Techno-Science Research Journal 11(2) (2023) 76-81

I.T.C

Content list available at ITC

Techno-Science Research Journal

Journal Homepage: http://techno-srj.itc.edu.kh/

Techno-Science Research Journal

Development of Control Framework Based on ROS Platform for a 3-Axis Gimbal

Rattana Seng [1*], Sarot Srang [1]

[1] *Dynamics and Control Laboratory, Department of Industrial and Mechanical Engineering, Institute of Technology of Cambodia, Russian Federation Blvd., P.O. Box 86, Phnom Penh, Cambodia*

Received: 06 October 2023; Accepted: 07 July 2023; Available online: December 2023

Abstract: *A Gimbal is a type of instrument used for 3D space orientation control. Over the past decade of research on 3-axis tracking gimbal, many research have been contributing to design and control of the system improving the accuracy and quality of the design. This research aims to develop a control framework for the system making it more user-friendly and more convenient for non-developers to use the final product. More specifically, this research focuses on system integration, controller design & simulation. We describe kinematic modeling before properly implementing controllers for velocity control and position control to reach desired pose of the end-effector. To validate a proper controller for suitable use, the simulation of the kinematics and motion control has been performed. We propose a study of 2 possible controllers, that is, conventional PI controller for velocity control and PD controller for position control of the actuators. ROS2 nodes are developed and used as a middleware for interfacing the motion control and command signal sent from another application. Two devices with different platform are used to communicate and send data to each other over the ROS framwork. One device is used to command the desired pose of the gimbal and visualize the response of the the system. The other device is used for computing feedback control of the system based on kinematics equation. The simulation result in this work is also presented for validation of this framework. Finally, we have found the simulation result of control system of gimbal start to converge to desired pose commanded from another device over the ROS framework that we have created..*

Keywords: 3-axis gimbal, Controller design, Motion Control, ROS

1. INTRODUCTION

A gimbal is a type of instrument used for 3D space orientation control. They can be installed on a helicopter or attached to a fixed frame to preserve axis alignment. Its applications are becoming more and more prevalent in our everyday life ranging from photography to satellite tracking. Although it is mostly being used for stabilization, the main purposes of a gimbal can either be for stabilizing, tracking, or all two together. Recognizing the benefits and viability of gimbals in tracking applications, this research mainly focuses on the development of a framework for gimbal systems to be used for sound-tracking application.

As for software development, the mathematical model of 3-axis gimbal is very important for modeling the simulation model of its motions and designing the control algorithms. The motion can be described in different equations; namely, kinematic and dynamic equations. Kinematics is the motion geometry of the robot manipulator from the reference position to the desired position with no regard to forces or other factors that influence robot motion. Whereas, the dynamics of a robotic describes how the robot moves in response to the actuator forces and other external forces acting on the system.

To have the gimbal actuators obey the reference signal, a regulator must be created. The regulator must be built on a dynamic model of the gimbal in order to react quickly while staying within the operating limits of the inner gimbal damping mechanism. A dynamic model is excellent for simulation-related purposes. The motion of the gimbal can be controlled by a feedback control system in order to move the gimbal according to a desired command provided by the trajectory planner. There are many algorithms such as adaptive control, PID controller, fuzzy logic controller, fractional order controller, and so on. Among these controllers, the PID controller is the most commonly used controller in industries by virtue of its effectiveness, simplicity, and feasibility [1].

To make the system more user-friendly and convenient for communicating with other applications, it is important to build a framework to optimize the time spent on reconfiguring the system in case some parameters need to be changed, provided a

* Corresponding author: Rattana Seng
E-mail: seng_rattana@gsc.itc.edu.kh; Tel: +855-96 353 7670

generic and straightforward control framework to implement and manage robot controllers to improve both real-time performance and sharing of controllers. Chitta et al., 2017 [2] presented a robot operating system (ROS) to develop and test autonomous control functions. ROS can support an extensive range of robotic systems and can be regarded as a low-cost, simple, highly flexible, and re-configurable system. Chivarov et al., 2019 [3] aimed to develop a cost-oriented autonomous humanoid robot to assist humans in daily life tasks and studied the interaction and cooperation between robots and Internet of Things (IoT) devices by using ROS platform.

The remainder of this paper is organized as follow: in Methodology section we will show the design and configuration of our 3-axis gimbal, its kinematic equations, control architecture, and explain the framework of system integration. Followed by section 3 which discusses the result of a simulation using this developed framework. The last section conclude this paper and introduce the following step for this work.

2. METHODOLOGY

2.1. Gimbal configuration

The three axes of rotation mentioned in the Three-Axis Gimbal are shown in Fig.1. These rotations are called yaw rotation, pitch rotation, and roll rotation. The yaw rotation is a rotation around z axis. Pitch rotation is a rotation around y axis. As well as roll rotation, there is rotation around x axis.

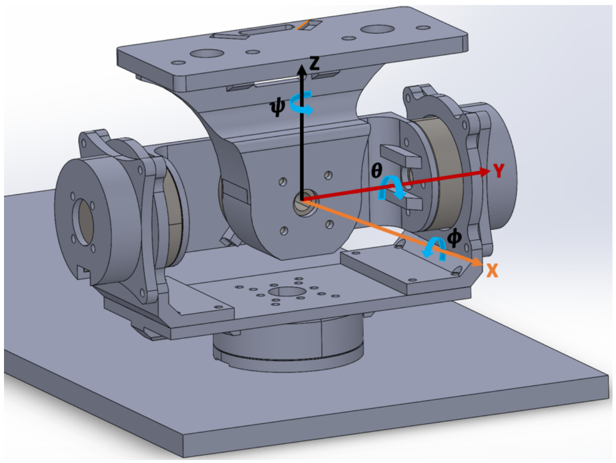

Fig. 1. Gimbal configuration.

2.2 Forward Kinematics

To obtain the kinematics equations of the gimbal, the well-known Denavit-Hartenberg convention will be used. There are four coordinate frames needed to be defined that consist of the gimbal base frame (X_0, Y_0, Z_0), frame 1 (X_1, Y_1, Z_1), frame 2 (X_2, Y_2, Z_2) and end-effector frame (X_E, Y_E, Z_E).

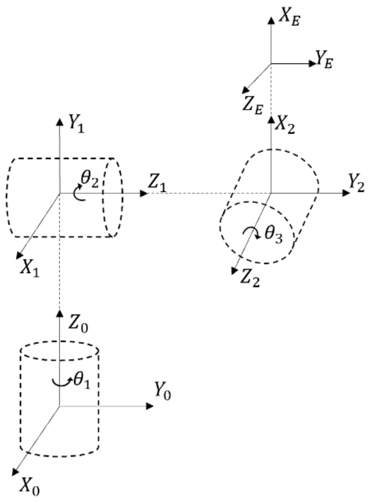

Fig. 2. Link frame assignment.

The forward kinematics equation is used to find the orientation of the end-effector with the known joint variable θ_1, θ_2 and θ_3. This equation can be derived by using the Denavit-Hartenberg convention, as shown in Table 1. We assume that the center of origin of the three axes is located at the same point.

Table 1 Denavit-Hartenberg parameters

Link i	θ_i	α_i	a_i	d_i
1	θ_1	$-\pi/2$	0	0
2	θ_2	$-\pi/2$	0	0
3	θ_3	0	a_3	0

where:
θ_i = is joint angle (rotation around Z_{i-1})
α_i = is link twist (rotation around X_i from Z_{i-1} to Z_i)
a_i = is link length (displacement along X_i)
d_i = is link offset (displacement along Z_{i-1})

The homogeneous transformation from link $i-1$ to link i can be written as:

$$T_{i-1,i} = \begin{bmatrix} c(\theta_i) & -c(\alpha_i)\,s(\theta_i) & s(\alpha_i)\,s(\theta_i) & a_i c(\theta_i) \\ s(\theta_i) & c(\alpha_i)\,c(\theta_i) & -s(\alpha_i)\,c(\theta_i) & a_i s(\theta_i) \\ 0 & s(\alpha_i) & c(\alpha_i) & d_i \\ 0 & 0 & 0 & 0 \end{bmatrix}, \text{(Eq. 1)}$$

where:
$c(\theta_i) = \cos(\theta_i)$,
$s(\theta_i) = sin(\theta_i)$.

Therefore, the homogeneous transformation from base link to end-effector can be written as:

$$T_{0,3} = T_{0,1} T_{1,2} T_{2,3}$$

Alternatively, the transformation matrix of each link of the robot can be written in form as in Eq.2 below,

$$T_{0,3} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(Eq. 2)}$$

where

r_{ij} is the element of the orientation matrix of the link frame, and p_x, p_y and p_z is the position vector of the origin of that frame.

2.3 Inverse kinematics

The inverse kinematics problem is to define the joint variable given the end-effector position. Because the center or origin of the two axes is located at the same point, we can apply spherical coordinate system of the center of origin as shown in Fig. 3.

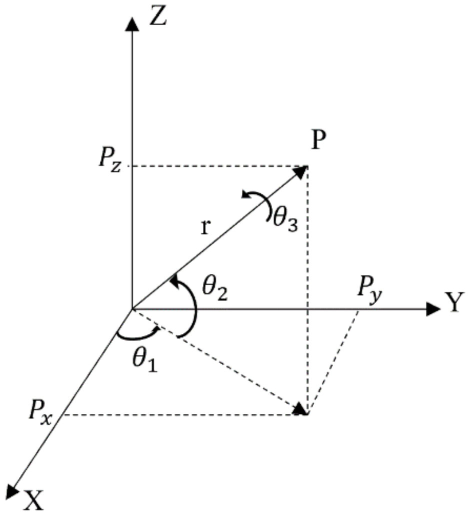

Fig. 3. Spherical coordinate system.

Therefore, we can find the angle of each joint based on spherical coordinate system as:

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} arctan2(P_y, P_x) \\ arctan2(P_z, r) \\ tilt \end{bmatrix} \quad \text{(Eq. 3)}$$

Where:

$$r = \sqrt{P_x^2 + P_y^2}$$

And tilt is angle of the other degree rotate respect to the horizontal ground plan. Its rotation depends on payload that we equip on the end-effector. For example, the camera is mounted to align with the 3rd axis of the gimbal, the relationship between the tilt angle and θ_3 is quite trivial.

2.4 Feedback control of gimbal

The Fig. 4 shows the control architecture of the 3-axis gimbal. It is a close loop feed forward control that take the desired pose from trajectory.

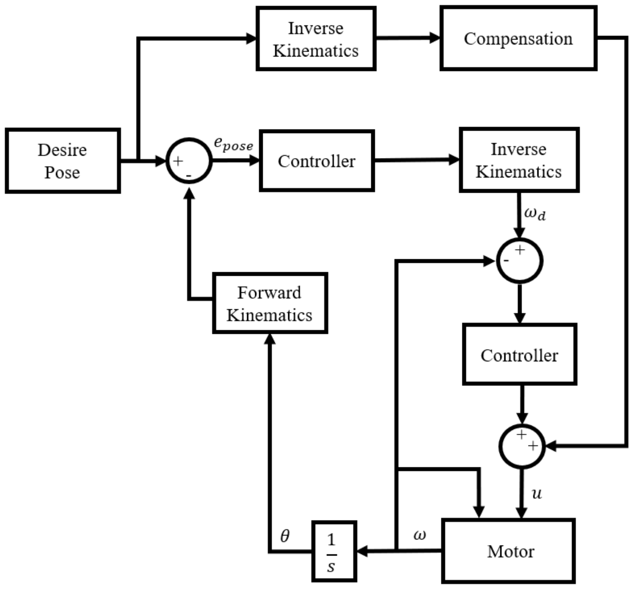

Fig. 4. Control architecture design for 3-axis gimbal.

The desired pose of the end-effector goes through inverse kinematics equation to find the desired angle of each joint and then use as the reference of actuators to rotate the gimbal to track the desired position. For actuators control we use the velocity control with PI (Propositional and integral) controllers. It is also a feedback control. The rotation of each joint goes through forward kinematics block to generate the output which is the actual positon of end-effector in order to verify with the command desired from trajectory. Moreover, to make tracking trajectory more efficient we have combined the feedback control with feedforward control.

2.6 Control framework

The framework for controlling the 3-axis gimbal is built based on ROS platform. We combine the advantages of MATLAB/Simulink and ROS. MATLAB is used to compute all feedback control, and ROS is used as middleware to deliver information from one node to another node. In this study, we have created four ROS nodes. The first one is for publishing a topic that contain data of joystick or generate random numbers into ROS environment or with a specific application. The second node subscribes the topic from the first node in order to take data and generate the desired trajectory for gimbal and publish it to the another node. The third node is in Simulink it subscribes to the topic trajectory generation in order to take data to use as input of feedback control. After that, the signal output from Simulink, is published back to ROS environment for simulation via another topic subscribed by the fourth node. Before we can simulate

CAD model in ROS environment, we have to create the CAD model in Solidwork and then export into URDF file.

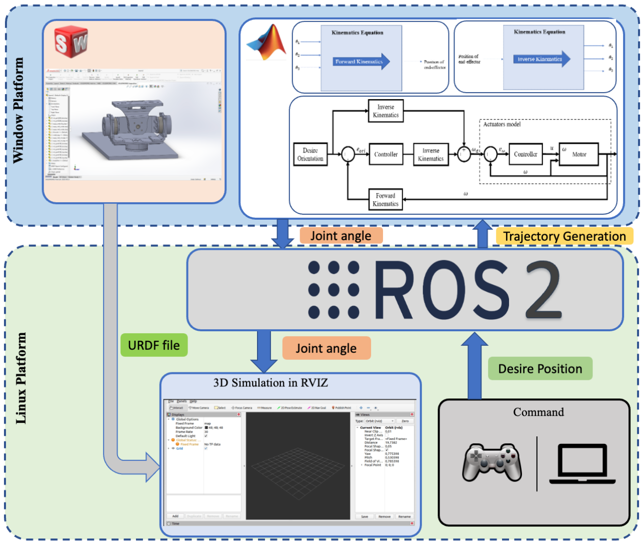

Fig. 5. Control framework for 3-axis gimbal.

3. RESULTS AND DISCUSSION

The result from the simulation is shown below. The control system of 3-axis gimbal is performed in MATLAB/Simulink. The Figure 6 show the CAD design in Solidworks which has been imported into RVIZ in ROS environment through URDF file in order to visualize its behavior.

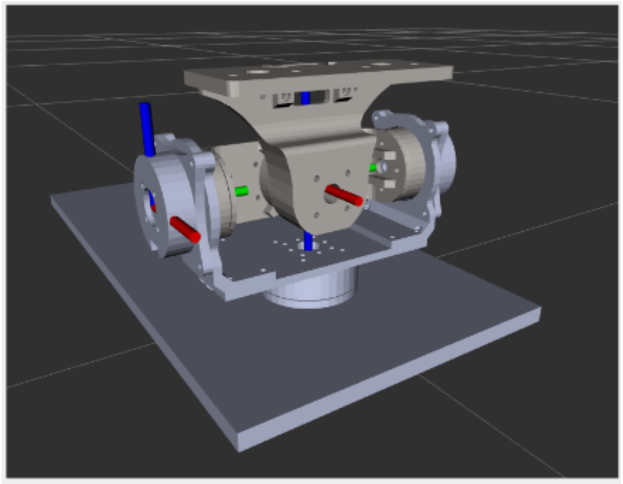

Fig. 6. Importing URDF file into ROS environment.

Fig. 7 shows the result of node graph generated in ROS platform. There are 4 nodes created for sharing information with each other. There are three nodes run on Linux platform. Two nodes, which are command node and trajectory generation node. They communicate with each other via pose_trajectroy topic. And another node run on Window platform, it subscribed the topic pose desired from trajectory generation node and publish joint angle to the topic sml_joint state topic. The last node that run on Linux subscribes that topic to visualize the CAD model.

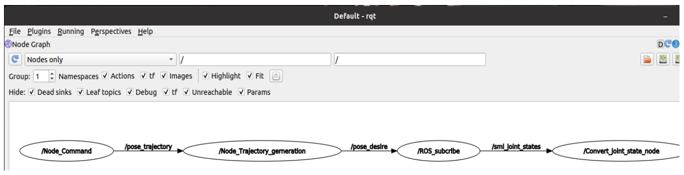
Default - rqt
File Plugins Running Perspectives Help
Node Graph
Nodes only
/
/
Group: 1 Namespaces ✓ Actions ✓ tf ✓ Images ✓ Highlight ✓ Fit
Hide: ✓ Dead sinks ✓ Leaf topics ✓ Debug ✓ tf ✓ Unreachable ✓ Params
/Node_Command
/pose_trajectory
/Node_Trajectory_gemeration
/pose_desire
/ROS_subcribe
/sml_joint_states
/Convert_joint_state_node

Fig. 7. Relation between each node in RQT graph.

Fig. 8 shows a graph representing the position of the end-effector $P_{x\,desire}$ and $P_{x\,actual}$ which is the respond of the sytem. It shows a position in unit vector versus time in second. In this simulation, we have chosen the desired position of end-effector as $P_{x\,desire} = r \times \cos(\alpha)\cos(\psi)$, $P_{y\,desire} = r \times \cos(\alpha)\sin(\psi)$ and $P_{z\,desire} = r \times \sin(\alpha)$, r is unit vector. And α, ψ have been chosen to be $0.32\sin(\frac{\pi}{5}t + 0.5)(rad)$ and $1.04\sin(\frac{\pi}{15}t + 0.4)(rad)$ respectively. We can see that the $P_{x\,actual}$ of the system is starting to converge to the $P_{x\,desire}$ after 0.7 second with some small and acceptable error.

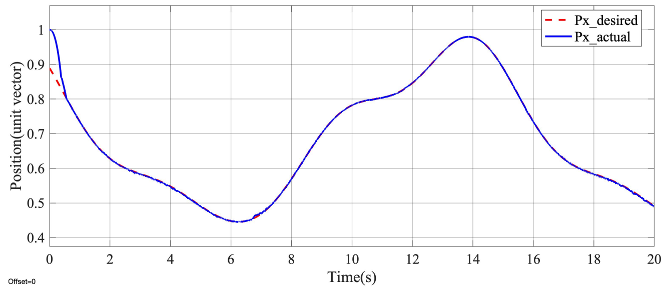

Fig. 8. Simulation result for position along x axis.

Fig. 9, the result of position $P_{y\,desire}$ and $P_{y\,actual}$ of the end-effector along y-axis is shown. The $P_{y\,actual}$ is start to converge to desired after 0.5s.

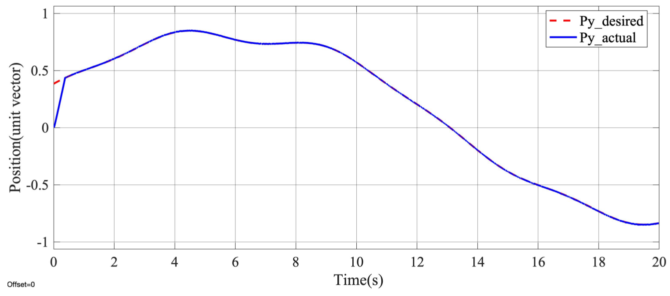

Fig. 9. Simulation result for position along y axis.

Similarly, the Fig. 10 shows the result of position $P_{z\ desire}$ and $P_{z\ actual}$ of the end-effector. The $P_{z\ actual}$ starts to follow desired after 0.6s.

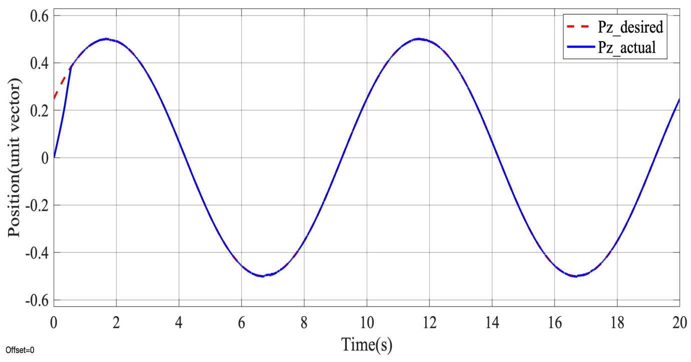

Fig. 10. Simulation result for position along z axis.

Moreover, Fig. 11 shows the result of end-effector in 3D space (P_x, P_y, P_z) which is the comparision between desired trajectory and response of the system. The desired path which is the red dash line, start at position $(P_{x\ desired} = 0.88, P_{y\ desired} = 0.38, P_{z\ desired} = 0.24)$ and finshed at $(P_{x\ desired} = 0.48, P_{y\ desired} = -0.83, P_{x\ desired} = 0.24)$. And the initial postion of gimbal starts from $(P_{x\ actual} = 1, P_{y\ actual} = 0, P_{z\ actual} = 0)$.

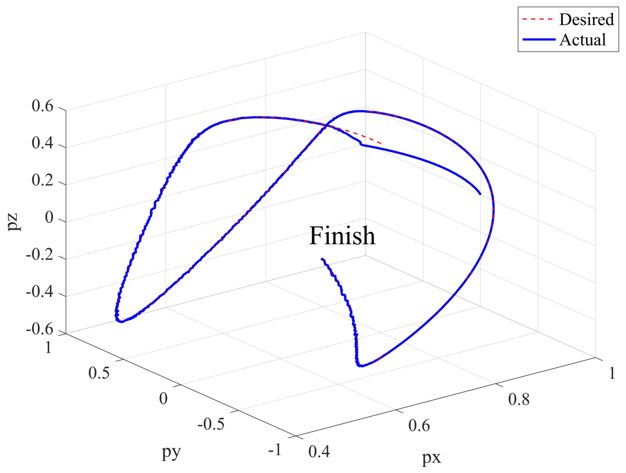

Fig. 11. Simulation result of positon of end-effector in 3D space.

Fig. 12. shows the result of desired tilt angle in unit degree and actual response from actuator. We can see that response is close to desired angle and follow the path to the end of simulation.

The error between desired and actual pose of the end-effector in each axis is shown in Fig. 13. Since the end-effector's starting position does not collide with the desired start position, it took roughly 0.5s for the robot to move toward the desired start point and follow along the desired path, achieving 0 error in all axes.

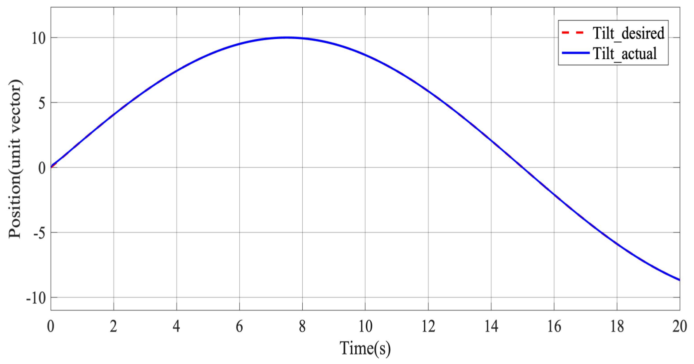

Fig. 12. Simulation result for tilt angle.

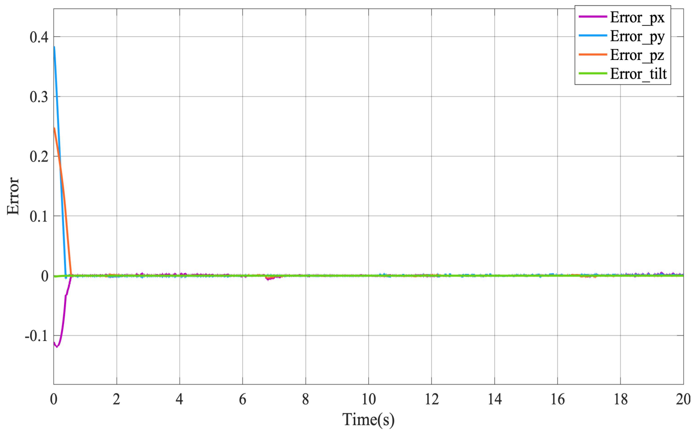

Fig. 13. Error between desired pose and actual pose.

4. CONCLUSION

This paper describes the development of a control framework for a 3-axis gimbal that can be easily integrated with another application via different platform. In ROS environment, we have created 4 nodes for sharing information with each other. We command desired point over ROS platform to control the 3-axis gimbal in Simulink. Moreover, this paper also includes the kinematic equations of the 3-axis gimbal, which are forward kinematics and inverse kinematics. We also illustrate feedback control of actuator which takes command from kinematic equations in order to control the end-effector to track desired pose. As the result of this framework, we can see that the end-effector of the gimbal tracked trajectory with small error that is acceptable in simulation. This result shows that the control framework and architecture can be implemented into real hardware for experiment. Therefore, for future work, another node will be created to publishes data to control actuators and data feedback of encoders for real hardware.

ACKNOWLEDGMENTS

The authors would like to thank members and alumni of Dynamics and Control Laboratory who have been offering their kind help and support throughout this work.

REFERENCES

[1] Rajesh, R. J., & Kavitha, P. (2016). Camera gimbal stabilization using conventional PID controller and evolutionary algorithms. IEEE International Conference on Computer Communication and Control, IC4 2015. https://doi.org/10.1109/IC4.2015.7375580

[2] Chitta, S., Marder-Eppstein, E., Meeussen, W., Pradeep, V., Rodríguez Tsouroukdissian, A., Bohren, J., Coleman, D., Magyar, B., Raiola, G., Lüdtke, M., & Fernandez Perdomo, E. (2017). ros_control: A generic and simple control framework for ROS. The Journal of Open Source Software, 2(20), 456. https://doi.org/10.21105/joss.00456

[3] Chivarov, S., Kopaeek, P., & Chivarov, N. (2019). Cost oriented humanoid robot communication with iot devices via MQTT and interaction with a smart home hub connected devices. IFAC-PapersOnLine, 52(25), 104–109. https://doi.org/10.1016/j.ifacol.2019.12.455