JIRAE, Vol. 4, No. 1, April 2019, 1-5
e-ISSN 2407-7259
DOI: 10.9744/jirae.4.1.1-5

Towards a Low-Cost Solution to Learn Robot Manipulator Control

Handy Wicaksono[1,a], Handry Khoswanto[2,b], Petrus Santoso[3,c]
[1,2,3] Petra Christian University, Jl. Siwalankerto 121-131, Surabaya 60236, Indonesia
[a]handy@petra.ac.id, [b]handry@petra.ac.id, [c]petrus@petra.ac.id

Abstract. Learn a robot manipulator in university requires a physical robot so students can learn to program it straight away. However, it is expensive and not affordable by most universities in Indonesia. We turn to a ROS-based solution for an open source, free and advanced robotic software system. Using ROS and Gazebo simulator, we develop a low-cost solution where students can learn an advanced robot manipulator such as Baxter. We divide our labwork into five sections: controlling the arm w.r.t. joints position, controlling the arm based on its inverse kinematics, object detection by a camera attached in robot's wrist, executing a simple plan: pick and place task, and enabling a robot to perform a tool-use task. By gradually increasing the complexities of the labwork material we expect students get a better understanding in controlling a robot manipulator. We provide an affordable solution for students to learn robot manipulator control.

Keywords: Educational robot, robot manipulator, robot software architecture.

1. Introduction

Electrical Engineering (EE) Department at Petra Christian University (Indonesia) provides a "Robotika" course so students can learn about kinematics, dynamics, and control of a robot manipulator. Most portions of the course are spent to learn the mathematical aspects of a robot by using MATLAB to assist the computation. As a final project, students are asked to make a simple robot arm and control it using techniques that have been learned before. This kind of final project is chosen because our laboratory has not provided an accurate robot manipulator yet, because it is very expensive. The absence of an accurate manipulator prevents students to gain more complete and advance understanding of the lecture.

To avoid the above problem, many educators turn to a robot simulator to replace a physical robot, while the open-source software is used instead of the dedicated one. There are some software frameworks for robotics (e.g. URBI [1], OROCOS [2], or Microsoft Robotics Studio [3]), however, none of them become standard *de facto*. Recently Robot Operating System (ROS) is introduced by Standford researchers [4] and gains a lot of popularities and adoptions among robotics researchers.

ROS main advantage is it encourages researchers to reuse the code (which are shared openly by other researchers) so they can focus to develop a specific part of robot software instead of keep "reinventing the wheel" by programming all components of the robot. ROS also supports integration of heterogeneous components of robots. In conjunction with ROS, we can use a physics simulator such as Gazebo, a realistic simulator environment that enables programmers to test their robot in simulation before they deploy it in a real robot [5].

Considering the popularity and the flexibility of ROS framework, we propose new learning modules for "Robotics" course based on ROS and Gazebo to replace the old ones. This will benefit students as they will have the experience to deal with a complex robot since the beginning of the course. This is important as the best approach to learn robotics is "learning by doing". On the other hand, the school is also gain an advantage because it does not have to pay anything for these open-source software.

Some researchers report their work on the implementation of a robot system in education. Ruzzenente et al. [6] review several robotic kits that are suitable for tertiary education based on four criteria: modularity, reusability, versatility and affordability. They choose to use a LEGO NXT Mindstorm kit to develop a robot manipulator and use integrated Matlab/ROS to control the robot. Honig et al. [7] propose to use the integration of V-REP simulator and iRobot Create 2.

We choose to use Baxter robot as it is a ROS-ready robot with 7 degrees of freedom (DoF) and a lot of built-in sensors (force sensors, ultrasonic sensors, a vision sensor, etc.). It has two arms, which is useful if the students want to learn about advance robotics applications in the future. Rethink Robotics provides API in Python that is useful to access Baxter's motors and sensors in a ROS environment. It also provides a Baxter simulator in a Gazebo environment.

The simulation by using ROS and Gazebo is beneficial as the programmer can use a relatively similar code in a real or simulated robot. If a robot's simulation model is accurate, then the simulation result is reliable and can be used to minimize real world experiment. Based on that premise, in our previous research, we enable a Baxter robot to learn how to use a tool in a simulated and real environment [8] (see Figure 1. for illustration).

The recent robot application involves an artificial intelligence (AI) technique to make a robot smarter. ROS enables researchers to incorporate an AI method in a robot system relatively easily. ROS is also used by computer science educators when they include robotics

in their classes [9]. Students in our department (Electrical Engineering) may also benefit to learn these interesting AI techniques.

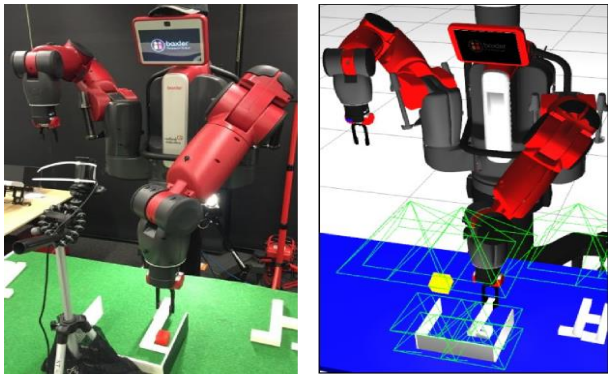

Figure 1. A physical and a simulated Baxter robot [8]

2. Architecture

We use an architecture that ensures the flexibility to use low-level behaviors and high-level actions. To support that, we develop a robot software architecture that has three layers: deliberative, translation, and reactive.

The reactive layer handles low level sensing and primitive actions in a robot. It is written in Python because Baxter has an easy-to-use Baxter API in Python. The deliberative layer might be written using Prolog, a logic programming approach that enables a robot to do symbolic planning, reasoning and learning. The translation layer bridges the gap between the deliberative and reactive layers. It is written in C++ as it has a library to access SWI Prolog. We describe this in detail in our research paper [10].

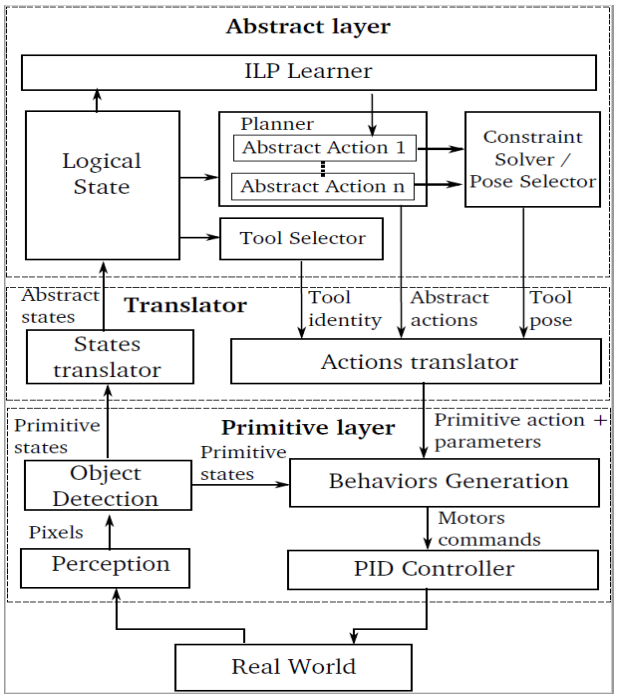

Figure 2. Robot architecture [10]

3. Learning Modules Development

We aim to use a Baxter robot in simulation to learn the concepts of a robot manipulator. Here are several main points that we want to emphasize in every learning module:

1. Control the arm movement with regards to their joint position and joint velocity
2. Control the arm movement by using Inverse Kinematics solver
3. Execute a sequence of actions: pick and place task
4. Object detection using a camera
5. Combining object detection and action: visual servoing pick and place task

We will combine the robot applications which are provided by Rethink Robotics and Active Robots. We also develop several applications by ourselves. We assume the familiarity of ROS and Gazebo environment for students who join our class. Each learning module will be explained in detail below.

3.1 Control the arm movement with regards to their joint position and joint velocity

The aims of this first module are twofold. The first one introduces students with Baxter simulator and its environment. Students need to understand existing sensors and actuators in Baxter and how to access them using Baxter API in Python.

The second aim is helping students to control the arm by giving each joint: a target joint position and/or a target joint velocity. An example program is provided by Rethink Robotics to change an arm's joint position by using a keyboard (Joint Position Keyboard Example). Baxter's joints are shown in Figure 3. Demonstrating this action and observing the code could help the students to learn about the joint position.

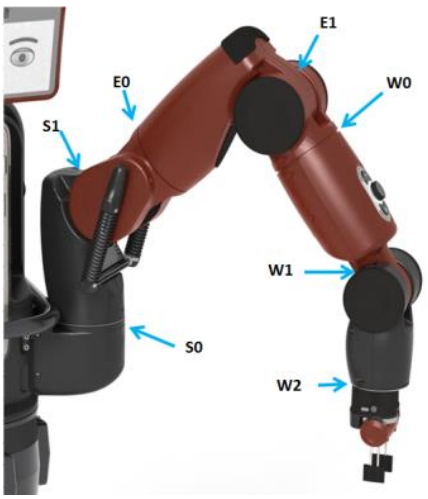

Figure 3. Baxter's joints

Rethink Robotics also provides Joint Velocity Wobbler Example, where Baxter arms perform the wobbly movement (see Figure 4. for the illustration).

The students learn to assign a cosine function to the joint velocity for each joint:

$$x = a * \cos(2 * pi * f * t)$$

Where *a* refers to amplitude, *f* refers to frequency and *t* refers to time.

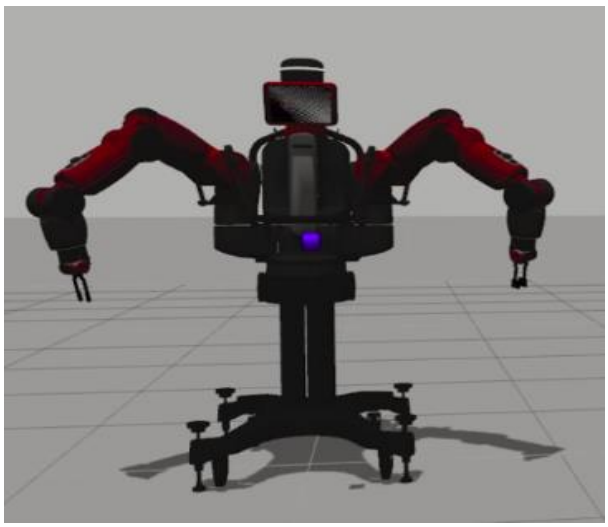

Figure 4. Baxter performs the wobbly movement.

In this module, students only need to open the Baxter in empty Gazebo world because there is no interaction between the robot and other objects.

3.2 Control the arm movement by using Inverse Kinematics solver

Controlling a manipulator robot by changing their joints positions and velocities are not too useful and intuitive for humans, who usually ask the robot to move to a particular target coordinate. This can be achieved by using Inverse Kinematics (IK) method to control a robot. Another code example from Rethink Robotics (IK Service Example) is useful to show students how to control an arm by giving its target coordinate (robot's position and orientation).

Another useful and popular option is by using KDL library to solve the IK equation. If students using this library, they should start by determining the kinematics structure of a manipulator, for example by determining its DH (Denavit-Hartenberg) parameters. An alternative option in KDL is by using the kinematics chain method as shown in Figure 5.

As in the first module, students only need to open the Baxter in empty Gazebo world here because they only need to observe the arm movement itself.

3.3 Execute a sequence of actions: pick and place task

After learning to control to move an arm to a single target location, students can proceed to perform a series of actions in a pick and place task. This sequence of actions can be considered as a simple plan. Task planning can bw done by just giving a predefined sequence to the robot. In this case, Baxter needs to do a sequence of these planned actions:

- Move to object location (based on its coordinate)
- Grip the object
- Move the object to a target location
- Ungrip the object

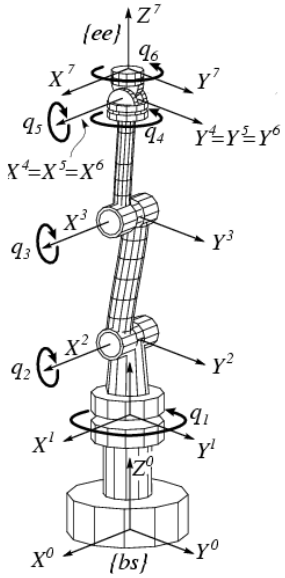

Figure 5. An example of a manipulator's kinematics chain[1]

To perform this module, Baxter robot, a table and a cube must be provided in Gazebo simulation. Rethink Robotics provides an example to perform this simple pick and place task (see Figure 6 for the detail).

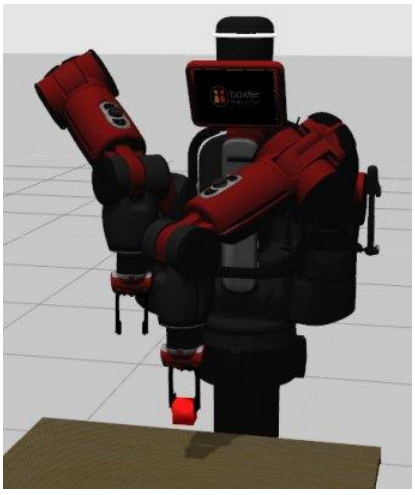

Figure 6. Baxter robot performs pick and place task

3.4 Object detection using a camera

Baxter is equipped with several sensors: vision sensor, force sensor, torque sensor, sonar, etc. In the third

[1] Orocos.org

module, the students begin to use a sensor (a camera) to observe a robot's environment. Students can make objects (or using an available object template) in Gazebo simulator. In this module, a table and objects basic-shape should be provided.

Baxter's has a built-in RGB camera in each of its grippers. It can detect objects on a table. Not just the internal camera, the external camera can be added to enable a robot to have a better view. Students can learn object detection techniques, such as Canny edge method and contour finding. OpenCV library can be used for this purpose.

Initially, a very simple object like a cube can be used. The RGB camera sees the object as a two-dimensional square or rectangle (depends on its view angle). By applying those two techniques above, the cube can be detected (see Figure 7).

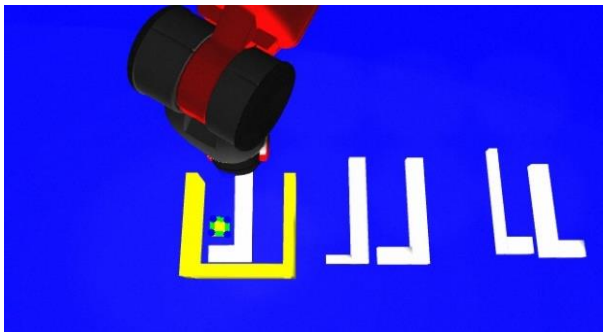

Figure 7. Cube detection by an external web camera

A more complex object, such as a hook-like object, can be used in the next stage. A hook is a composed object consists of a handle and one or more hooks. When detected by an RGB camera, those parts are appeared as several rectangles. By detecting those rectangles, a particular hook can be found. In our system, we provide five different shapes of hooks located on a table (see Figure 8 for the detail).

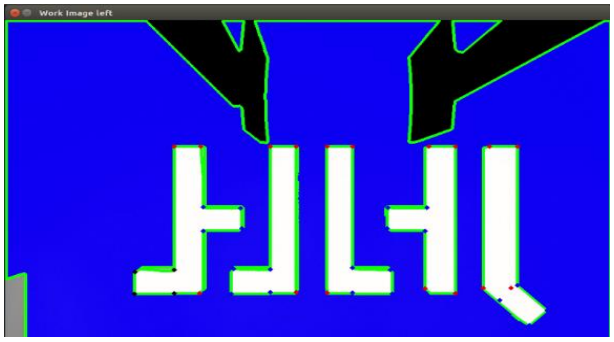

Figure 8. Hook detection by Baxter's gripper camera

3.5 Combining object detection and action: using a hook-like tool to pull a cube out of a tube

In this last module, we combine action execution (which students have learned in section 3.2 and 3.3) and object detection (appeared in section 3.4). As in section 3.3, we make a simple plan, however, actions in our plan are performed based on the sensor reading. Our refined plan is as follows:

- Find the target object (based on its 2D appearance)
- Grip the object
- Move the object to a target location (could be based on the 2D appearance of an "arena")
- Apply the object on another object
- Move the object to a home location
- Ungrip the object

The main difference between this module and previous module in section 3.3 is the utilization of cameras to detect the location of objects.

Practically, we modify the application developed by Active Robots. Here, a robot must detect all objects on the scene, pick the correct object, and apply it to a particular target object. To increase the accuracy of Baxter arm movement (which is not so great compared to a conventional industrial robot), a visual servoing technique is used here. This technique enables the robot to use visual information from the gripper camera to move its arm accurately. We use this technique to pick the cube in the early stage.

Figure 9 shows the complete experimental scene which consists of five hook-like tools that have different shapes, a tube, and a cube. We utilize an external webcam (appeared as yellow object) to equip the robot with the complete view of its experimental scene.

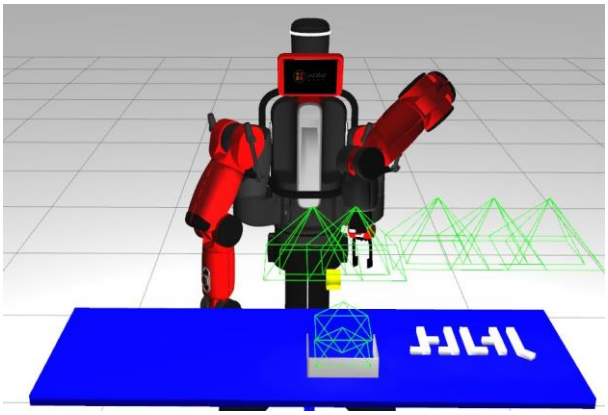

Figure 9. The experimental scene of Baxter performing a tool-use task

3.6 Possible extensions

Apart from the previous five modules, further development can be done by adding the other two modules to learn about: planning and machine learning. Those are advanced topics because the students will learn about artificial intelligence which is applied in robotics. The abstract layer (see our architecture in Figure 2) must be used here, as students need to learn about symbolic planning or learning written in SWI Prolog.

Having a symbolic task plan means that we do not have to predefine a plan such as a module in section 3.5. Having complete states and actions, problem solving can be performed and the plan is produced automatically. A simple symbolic planner such as means-ends planner or STRIPS planner can be used here.

Another extension that can be done is by adding a machine learning algorithm (such as inductive logic programming) in the abstract layer. If a robot is not given a complete knowledge, the results of tool-use experiments can be fed to the machine learning algorithm, so the robot can learn incrementally which tool is correct and how to use it in a way that guarantees a successful tool-use task. This includes learning the structure and the pose of a useful tool. We explain this approach in our paper [6, 8], however, we have not developed it as a learning module yet.

4. Conclusion

In this paper, we propose to develop learning modules for students to learn a manipulator robot without having a physical robot that is not affordable for many universities in Indonesia. Open source projects such as ROS, as robotics middleware, and Gazebo, as a simulation environment, enable us to achieve our goal. We explain five modules that have gradual complexities in relation to robot action, perception, and planning. We suggest another two modules about symbolic planning and machine learning to introduce AI applications in robotics.

These learning modules still in the development stage, so they have not been used by the students yet, so we can not conclude anything from the students' perspective. In the future, we will conduct a survey to evaluate the effectiveness of these modules to improve the students' understanding of the robot manipulator.

Acknowledgment

This work is supported by a research grant (Hibah Penelitian Terapan Unggulan Perguruan Tinggi, contract number 002/SP2H/LT/K7/KM/2017) from The Directorate General of Resources for Science, Technology and Higher Education (DG-RSTHE), Ministry of Research, Technology, and Higher Education of the Republic of Indonesia.

References

1. Baillie, J. C, URBI: Towards a universal robotic low-level programming language, Proceedings of 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, Aug 2005, pp. 820-825.
2. Bruyninckx, H., Open robot control software: the OROCOS project, Proceedings 2001 ICRA. IEEE international conference on robotics and automation, May 2001, pp. 2523-2528.
3. Jackson, J., Microsoft Robotics Studio: A technical introduction, IEEE robotics & automation magazine, 14(4), 2007, pp.82-87.
4. Quigley, M., et al: ROS: an open-source Robot Operating System, ICRA workshop on open source software, 2009.
5. Koenig, N. P., and Howard A.: Design and use paradigms for Gazebo, an open-source multi-robot simulator, Proceedings of IROS, 2004.
6. Ruzzenente, M., Koo, M., Nielsen, K., Grespan, L. and Fiorini, P.: A review of robotics kits for tertiary education, in Proceedings of International Workshop Teaching Robotics Teaching with Robotics: Integrating Robotics in School Curriculum, April 2012, pp. 153-162.
7. Hönig, W., Tavakoli, A. and Ayanian, N.: Seamless robot simulation integration for education: a case study, in Workshop on the role of simulation in robot programming at SIMPAR, 2016.
8. Wicaksono, H., and Sammut C.: Relational tool use learning by a robot in a real and simulated world, Proceedings of ACRA, 2016.
9. Michieletto, S., Ghidoni, S., Pagello, E., Moro, M. and Menegatti, E., Why teach robotics using ROS? Journal of Automation Mobile Robotics and Intelligent Systems, 8, 2014.
10. Wicaksono, H. and Sammut, C., Tool use learning for a real robot. International Journal of Electrical and Computer Engineering, 8(2), 2018.