ROS-based Control of a Manipulator Arm for Balancing a Ball on a Plate

Mr. Khasim Ali Khan
Dr. Ji-Chul Ryu, Northern Illinois University

Dr. Ji-Chul Ryu received the B.S. and M.S. degrees in mechanical engineering from Korea Advanced Institute of Science and Technology (KAIST) and the Ph.D. degree in mechanical engineering from the University of Delaware in 2009. From 1999 to 2004, he was a Research Engineer with several companies, including Samsung, where he developed various types of automated robotic machines. He worked as a Postdoctoral Fellow with the Neuroscience and Robotics Laboratory, Northwestern University prior to joining the Department of Mechanical Engineering at Northern Illinois University as Assistant Professor in 2013. His research interests include integrated planning and control of autonomous robotic systems, its application to mobility assistive robots, dynamic robotic manipulations with applications to manufacturing/industrial processes, and robotic machine/factory inspection systems.

ROS-based control of a manipulator arm for balancing a ball on a plate.

Abstract

Automation and robotics are the growing phenomena replacing human labor in the industries. The idea of robots replacing humans is positively influencing the business thereby increasing its scope of research. This paper discusses the development of an experimental platform controlled by a robotic arm through Robot Operating System (ROS). ROS is an open source platform over an existing operating system providing advanced capabilities for various types of robots from an operating system to low-level control.

We aim in this work to control a 7-DOF manipulator arm (Robai Cyton Gamma 300) through ROS to accomplish the task of balancing a ball on a plate-type end effector. In order to perform feedback control of the balancing task, the ball is designed to be tracked using a camera (Sony PlayStation Eye) through a tracking algorithm written in C++ using OpenCV libraries. The joint actuators of the robot are servo motors (Dynamixel) and these motors are directly controlled through a low-level control algorithm. To simplify the control, the system is modeled such that the plate has two-axis linearized motion. This system along with the approaches proposed in this work has potential to be used as a demonstration testbed for students to learn ROS with control theories in robotics.

1. Introduction

Robotics has been growing past the traditional engineering techniques emerging as a newer field of modern technology that requires various knowledge from mechanical and electrical engineering and computer science. When commercial robots are used, the approaches to these robots are provided by their manufacturers in the form of GUI (Graphical User Interface) or APIs (Application Program Interfaces) are provided for further user-customized functionality.

The GUI provided by the manufacturer often may not allow users to fully exploit the abilities of the robot and dealing with the APIs may be unwieldly. To overcome these inadequacies, Middleware packages have been developed [1], [2] and one such package is Robot Operating System (ROS). ROS is an open source middleware package that works between multiple platforms and performs OS-like functionality. Quigley et. al. summarize the goals of ROS as "Peer to Peer, Tools-based, Multi Lingual, Thin, Free and Open-sources" in [3]. Allgeuer et. al. describe ROS community to be a popular choice in the robotic research due to its flexibility, portability, and modularity while implementing it to an humanoid robot providing visual perception, hardware abstraction, and behavior generation [4]. Joseph chooses ROS over other platforms such as Player, YARP, and Orocos because of its support for high end capabilities like SLAM (Simultaneous Localization and Mapping) [5]. For these reasons, it has been gaining growing popularity in the robotics community.

The goal of this work is to develop a solid ROS-based interface for a manipulator arm which is designed to accomplish complicated tasks. For a demonstration purpose, the task of balancing a ball on a plate has been set on the manipulator arm (Robai Cyton Gamma 300). This paper presents the development of ROS-based interface for the chosen manipulator integrated with a low-level control and vision tracking system and demonstrates a task of balancing a ball on a plate.

2. System Overview

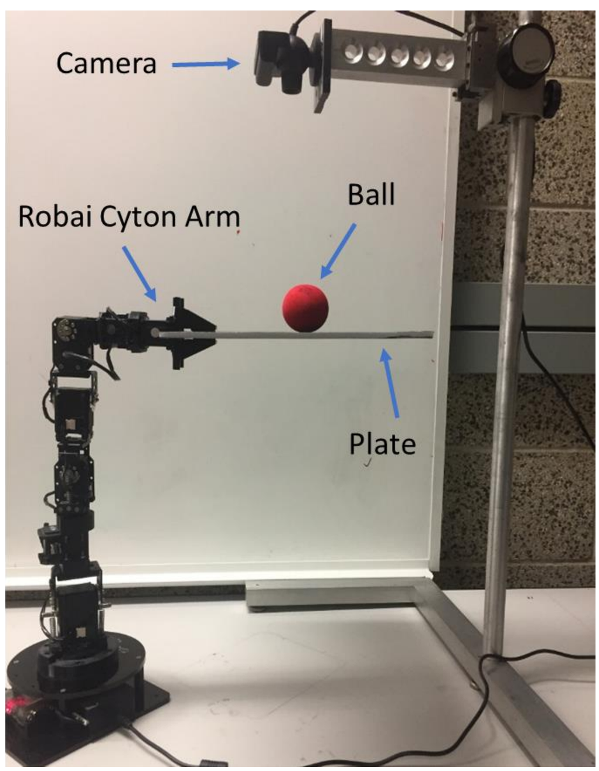

Figure 1. Experimental setup of the system

The system used for the demonstration consists of a 7-DOF robotic arm (Robai Cyton Gamma 300) having 7 servo motors (Dynamixel) as joint actuator, a plate-type end effector, and a vision camera system for tracking. Even though the manipulator has kinematic redundancy with its 7 DOFs, only 2 axes are used as it is sufficient to control the plate for the balancing task. The vision system uses a low-cost USB camera (Sony PlayStation Eye) with frame rates of up to 120 Hz.

As shown in Figure 1, the plate is held by the gripper of the arm. The plate was given two-axis motion by using two joints of the robot, which is explained in more detail in Section 4. For material of the plate and ball, foam and soft rubber were selected respectively, based on the payload limitations of 300 g of the robot arm. Although the model-based control algorithm is supposed to work with any mass and size of the ball in principle, a smaller ball was considered to provide enough workspace on the plate along with the payload constraint.

3. ROS with the System

3.1 Structure of the System Software in ROS

ROS is an open-source framework and integrated development environment (IDE) for writing robot software [6]. One central idea behind ROS is to create blocks of software or tools for configuring, visualization, manipulation, sensing, debugging robotic components that can be reused with minimal modifications. Another is to overcome limited capabilities when using manufacturer's GUI-based interface. Since the manufacturer's interface software (CytonViewer) of Robai Cyton Gamma does not allow full control of the manipulator arm, we programmed an ROS-based interface that enables direct control of the Dynamixel joint actuators.

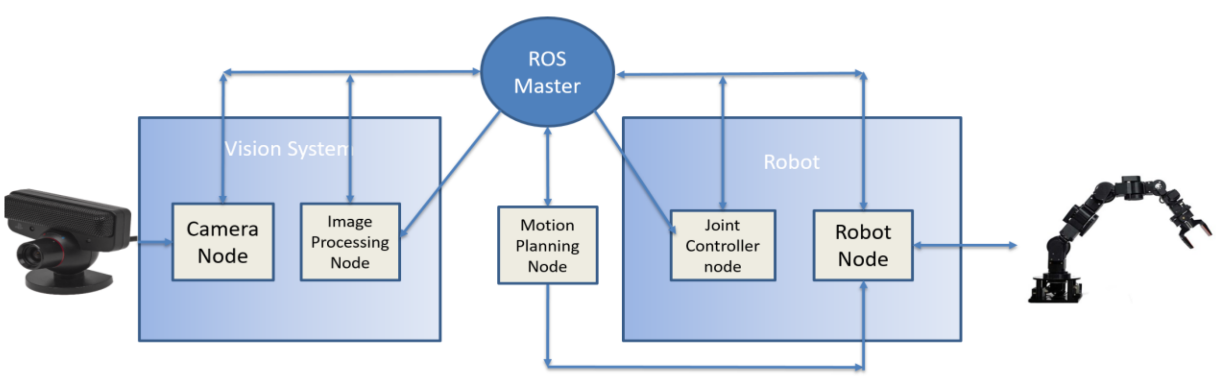

Figure 2. Employed software architecture of the system in ROS.

The version of ROS Indigo over Linux OS of Ubuntu Trusty 14.04 was selected to make the best use of pre-developed work such as packages for Dynamixel motors and USB cameras.

A package which has both executable and supporting files comprises of nodes, external libraries, data, configuration files and one xml configuration file. The Master is a node for declaration and registration service, which makes it possible for nodes to find each other and exchange data. Each node created has its own topics and services which can be used to publish or subscribe the messages [6].

The architecture of the system created using ROS is shown in Figure 2. Dynamixel drivers and their joint torque controllers are available to the public [7], [8]. ROS packages for motion planning and control are newly written in C++ and taking inputs or sending control actions from/to the robot and vision system through nodes created in each package.

3.2 Vision Tracking System in ROS

A low-cost Sony PlayStation Eye USB camera is mounted above the plate to track the position of the ball as shown in Figure 1. The object tracking is done by a series of processing on grabbed image frame to get the position of the ball [9]: conversion to grey scale to HSV, thresholding, and filtering. OpenCV libraries are used to accomplish the task of tracking a red-colored object [10].

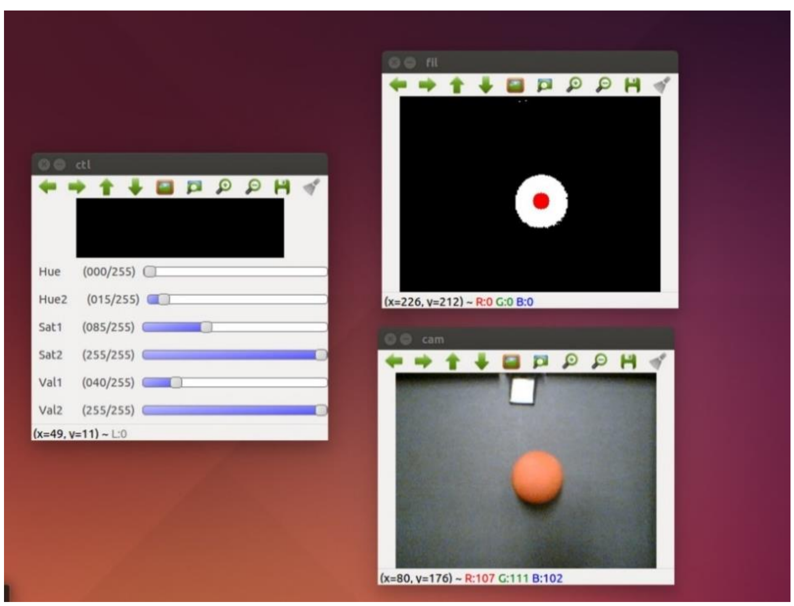

Figure 3. Image windows of the vision tracking system: (In the clockwise from the left) slide bar control window, tracking-processed image, and original image.

A slide bar control windows are created to set the values for HSV, thresholds, and filtering. The PlayStation Eye camera can produce an image frame with a resolution of 320×240 pixels at a frame rate of 120 Hz. Therefore, the camera is strategically mounted on the fixture above the plate of 320×240 mm^2 at a height so that it can cover the entire plate with a position resolution of 1 mm. The vision tracking system working in multiple windows in Ubuntu OS is presented in Figure 3.

4. System Modeling and Control

The ball-on-plate on the robot arm is simplified into two decoupled ball-on-beam systems such that the motion of the ball in each axis is treated as an independent ball-on-beam problem. The ball is assumed to be in contact with the plate all the time under the no-slip assumption.

4.1 Equations of Motions

First, the equations of motion of a ball-on-beam is derived using Lagrangian method [11]. The coordinate frame used with the system is shown in Figure 4.

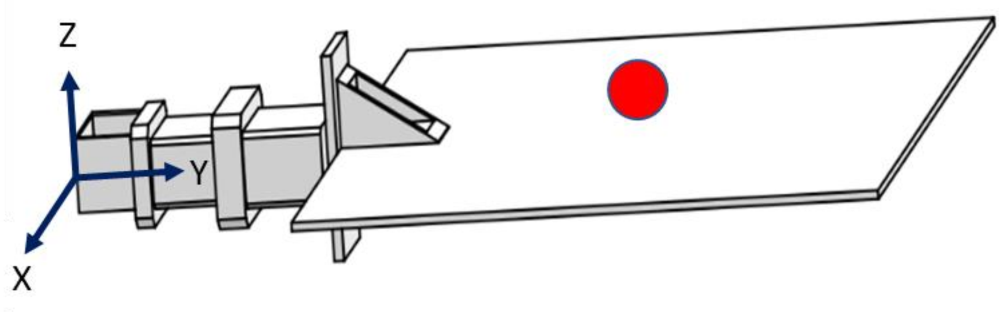

Figure 4. The coordinate frame used in the system.

The two plate motions are rotations about X- and Y-axes. We derive the equations of motion for each rotational motion as below.

About Y-axis:

The parameters of the ball are given by the mass m_b, radius r_b, moment of inertia J_b and the plate by moment of inertia J_p about Y-axis. Then, using the body-attached coordinate frame u_1–u_2 as defined in Figure 5, the velocity $\tilde{v}_b$ and angular speed ω_y of the ball are written as, respectively, given the position $x_b\hat{u}_1$,

$$\tilde{v}_b = \dot{x}_b\hat{u}_1 + x_b\dot{\theta}_y\hat{u}_2, \qquad \omega_y = \frac{\dot{x}_b}{r_b}$$

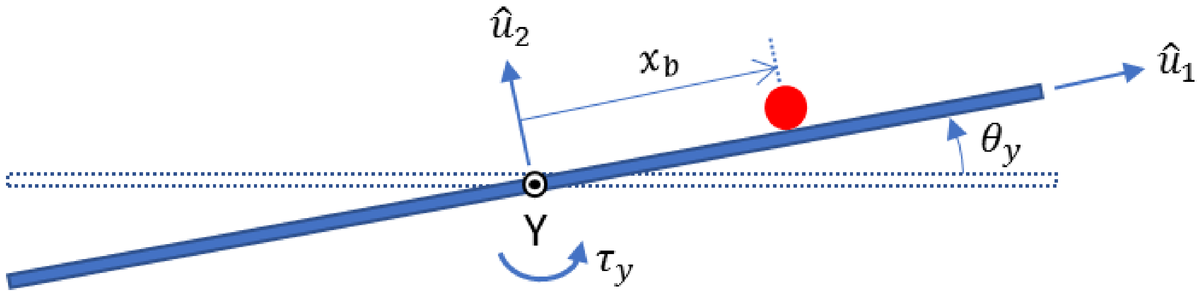

Figure 5. Y-axis motion of the system.

We use Euler-Lagrange's equation to derive the equations of motion, with $q = [x_b, \theta_y]^T$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = \tau \tag{1}$$

where $L = K - V$ and τ denotes the generalized force vector. Here, K and V are the kinetic and potential energies of the system, respectively and they are given by

$$K = \frac{1}{2}m_b v_b^2 + \frac{1}{2}J_b\omega_y^2 + \frac{1}{2}J_p\dot{\theta}_y^2 = \frac{1}{2}\left(m_b + \frac{J_b}{r_b^2}\right)\dot{x}_b^2 + \frac{1}{2}\left(m_b x_b^2 + J_p\right)\dot{\theta}_y^2$$

$$V = m_b g x_b \sin\theta_y$$

Finally, the equations of motion about Y-axis of rotation are obtained as

$$\left(m_b + \frac{J_b}{r_b^2}\right)\ddot{x}_b - m_b x_b\dot{\theta}_y^2 + m_b g\sin\theta_y = 0 \tag{2}$$

$$\left(m_b x_b^2 + J_p\right)\ddot{\theta}_y + 2m_b x_b\dot{x}_b\dot{\theta}_y + m_b g x_b\cos\theta_y = \tau_y \tag{3}$$

where τ_y denotes the torque input on the Y-axis joint.

About X-axis:

The rotational motion about X-axis can also be described as a ball-on-beam system, similar to that of the motion about Y-axis. However, the crucial difference of this motion is the center of mass

(COM) of the system is not located at the axis of rotation as shown in Figure 6. Also, the equilibrium point–the center of the plate– at which the ball is designed to be balanced is not at the axis of rotation, i.e., $y_b \neq 0$ at the balanced position.

We use again Euler-Lagrange's equation to derive the equations of motion. However, due to the aforementioned differences, compared to the previous case, it has additional kinetic and potential

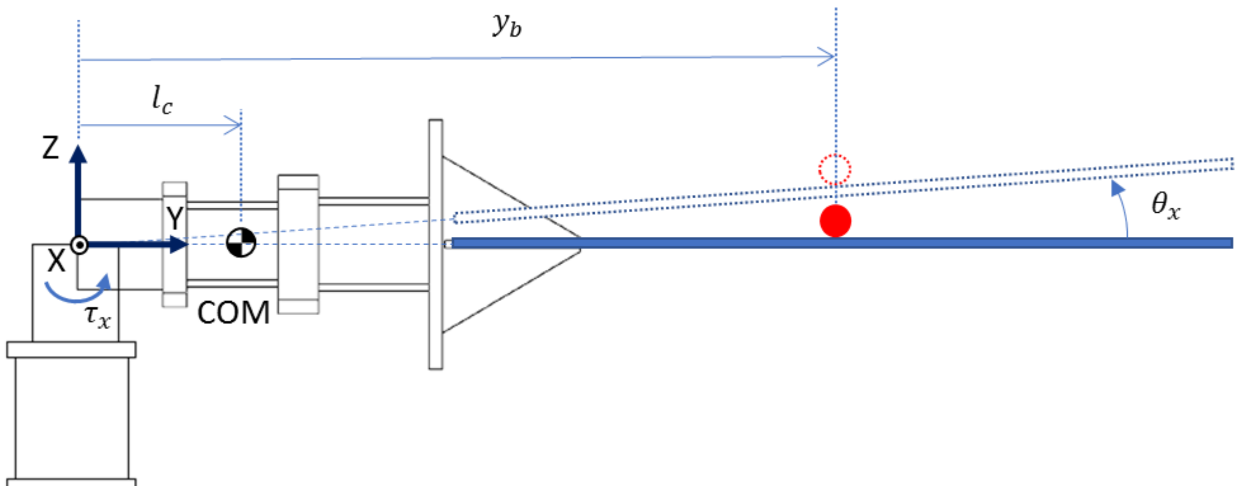

Figure 6. X-axis motion of the system.

energies. Consequently, the kinetic and potential energies of the system are given by

$$K = \frac{1}{2}m_b v_b^2 + \frac{1}{2}J_b\left(\frac{\dot{y}_b}{r_b}\right)^2 + \frac{1}{2}m_s v_p^2 + \frac{1}{2}J_s\dot{\theta}_x^2$$

$$V = m_b g y_b \sin\theta_x + m_s g l_c \sin\theta_x$$

where m_s and J_s are the mass and moment of inertia of the entire system including the links, gripper, and plate; l_c is the location of the COM measured from the axis of rotation; and v_p is the velocity of the system at the COM, given by $v_p = l_c\dot{\theta}_x$. Finally, we obtain the equations of motion as follows:

$$\left(m_b + \frac{J_b}{r_b^2}\right)\ddot{y}_b - m_b y_b \dot{\theta}_x^2 + m_b g \sin\theta_x = 0 \tag{4}$$

$$(m_b y_b^2 + J_s + m_s l_c^2)\ddot{\theta}_x + 2m_b y_b \dot{y}_b \dot{\theta}_x + (m_b y_b + m_s l_c) g \cos\theta_x = \tau_x \tag{5}$$

4.2 Linearization around the Center of the Plate.

The objective of the proposed demonstration is to balance the ball at the center of the plate. Assuming the rotation angles θ_x and θ_y are small, the control problem can be much simpler by further applying approximation linearization for the equations of motions in Eqs. (2) to (5).

About Y-axis:

Equations (2) and (3), which describes the motion about Y-axis, can be represented in the state space form with $z = \left[x_b, \theta_y, \dot{x}_b, \dot{\theta}_y\right]^T$

$$\dot{z} = f\left(z, \tau_y\right) = \begin{bmatrix} \dot{x}_b \\ \dot{\theta}_y \\ -\frac{5}{7} g \sin\theta_y + \frac{5}{7} x_b \dot{\theta}_y^2 \\ \frac{1}{m_b x_b^2 + J_p}\left(-2 m_b x_b \dot{x}_b \dot{\theta}_y - m_b g x_b \cos\theta_y + \tau_y\right) \end{bmatrix}$$

Note that we assume the mass of the ball is uniform, so the moment of inertia of the ball $J_b = \frac{2}{5} m_b r_b^2$. This system is linearized about an equilibrium point $z^* = \left[x_b^*, \theta_y^*, \dot{x}_b^*, \dot{\theta}_y^*\right]^T = [0, 0, 0, 0]^T$ and $\tau_y^* = 0$. By the linearization method [12], the system is now given by

$$\dot{z} = Az + B\tau_y \tag{6}$$

where

$$A = \left.\frac{\partial f}{\partial z}\right|_{z^*, \tau_y^*} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{5}{7} g & 0 & 0 \\ -\frac{m_b g}{J_p} & 0 & 0 & 0 \end{bmatrix}, \qquad B = \left.\frac{\partial f}{\partial \tau_y}\right|_{z^*, \tau_y^*} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{J_p} \end{bmatrix}.$$

Alternatively, Eq. (6) can be expressed in terms of x_b and θ_y.

$$\ddot{x}_b + \frac{5}{7} g \theta_y = 0 \tag{7}$$

$$J_p \ddot{\theta}_y + m_b g x_b = \tau_y \tag{8}$$

About X-axis:

Similarly, we can approximately linearize Eqs. (4) and (5) around the center of the plate, given by $z^* = \left[y_b^*, \theta_x^*, \dot{y}_b^*, \dot{\theta}_x^*\right]^T = [y_b^*, 0, 0, 0]^T$ and $\tau_x^* = (m_b y_b^* + m_s l_c)g$. It should be noted that the equilibrium point of y_b^* is not zero in this case since the equilibrium point is off the axis of rotation by y_b^* as seen in Figure 6. Consequently, the equilibrium torque τ_x^* is not zero either.

The equations of motion in (4) and (5) can also be rewritten in the state space form with $z = \left[y_b, \theta_x, \dot{y}_b, \dot{\theta}_x\right]^T$ as

$$\dot{z} = f(z, \tau_x) = \begin{bmatrix} \dot{y}_b \\ \dot{\theta}_x \\ -\frac{5}{7} g \sin\theta_x + \frac{5}{7} y_b \dot{\theta}_x^2 \\ \frac{1}{m_b y_b^2 + J_s + m_s l_c^2} \left(-2 m_b y_b \dot{y}_b \dot{\theta}_x - (m_b y_b + m_s l_c)\cos\theta_x + \tau_x\right) \end{bmatrix}$$

Due to the nonzero equilibrium point, the approximation linearization yields

$$\dot{\Delta z} = A\Delta z + B\Delta\tau_x \tag{9}$$

where

$$\Delta z = z - z^*, \qquad \Delta\tau_x = \tau_x - \tau_x^*,$$

$$A = \left.\frac{\partial f}{\partial z}\right|_{z^*, \tau_x^*} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{5}{7}g & 0 & 0 \\ -\frac{m_b g}{m_b y_b^* + J_s + m_s l_c^2} & 0 & 0 & 0 \end{bmatrix}, \text{ and } B = \left.\frac{\partial f}{\partial \tau_x}\right|_{z^*, \tau_x^*} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{m_b y_b^* + J_s + m_s l_c^2} \end{bmatrix}.$$

Alternatively,

$$\ddot{y}_b + \frac{5}{7} g\theta_x = 0 \tag{10}$$

$$(m_b y_b^* + J_s + m_s l_c^2)\ddot{\theta}_x + m_b g(y_b - y_b^*) = \tau_x \tag{11}$$

4.3 Controller Design

In this section, we discuss the controller design based on the linearized equations of motion, described by Eqs. (6) and (9) for each rotational motion, respectively. The pole placement method, a well-known linear control technique, is used such that

$$\tau = -Kz \tag{12}$$

where τ denotes the torque control input, z the full state feedback, and K the 1×4 gain matrix given by $K = [K_1\ K_2\ K_3\ K_4]$. With this form of control law, the equations of motion can be rewritten as

$$\dot{z} = (A - BK)z \tag{13}$$

The stabilization to the equilibrium point is guaranteed by determining the gain matrix K in a way that $(A - BK)$ is a Hurwitz.

It should be mentioned that in practice the Cyton Gamma manipulator does not allow torque inputs, but velocity inputs. Hence, we use an approach of numerically integrating the equations of motion with the calculated torque input in Eq. (12) over one control loop interval in order to estimate the target velocity at the next control step. Then, we use the calculated target velocity as the input to the manipulator at the current control step. The schematic of the full state feedback control loop is provided in Figure 7.

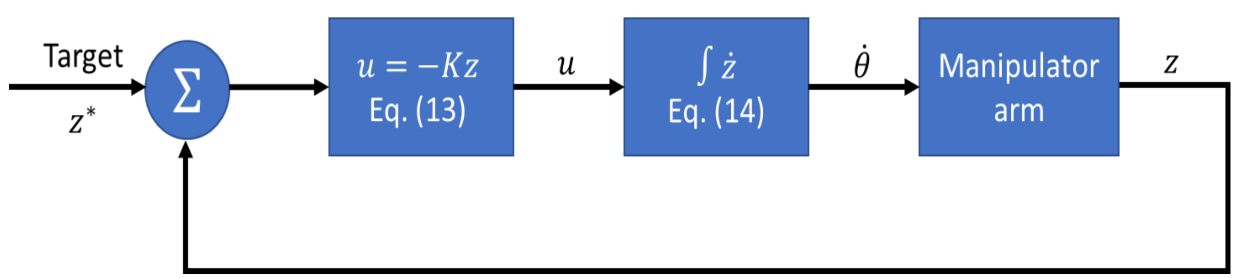

Figure 7. Schematic of the feedback control loop.

5. Simulation and Experimental Results

The values of the system parameters used in simulations and experiments are listed in Table 1. Since the rotating links which are considered part of the plate are irregularly shaped, the moment of inertia of the plate (together with the links) and the COM (see Figure 6) was determined using a model developed on a CAD modeling software, SOLIDWORKS.

5.1 Simulation Results

In the application of the pole placement method, the gain matrix K in Eq. (12) was chosen as $K = [-44.196, 0.152, -0.055, 0.026]\times 10^{-3}$ and $K = [-0.104, 0.186, -0.065, 0.033]$ for the rotational motion about Y-axis and X-axis, respectively. Which are the cases when the poles of

Table 1. System parameter values used in simulation and experiment.

System Parameters	Variables	Actual Values
Mass of the ball	m_b	0.0045 [kg]
Moment of inertia of the ball	J_b	1.046×10^{-6} [kg·m^2]
Radius of the ball	r_b	0.024 [m]
Mass of the plate	m_s	0.0228 [kg]
Moment of inertia of the plate (about Y-axis)	J_p	1.702×10^{-6} [kg·m^2]
Moment of inertia of the plate (about X-axis)	J_s	1.60577×10^{-3} [kg·m^2]
Distance to the plate center from the rotation of axis	y_b^*	0.23 [m]

the closed-loop system in Eq. (13) are both at $(-3, -3.5, -4, -4.5)$. Figure 8 shows the simulation results. The initial conditions are $(x_b, \theta_y, \dot{x}_b, \dot{\theta}_y) = (0.1\text{ m}, 0, 0, 0)$ and $(y_b, \theta_x, \dot{y}_b, \dot{\theta}_x) = (0.33\text{ m}, 0, 0, 0)$, respectively. Note that the initial condition of $y_b(0) = 0.33$ m implies the ball is to be at 0.1 m from the center of the plate. As expected, the ball is stabilized to the center of the plate where $(x_b, y_b) = (0, 0.23)$ with eventual zero velocity of the ball.

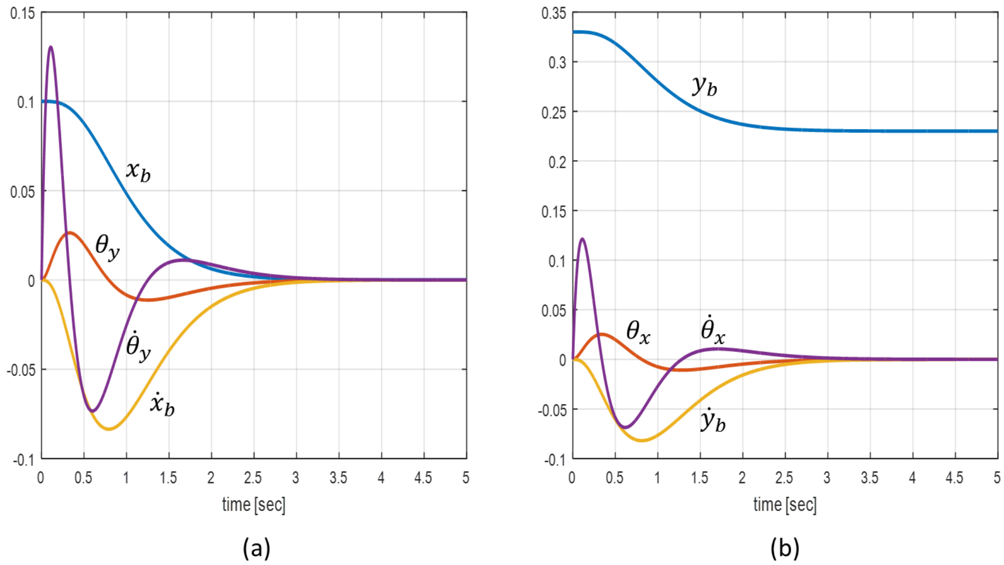

Figure 8. Simulation results: (a) Rotational motion about Y-axis (see Fig. 5); (b) Rotational motion about X-axis (see Figure 6).

5.2 Experimental Results

The system performs balancing control to some extent. However, it was not satisfactorily able to keep the ball within a small area around the center of the plate. Figure 9 shows measured ball position during balancing control. Figure 10 shows the joint angular velocities for each joint axis. It is noticed that some delay exists for the actual velocities to follow the target (commanded) velocities.

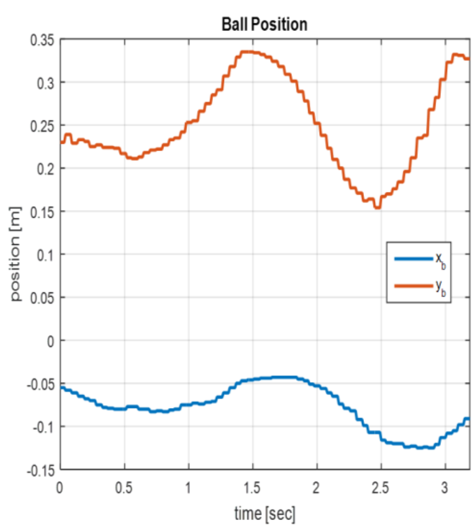

Figure 9. Actual ball positions

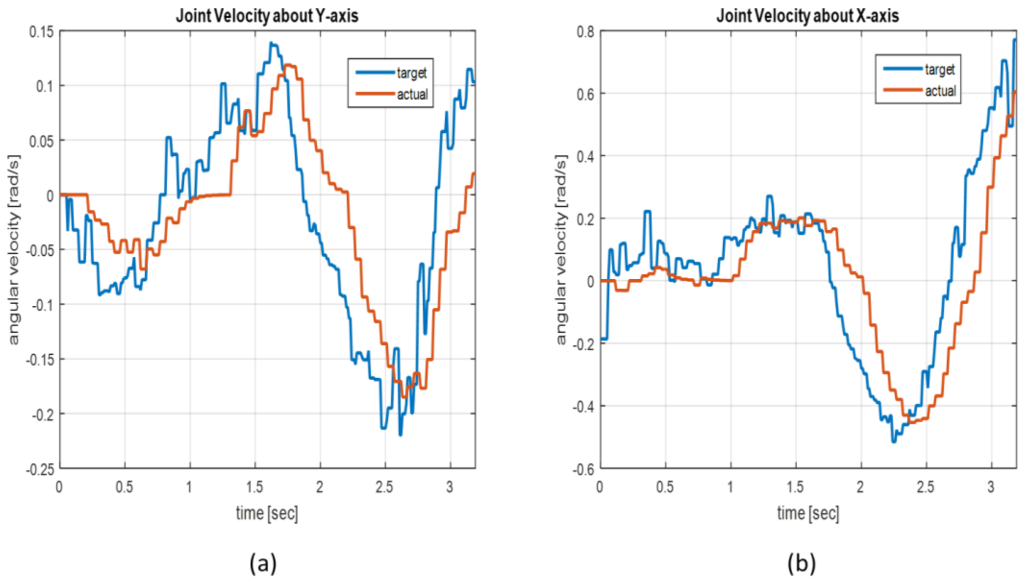

Figure 10. Target and actual Joint angular velocities: (a) joint about Y-axis; (b) joint about X-axis (see Figures 5 and 6 for the corresponding rotational motions, respectively.)

Causes of such results may include discrepancy between the actual and simplified dynamics, invalidity of the small angle assumption, delays in image processing and communication, insufficient control loop rate, unwanted dynamics from the flexible plate chosen, and so on. Other linear control technique such as LQR to optimize the control action could be further applied to improve the task performance.

6. Suggestions for Courses and Lab Sessions

This experimental set-up is supposed to be used in courses teaching dynamic systems and control and/or robot programming. In the dynamic systems and control course, we're first planning to use the system as a demonstration of a feedback system. It is designed to give a good understanding of what feedback control is, how it works, and how it is applied to real applications. With a working system, it can also be used in a lab session in which students can design a PID feedback control by determining its control gains properly, or even test other types of control laws in an advanced course. Through this lab, students will acquire an ability to apply the knowledge of control theories they have learned from class to an actual system.

For the robot programming course, this system will be used as a good project to carry out. As it is shown in this paper, students will learn not only ROS, but also programming to run various sensors and actuators along with a vision system. With adequate instructions, we believe students will be able to complete this project in one semester through four separate modules: vision, actuation (servo motors), control, and system integration. Additionally, other relevant topics such as robot dynamics and digital control can also be taught with the system presented in this paper.

7. Conclusion

This paper presents a ROS-based solid multi-platform interface to control a 7-DOF manipulator arm integrated with a vision tracking system. This system is developed such a way that each component communicates each other to receive the state feedback from the vision system and motors, and to compute and send control commands to achieve preset control tasks. For a demonstration purpose, the task of balancing a ball on a plate was performed and simulation and experimental results are provided. The experimental results were not satisfactory compared to the simulation results, but there is room to improve. This system with the proposed approaches has potential to be used as a testbed for students to learn ROS with control theories used in robotics.

References

[1] "What is ROS?" [Online]. Available: http://wiki.ros.org/ROS/Introduction.

[2] A. Bonarini, M. Matteucci, M. Migliavacca, and D. Rizzi, "R2P: An open source hardware and software modular approach to robot prototyping," *Rob. Auton. Syst.*, vol. 62, no. 7, pp. 1073–1084, 2014.

[3] M. Quigley *et al.*, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, 2009, vol. 3, p. 5.

[4] P. Allgeuer and M. Schwarz, "A ROS-based Software Framework for the NimbRo-OP Humanoid Open Platform," in *IEEE-RAS International Conference on Humanoid Robots*, 2013.

[5] L. Joseph, *Mastering ROS for Robotics Programming*. Packt Publishing, 2015.

[6] A. Martinez and E. Fernández, *Learning ROS for Robotics Programming*. Packt Publishing, 2013.

[7] "Creating a joint torque controller." [Online]. Available: http://wiki.ros.org/dynamixel_controllers/Tutorials/Creating a joint torque controller.

[8] "Dynamixel Motor." [Online]. Available: http://wiki.ros.org/dynamixel_motor.

[9] M.-T. Ho, Y. Rizal, and L.-M. Chu, "Visual Servoing Tracking Control of a Ball and Plate System: Design, Implementation and Experimental Validation," *Int. J. Adv. Robot. Syst.*, vol. 10, no. 7, p. 287, Jul. 2013.

[10] "Tracking colored objects in OpenCV." [Online]. Available:

http://www.aishack.in/tutorials/tracking-colored-objects-opencv/.

[11] C. G. Bolívar-Vincenty and Beauchamp-Báez, “Modelling the Ball-and-Beam System From Newtonian Mechanics and from Lagrange Methods,” in *Twelfth LACCEI Latin American and Caribbean Conference for Engineering and Technology*, 2014.

[12] H. Benaroya and M. L. Nagurka, *Mechanical vibration : analysis, uncertainties, and control*. CRC Press, 2010.