. 8.

```
roslaunch lr1_move demo.launch
roslaunch lr1_description display.launch
roslaunch lr1_robot_moveit_config move_group.launch
```

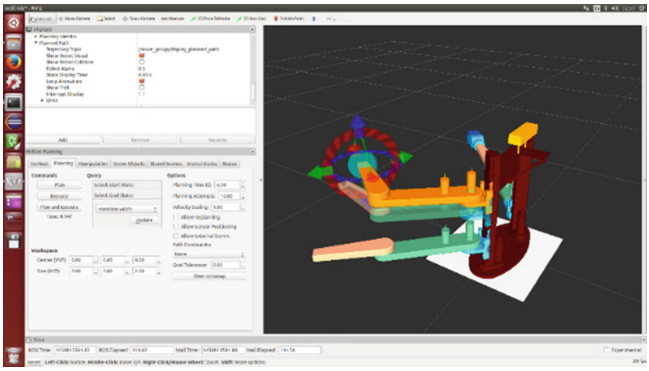

Fig. 8. MoveIt! interface in RViz and the arm trajectory simulation

3.6 Machine Vision

In fruits harvest mission, the relative position and posture of fruit relative to the robot are needed to control the end actuators to reach the ideal position operating harvest. Binocular stereo vision is similar to human visual system, and enjoys good accuracy and efficiency. Here we choose the binocular camera Bumblebee2.

There exists the matched software package for Bumblebee2 in ROS, thus the configuration of the camera is much easier. Project uses Bumblebee2 camera on Ubuntu Linux, uses libdc1394 library to control camera and capture the images, and then use the Triclops library to correct image and complete the image matching and depth calculation. Program runs as a ROS node, and publish the image, depth point cloud information as ROS topic, for the subscription of other ROS node. The basic logic is shown in Fig. 9.

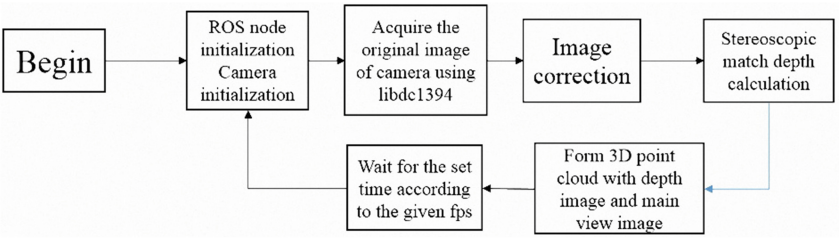

Fig. 9. The general procedure of stereo vision

3.7 Field Experimentation

To examine the working effect of the harvest dual-arm robot, we conducted a series of experiments in farming base. We took the robot to operate the independent harvesting process, realized the full autonomous fruit and vegetable harvesting operations by agricultural robot. The working situation is shown in Figs. 10 and 11.

Fig. 10. The Operating scenario of BUGABOO

Fig. 11. The operating details driven by ROS

Field experimentations demonstrate that the tomato harvesting dual-arm robot BUGABOO works with excellent stability as the data communication is based on TCP/IP protocol. Decent location accuracy is ensured by the binocular stereo vision. This experiment proved the advantages of ROS when applied in autonomous robot control system, which provides good reference for further improvement of the control system (Table 2).

Table 2. General index of BUGABOO

Parameters	Values
Weight	40 kg
Total power	2.2 KW
Work breadth	350 mm × 350 mm × 580 mm
Total degree of freedom	12
Average harvesting efficiency	45 s/each
Method of target location	Binocular stereo vision

3.8 Compare with MFC-Based Control System

At the beginning of program, we tried to build the control system on Windows platform, using Visual Studio development tool to write the C++ program and user interface.

As shown in Fig. 12, the interface can be divided into four part, including the photo view part, the 3-D simulation part, and two parameter-control parts. In the developing process, we found that there were many disadvantages compared with ROS:

- The program frame with single project is hard to develop when several programmers cooperate. As a single executive file is formed with all the sub-programs, debugging process takes a lot of time. While in ROS frame, the function module can be divided completely in different projects (nodes), which increases the developing efficiency.

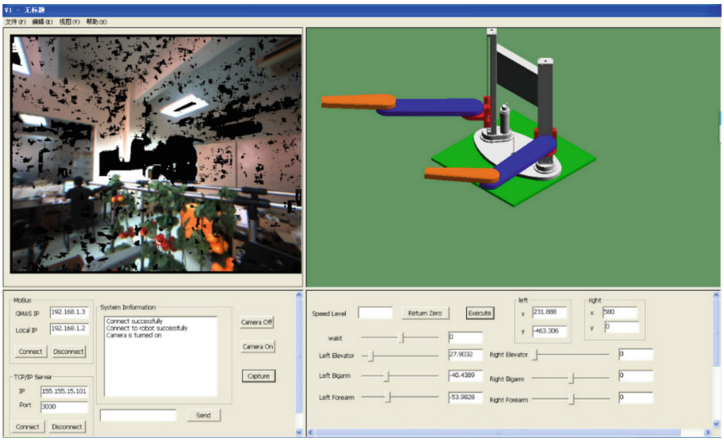

Fig. 12. Early control interface in Windows

- As the algorithm of the robot researching is changing constantly, the codes from Internet require to be edited greatly to adapt to the existing program frame, which is not worthy for selecting algorithms at the beginning step. While ROS has open-sourced internet community, in which there are many excellent codes and algorithm, and many algorithm libraries provide support to ROS.
- Multi-process communication is relatively messy in Windows platform. ROS provides the practical communication standard by each node, which could avoid the conflict due to dependencies and synchronous conditions.

4 Conclusion

In summary, this paper describes a general method of building robot control system software applying ROS, mainly including modelling with URDF, visualization with Rviz, motion planning with MoveIt!, and vision with OpenCV library, communication with serial and Modbus.

To illustrate the method more concretely, the paper takes the tomato harvesting dual-arm robot running ROS as example. Experiments prove that the robot control system built with ROS enjoys convenience of use and good stability.

In the research process, various kinds of problems arise. Further work is needed to improve the performance of the control system developed by ROS. Such as developing a performance evaluation system based on the data from field experiments to measure the reposition precision of the robot. And try to apply the control system developed by ROS in complex mechanical systems to make use of the advantages in complex communication of ROS module program.

With robot technology developing, the application of ROS in robot control system is expected to be wider and more efficient.

Acknowledgements. This research is founded by MOST of China under Grant No. 2014 BAD08B01 and No. 2015BAF13B02, and partially supported by the National High Technology Research and Development Program of China under Grant No. 2013AA102307.

References

1. Bac, C.W., Henten, E.J., Hemming, J., et al.: Harvesting robots for high-value crops. In: State of the Art Review and Challenges Ahead, pp. 888–911 (2014)
2. Wang, Y.H., Lee, K., Cui, S.X., Risch, E.: Research on agricultural robot and applications. In: Southern Plains Agricultural Research Center, College Station (2014)
3. Quigley, M., Conley, K., Gerkey, B., et al.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
4. Cao, Z.W., Ping, X.L., Chen, S.L., Jiang, Y.: Research on method of developing robot model based on ROS (2015)
5. Yousuf, A., Lehman, W., Mustafa: Introducing kinematics with robot operating system (ROS). In: ASEE Annual Conference and Exposition (2015)
6. Chitta, S., Sucan, I., Cousins, S.: Moveit![ROS topics]. IEEE Robot. Autom. Mag. **19**(1), 18–19 (2012)
7. Bradski, G., Kaehler, A.: Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media Inc., Sebastopol (2008)
8. Hoske, M.T.: ROS Industrial aims to open, unify advanced robotic programming. Control Eng. **60**(2), 20 (2013)
9. https://en.wikipedia.org/wiki/RS-232