University of Texas at El Paso
ScholarWorks@UTEP

Open Access Theses & Dissertations

2020-01-01

Brian Valdez - Dynamics and Control of a 3-DOF Manipulator with Deep Learning Feedback

Brian Orlando Valdez
University of Texas at El Paso

Follow this and additional works at: https://scholarworks.utep.edu/open_etd

 Part of the Artificial Intelligence and Robotics Commons, and the Mechanical Engineering Commons

Recommended Citation

Valdez, Brian Orlando, "Brian Valdez - Dynamics and Control of a 3-DOF Manipulator with Deep Learning Feedback" (2020). *Open Access Theses & Dissertations*. 3200.
https://scholarworks.utep.edu/open_etd/3200

This is brought to you for free and open access by ScholarWorks@UTEP. It has been accepted for inclusion in Open Access Theses & Dissertations by an authorized administrator of ScholarWorks@UTEP. For more information, please contact lweber@utep.edu.

DYNAMICS AND CONTROL OF A 3-DOF MANIPULATOR

WITH DEEP LEARNING FEEDBACK

BRIAN O. VALDEZ

Master's Program in Mechanical Engineering

APPROVED:

Angel Flores-Abad, Ph.D., Chair

Ahsan R. Choudhuri, Ph.D., Co-Chair

Virgilio Gonzalez, Ph.D.

Joel Quintana, Ph.D.

Stephen L. Crites, Jr., Ph.D.
Dean of the Graduate School

DYNAMICS AND CONTROL OF A 3-DOF MANIPULATOR WITH DEEP LEARNING FEEDBACK

by

BRIAN O. VALDEZ

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Mechanical Engineering

THE UNIVERSITY OF TEXAS AT EL PASO

December 2020

Acknowledgements

I wouldn't be writing an acknowledgement statement if it weren't for UTEP's effort in accomplishing the mission of increasing access to excellent higher education. During my time in the university both as a graduate and as an undergraduate student I have been provided with opportunities that I wouldn't have had otherwise. It is one of the reasons why I moved back from Denver, CO after realizing that only a full-time job was not enough to afford college in the U.S. while living away from my family. I think many professors contribute to achieving the mission of UTEP and I feel thankful to them for their guidance throughout my academic journey with the university. Most professors commit their effort with us the students for extra time either after class or during office hours and one of them professors I will always remember is my thesis advisor professor Dr. Angel Flores-Abad. I appreciate his understanding and commitment with the mission of the institution as well as his dedication to guide and help students pursue their academic interests. I am thankful for having my former mentor Dr. Mike Everett introduce me to aerospace technology research work which along with my internship experiences has been a gift along my path thought higher education. It has been a unique experience to witness different management styles, leadership, and team's dynamics while working with the research center for almost 4 years. I got the opportunity to put my name in space thanks to a team's effort that I will treasure deeply, and thanks to the support from my family and friends throughout this academic journey. Particularly, to my mom Esmeralda who has been an instance of Aristotle's moral virtue definition to me. One of my mentors says "Luck is where preparation meets opportunity", and I feel luckily fulfilled for graduating with such preparation to meet the opportunities that I will encounter while following my passion of helping undeserved and underrepresented students and helping mankind pursue space exploration; the most beautiful expression of human ambition.

Abstract

With the ever-increasing demands in the space domain and accessibility to low-cost small satellite platforms for educational and scientific projects, efforts are being made in various technology capacities including robotics and artificial intelligence in microgravity. The MIRO Center for Space Exploration and Technology Research (cSETR) prepares the development of their second nanosatellite to launch to space and it is with that opportunity that a 3-DOF robotic arm is in development to be one of the payloads in the nanosatellite. Analyses, hardware implementation, and testing demonstrate a potential positive outcome from including the payload in the nanosatellite and a deep learning model that could leverage already existing satellite resources to function as a verification method to the robotic arm's activities.

Table of Contents

List of Figures

List of Equations

Chapter 1: Introduction

A nanosatellite is any satellite with mass from 1 kilogram to 10 kilograms and it is within this category that we find CubeSats which are cubic-shaped satellites developed to a specific number of cubic units that describe the total volume of the satellite e.g. 1 unit, 3 units, 6 units, etc. Naturally, CubeSats allow for low mass low-cost missions that have been helpful to test specific technologies developed from within the scientific community. One area of interest is the efforts that have been made to demonstrate advanced robotic manipulation capabilities on-orbit with a special interest in Low-Earth Orbit (LEO). More recently, CubeSat projects such as the Robotic Experimental Construction Satellite (RECS) [1] and the Intelligent Space Assembly Robot (ISAR) [2] propose multiple degrees of freedom robotic arm systems for on-orbit assembly, servicing, and space systems construction. These projects propose the use of 3U satellites to confine two extendable robotic arms, the required sensing instrumentation, and the on-board computers. It is important to highlight that systems of such complex dexterity occupy a minimum volume of 3U CubeSat units, that is approximately 10cm x 10cm x 30cm. A key capability of a manipulator for on-orbit servicing relies on its ability to estimate the position and orientation of the target relative to a known frame. This capability is critical to achieve successful manipulation which may be critical for a space systems mission. Some state-of-the-art approaches propose the use of neural networks for real-time pose, that is position and orientation, estimation of an object relative to the camera [3].

Chapter 2: Rationale and Merit

The National Aeronautics and Space Administration (NASA) envisions that robots will be leveraged for multiple tasks within the domains of human exploration, science exploration, and planetary capabilities, as stated in its 2020 Technology Taxonomy [4]. The research work presented in this paper describes the dynamics analysis, control design, and hardware implementation of a three degrees-of-freedom robotic manipulator and the use of a deep learning convolutional neural network as a verification method of the task at hand by the manipulator. Furthermore, the research envisions the implementation of the robotic arm system as a whole subsystem within a notional 6U CubeSat of an approximate volume of 10cm x 20cm x 30cm. The notional 6U CubeSat is not a hard requirement for the subsystem, but it allows to assume that the power requirements of the robotic arm are satisfied by an on-board electrical power system. This research work aligns with a 6U CubeSat project that is currently being developed by the MIRO Center for Space Exploration and Technology Research (cSETR), thus some aspects of interest throughout the methodology will visit the idea of how the robotic arm could affect the CubeSat. Likewise, the research work will describe how the robotic arm subsystem can leverage hardware already on board of the CubeSat.

Chapter 3: Methodology

3.1 Dynamics Analysis

The dynamics is concerned with the relationship between the motion of a mechanism and the forces that it generates. Within dynamics two different mechanisms can be considered:

- Forward dynamics: computes accelerations from given forces.
- Inverse dynamics: computes forces from given accelerations.

The rigid body assumption is a fundamental assumption with modeling dynamics of a system such as the robotic system presented as it assumes the rigid body remains undeformed despite its interactions with the environment or by any contact forces. Another important aspect this section will analyze is the kinematics of the system which is the analysis of motion without considering forces. Thus, only the geometric properties, and the position and degrees of freedom of the links of the system is necessary for the analysis. Similar to dynamics, kinematics has two different mechanisms: a forward kinematics mechanism that computes a location in the cartesian coordinate system from given joint positions of the links in the system, and an inverse kinematics mechanism that computes the opposite, that is, the combination of joint positions from a given position in the cartesian coordinate system. See Fig. 1.

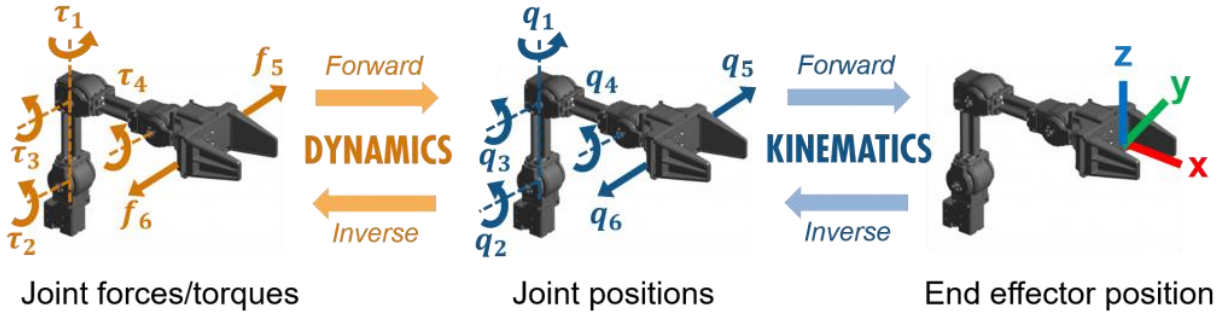

Figure 1: Mathworks.com

The focal point of the methodology is to control the robotic arm in the cartesian coordinate system thus the natural first step is to compute the inverse kinematics to obtain a combination of joint positions to satisfy the desired end effector motion. To do so, a simulation model uses a Transform Trajectory block which accepts two 4x4 homogeneous transformation matrices to compute a trajectory between the initial and the final state of the end effector over a time specified. The trajectory is computed using linear interpolation for the positions, and spherical linear interpolation (SLERP) for the rotations. A separate Inverse Kinematics block uses a rigid body tree configuration of the robotic arm, the previously computed 4x4 transformation matrix as the desired end effector pose, and its previous computation as an initial guess to calculate a vector of joint positions (*q1, q2, q3*) for the robot. The joint positions are computed using the Levenberg-Marquardt algorithm [5] which is an error-damped least-squares method optimize to converge faster if the initial guess is close to the solution. Thus, the decision of using the previous computed solution as the initial guess yielding to faster results.

Each element in the vector of joint positions is routed to a closed-loop controller, one for each actuator in the simulation model. Each close-loop path accepts one joint position as its desired value and uses feedback from a sensor to compute an output as torque to actuate the motors. Multiple sensors in the simulation measure both linear and angular positions, velocities, and accelerations of the motors and the links for the purpose of calculating the reactions at the base of the robotic system. An open-source modeling toolkit for mobile-base robotic multibody systems (SPART)[6] calculates the kinematics and dynamics of the system given the current and the previous position of the three joints computed at each time step. The SPART simulation tool uses the velocities and accelerations of the base and joints to compute the inverse dynamics and output a vector of 6 elements: 3 elements to describe the reaction forces at the base and 3

elements to describe the torques created by the motion of the robotic arm. The vector can be represented with $\mathbf{f}_r$ and $\boldsymbol{\tau}_r$ for the forces and torques, respectively. Newton's second law of motion is used to verify the results of the SPART dynamics simulation tool. Eq. 1 represents the reaction forces and torques induced by the robotic arm on the base where it would be mounted.

$$\begin{bmatrix} \mathbf{f}_r \\ \boldsymbol{\tau}_r \end{bmatrix} = \begin{bmatrix} -\sum_{i=1}^{n} m_i \dot{\mathbf{v}}_i \\ -\sum_{i=1}^{n} \left(\mathbf{I}_{ci} \dot{\boldsymbol{\omega}}_i + (\mathbf{r}_i - \mathbf{a}_0) \times m_i \dot{\mathbf{v}}_i \right) \end{bmatrix} \tag{1}$$

Where $\mathbf{f}_r \in R^3$ and $\boldsymbol{\tau}_r \in R^3$. *n* loops over the total number of links in the system (three), *m* is the mass of the link *i*, $\dot{\boldsymbol{v}}$ is the linear acceleration at the link's center of mass (COM), $\boldsymbol{I}$ is the inertia tensor of the link, $\dot{\boldsymbol{\omega}}$ is the angular acceleration of the link, $\mathbf{r}$ is the distance in R^3 from the link's COM to the inertial frame, and $\mathbf{a}_0$ is an offset constant distance in R^3 from the inertial frame to the base-link COM. Fig. 2 shows a visual representation of the elements described in Eq. 1. Notice that for the robotic arm system proposed in this work three links pertain to the links and one for the so-called base-link which is fixed with the inertial frame.

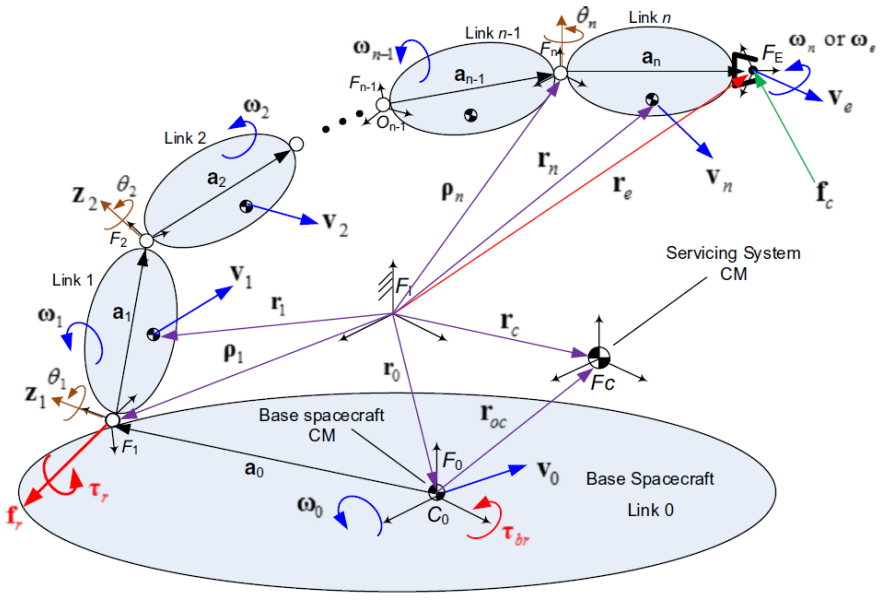

Figure 2: Multibody dynamic system of a servicing spacecraft with an onboard manipulator [7]

Fig. 3 portraits a notional trajectory of 2.5 seconds emulating the initial deployment of the robotic arm. This trajectory has been generated from obtaining an initial and a final 4x4 homogeneous transformation matrix from the computer-aided design; one for the initial stowed configuration of the robotic arm and one for a deployed configuration.

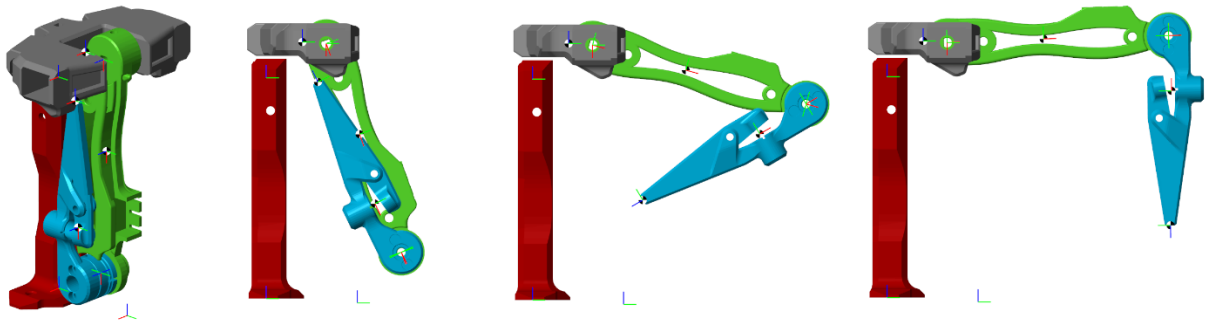

Figure 3: Simulation deploying the robotic arm

Such trajectory is simulated and Eq. 1 gets populated from the outputs of the sensors implemented in the simulation model, and it calculates the resulting reaction forces and torques for every time step at the base where the robot would be attached to. Fig. 4 compares the results of the SPART simulation tool and the calculations from Newton's second law.

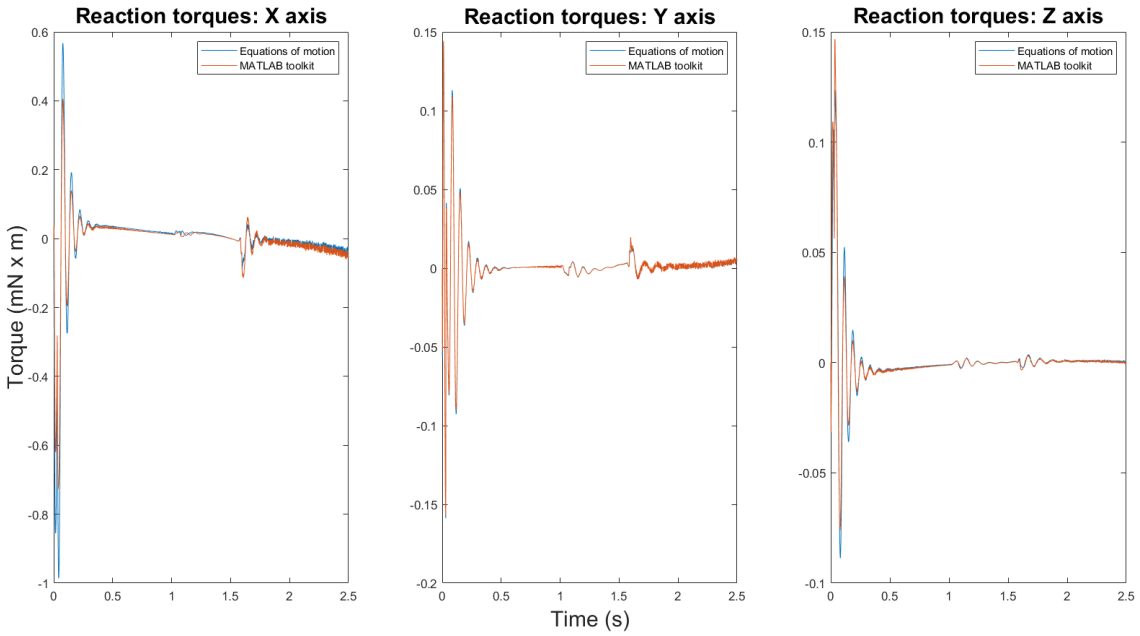

Figure 4: Comparison of reaction torque results

3.2 Control Design

The controller to drive the motors of the system implements an empirical method developed experimentally with the hardware implemented. The design of the controller uses an approach of reaching local waypoints along the trajectory instead of calculating an output from the initial to the final waypoint of the trajectory. An important aspect to keep in mind with the proposed approach of reaching local waypoints is that moving in relatively small steps along the trajectory could cause sudden stops if a smooth behavior is not achieved. Sudden stops cause steep slopes in the velocity versus time curve which also cause the acceleration to flip signs along the trajectory. This is an undesirable behavior because adding unwanted vibration to the system could cause failures such as cables disconnecting, hardware failure, or resonance failure.

As for the computed output, the controller has the same structure as a typical PID controller as shown in Fig. 5:

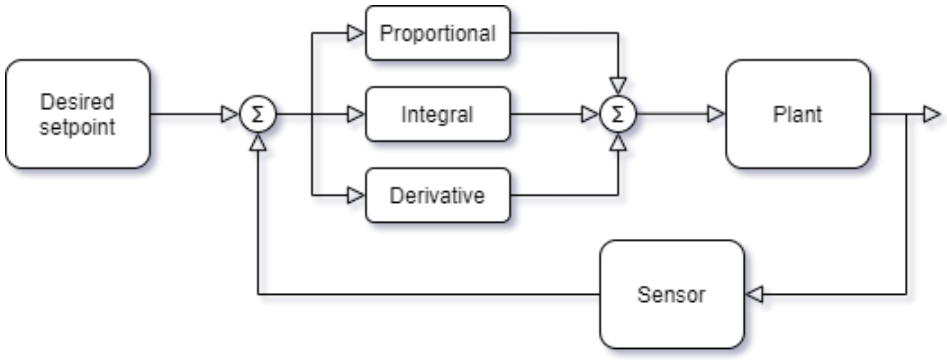

Figure 5: Closed-loop feedback system

The controller implements a function that keeps track of the previous errors calculated in the closed-loop system and determines whether a specific motor has continuous motion or not. In the case that a specific motor is deemed to have enough angular velocity to stay in motion, the algorithm makes the decision of keeping the integral part of the controller constant to the last known value. As a result, the output of the controller is mainly driven by the integral part.

The decision-making flowchart in Fig. 6 describes the logic implemented for the integral part of the controller that influences directly the Pulse-Width Modulation (PWM) output of the controller:

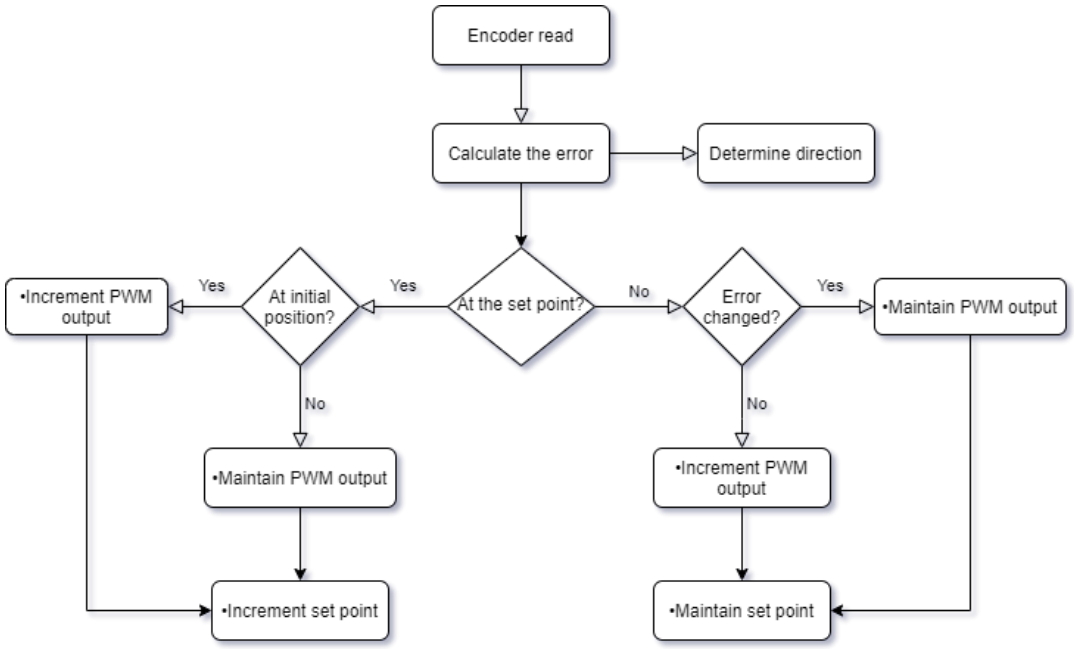

Figure 6: Controller logic for incrementing its output

The first step in the decision tree is to calculate the difference between the desired setpoint for the motor and the reading from the encoder to determine the direction in which the motor needs to rotate. Then, the question of whether the motor is at a desired setpoint splits the diagram in two sections. The section on the left determines that if the motor is at the first setpoint the output of the controller must be incremented, but if the opposite happens then it means the motor is at a setpoint along the trajectory and it potentially has enough angular velocity to stay in motion thus the output stays at a constant value. The section on the right explores two possibilities: if the measured error is different from the previous error an angular velocity exists and the controller's output stays at a constant value whereas if the measured and the previous error are identical the controller continues its calculation of the pulse width modulation signal to move the motor to the desired setpoint. It is important to notice that the decision tree applies separately for every motor

implemented in the system. Lastly, the proposed dynamic controller design does not define a setpoint as a single angular position but rather as a range given by the desired setpoint of the motor plus and minus a tolerance value.

Three test runs of the controller implemented in the hardware are plotted in Fig. 7 comparing the angular position of the three motors with the desired position acquired from the simulated model.

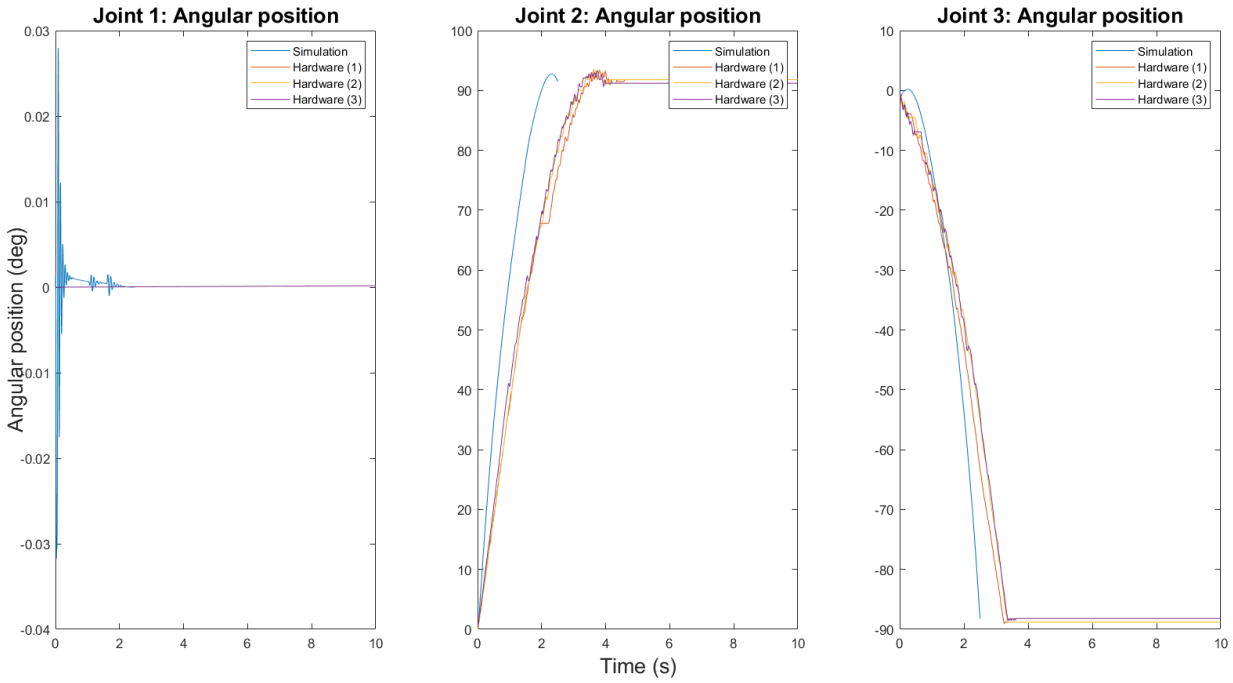

Figure 7: Simulation vs. Hardware testing results

The plots compare the outputs computed from the dynamics analysis and later implemented to a simulation model that calculates the reaction torques previously shown in Fig. 4. The solid blue line draws the expected angular position from initial time zero to the final trajectory time at 2.5 seconds. However, with the control implemented in the hardware the results show that the controller drives the motor to the expected final position with an average error of less than 0.4° and it achieves the final position at a slightly lower velocity which in practice is a more desirable outcome.

3.3 Hardware Implementation

A physical additive manufactured prototype of the robotic arm has been designed to implement the proposed control design. The prototype consists of three links each one with one degree of freedom in the form of a revolute joint, and a base-link with zero degrees of freedom that remains always attached to the base as shown in Fig. 8 and Fig. 9.

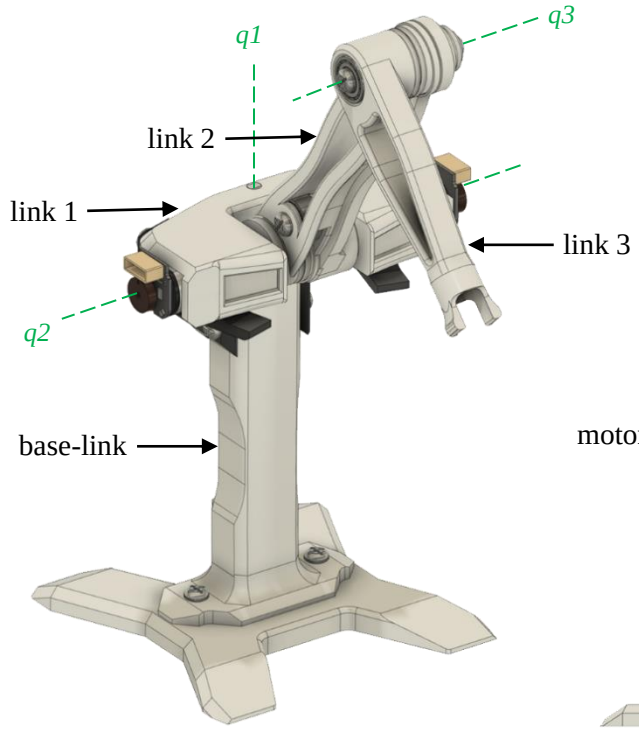

Figure 8: Robotic arm CAD model

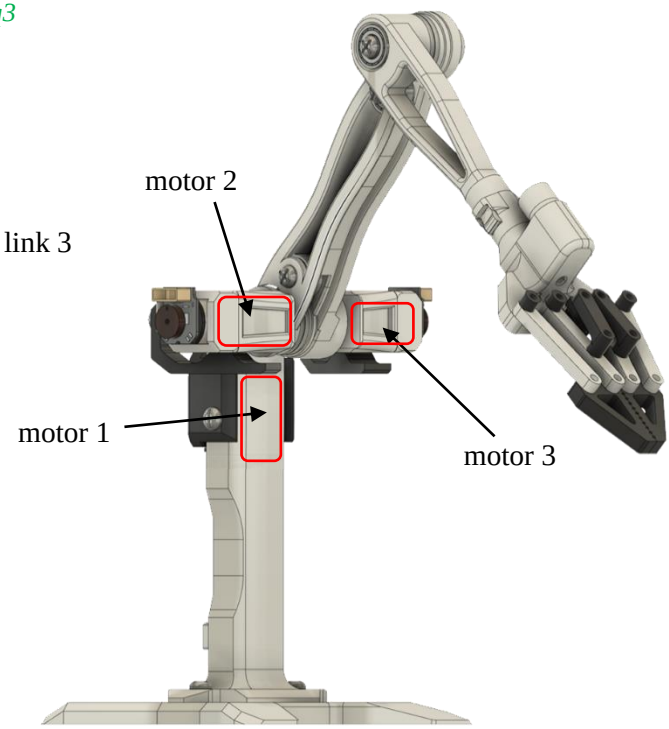

Figure 9: Robotic arm CAD model with a different end effector

Different designs for the end effector have been explored to implement the ability to autonomously change its end effector tool depending on the task at hand. A key feature to the design is the location of the motor to actuate link 3. The motor is inserted into a bushing that has a wire wrapped around it and is connected to another bushing that rotates link 3. Such implementation could for future iterations to be of a modular configuration allowing multiple links to be connected increasing the workspace of the robot.

The hardware implemented in the robotic arm structure runs in a loop with a microprocessor unit and motor drivers to distribute signals. As discussed during section 2.1 Overview, the 6U CubeSat project has manifested a Jetson Nano microprocessor and an Electrical Power System as included payloads. It is under that assumption that the robotic arm subsystem leverages the two capabilities to operate as illustrated in Fig. 10.

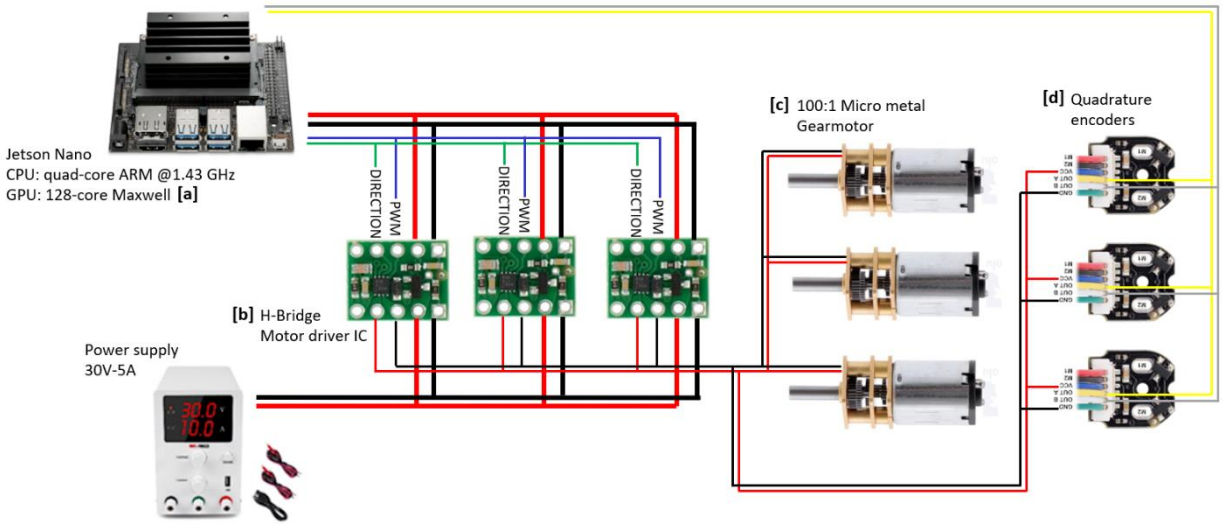

Figure 10: Signal diagram Level 1

A board equipped with a Linux kernel [a] runs the programming code that controls the robot over general-purpose input/output (GPIO) pins and powers the drivers [b] of the motors. At the same time, the drivers receive a higher power from a separate power supply that matches the specification of the motors [c]. The motors have a gearbox with a ratio of 100.37:1. Each motor actuates one joint of the robot, and it has a quadrature encoder [d] attached to provide a feedback signal back to the microprocessor. The quadrature encoders count 12 steps per revolution. Thus, combined with a gear ratio of 100.37:1 the encoders count 1204.44 steps per every revolution of the motor's shaft which conversely it translates to a resolution of ±0.2988°, the equivalent to one step of the encoder.

3.4 Deep Learning Feedback

This section will describe the set up for the training and testing results of a convolutional neural network (CNN) using the Keras library in Python to solve a binary classification problem. A CNN is a type of deep learning network, commonly used to analyze imagery data, that systematically applies filters to an image to extract features from the image such as lines, contours, corners, etc. Thus, a series of hidden layers in the network are required to perform convolutions, pooling, and apply activation functions to the data.

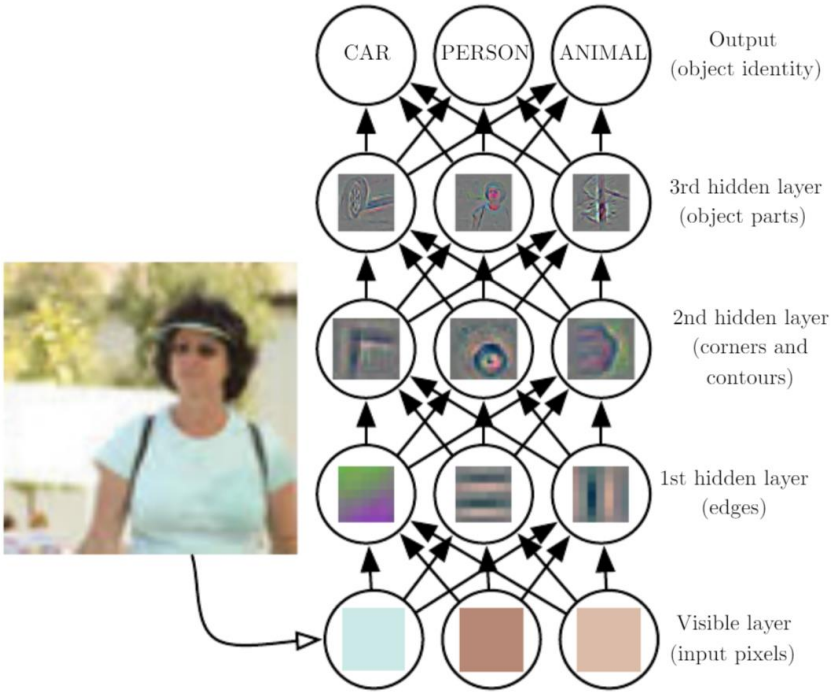

Figure 11: Illustration of a Deep Learning model [8]

The deep learning feedback system proposed in this research could solve the problem of correctly classifying when the robotic arm successfully captures an object to manipulate. Moreover, the feedback system makes use of already existing hardware in the 6U CubeSat to implement the algorithm and it prevents the downside of altering the system by adding more elements that would increase the mass, power consumption, and overall complexity of the

CubeSat design. The feedback system uses a Sony IMX-219 imager embedded to a Raspberry Pi Camera Module v2 connected through a ribbon connector to the Jetson Nano microprocessor as illustrated in Fig. 12.

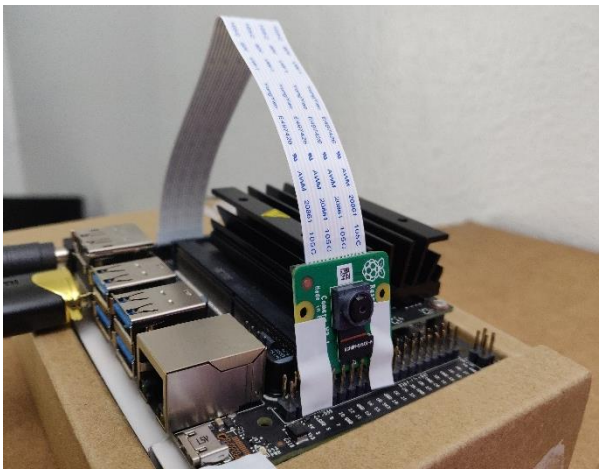

Figure 12: Camera setup for testing

The image dataset for the CNN contains two subfolders: true and false. Each subfolder contains image files for each classification category: **'true'** contains jpg images of the robot successfully grabbing the tool and is represented with the number 1. **'false'** contains jpg images of the robot unsuccessfully or attempting to grab the tool while the tool is in the frame and is represented with the number 0.

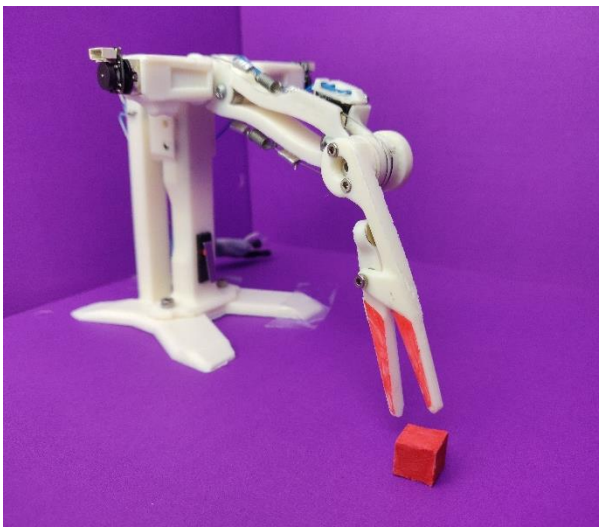

Figure 13: Protype attempting to grab an object

It is a good practice to run a code that deletes corrupted images when working with large datasets but this dataset has no corrupted images. The dataset images were obtained from running an automated Python script that creates a camera object and captures a burst of images while the robotic arm is manipulated by hand to capture different pose configurations as seen in Fig. 14.

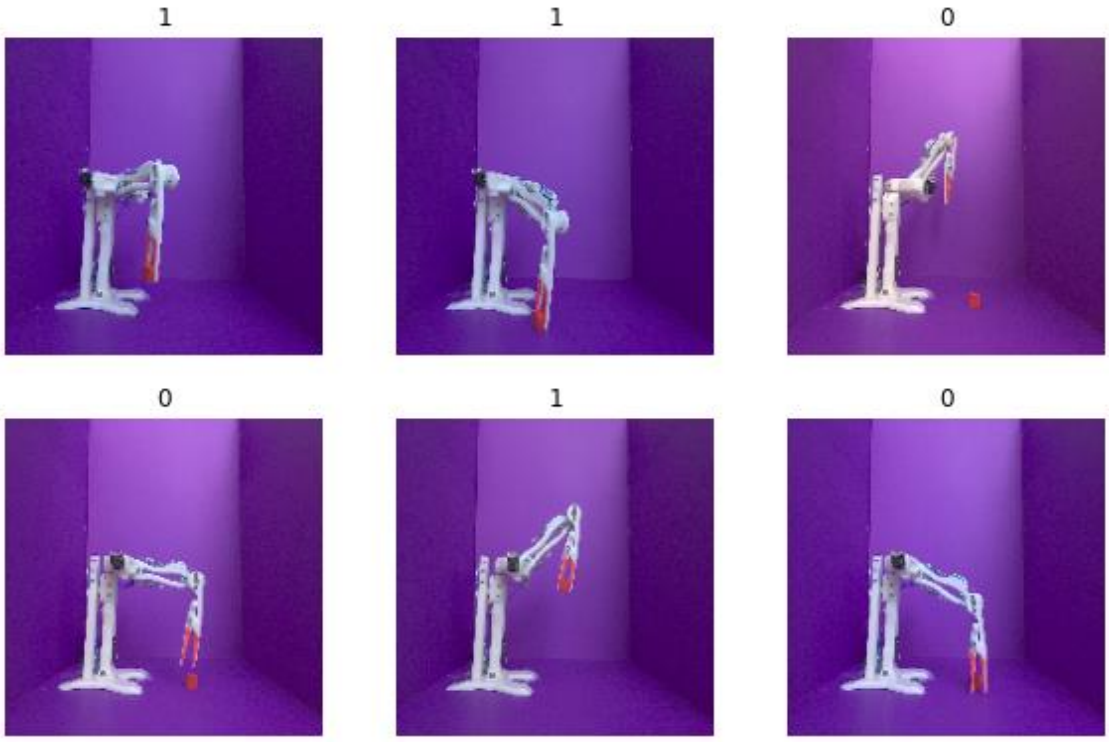

Figure 14: Images labeled for dataset

The dataset obtained consists of 726 images where an 80% / 20% split is used for training and validation, respectively. The images obtained from the camera are 1640x1232 pixels in size which are scaled to 180x180 pixels in RGB channels, thus the matrix form and vectorized representation of each jpg image is 180x180x3 and 97,200 pixels values, respectively. The resizing of the images allow room for data loss or for an inaccurate representation of the pixel data and there are different interpolation algorithms that can be used for scaling such as nearest neighbor, bilinear, cubic, lanczos3, lanczos5, etc. Bilinear will be used as it is desired set boundaries of the objects in the frame mainly by its color. The bilinear interpolation algorithm

allows to get information from the neighboring 2x2 pixels, and its output is not affected by other pixels farther away e.g., Cubic interpolation takes the input from 16 neighboring pixels.

Another step desired during preprocessing is the addition of random realistic transformations to the dataset. A random horizontal flip is a layer added to mirror the image horizontally, and a random rotation layer rotates the images within a predefined range 0° to 36° either clockwise or counterclockwise randomly as illustrated in Fig. 15.

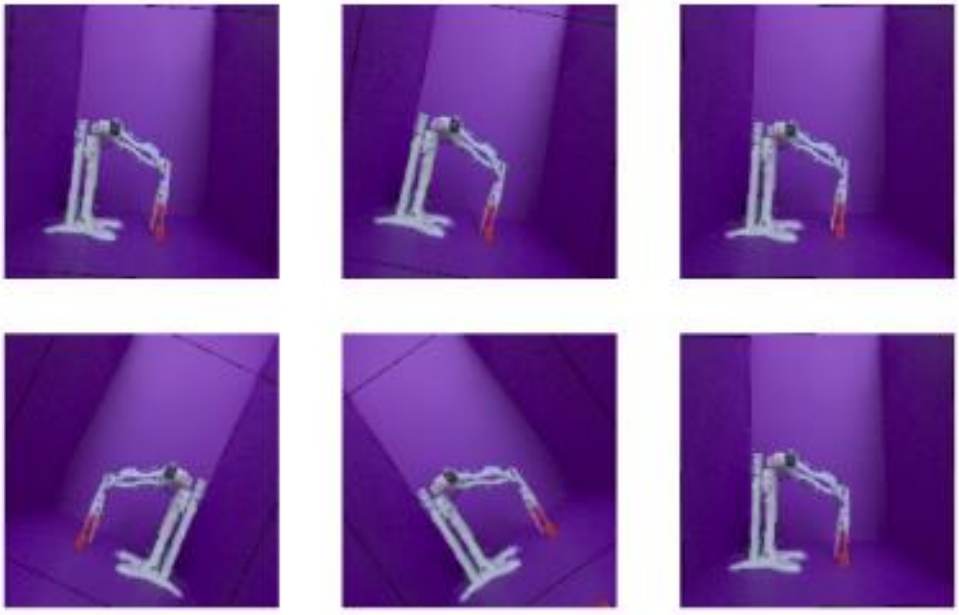

Figure 15: Image transformations applied to the dataset

Keras has a feature that shuffles the data in alphanumeric order and it is a desired feature for the training of the network due to the method that was used to obtain the images. Another step implemented during data preprocessing is a typical approach used in CNN that is of normalizing the pixel data that ranges from 0 to 255 to floating point values ranging from 0 to 1. This normalizing layer is the first layer implemented in the model.

The following layers in the model mimic the entry flow of the Xception network [9], which has a top-1 accuracy of 0.79 and top-5 accuracy of 0.945 on the ImageNet dataset. Fig. 16 shows the stack of layers in the network used for training, validation, and testing.

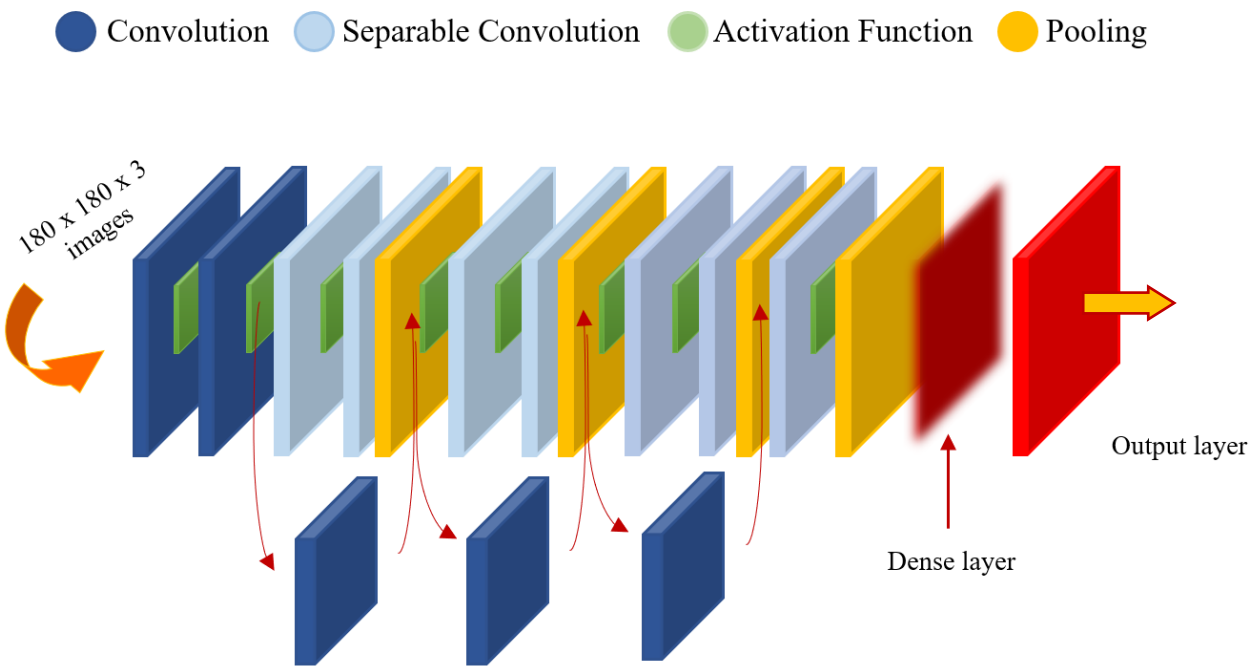

Figure 16: Convolutional Neural Network architecture

However, the optimizer the network will use is Adam which is a stochastic gradient descent method derived from adaptive moment estimation. The learning rate hyperparameter will be the one Keras defaults to as 0.001 and the number of epochs to train according to the results from previous training iterations will be 50 epochs. Because we are looking at making a classification between two subsets, a probabilistic loss is more accurate to the problem. Thus, the binary cross-entropy loss function is used to output a floating-point prediction of the output belonging to the class. The metric to judge the performance of the model is binary accuracy which computes the frequency with which the predicted output matches the true value. Metrics are similar to loss functions, except that the results from the metric are not used when training the model.

After capturing a burst of images to create the dataset, 10 images were excluded from the process meaning that the network never sees them during training. These 10 images are labeled as a 'twin' dataset and are used for testing. However, the 'twin test' images are very similar to the images used for training as they were obtained from the same burst of images. For this reason,

another test set labeled ‘blind’ dataset is made of 30 images which are obtained from another burst of images with different configurations of the robotic arm. Fig. 17 displays the overall breakdown of the images used for training, validation, and testing.

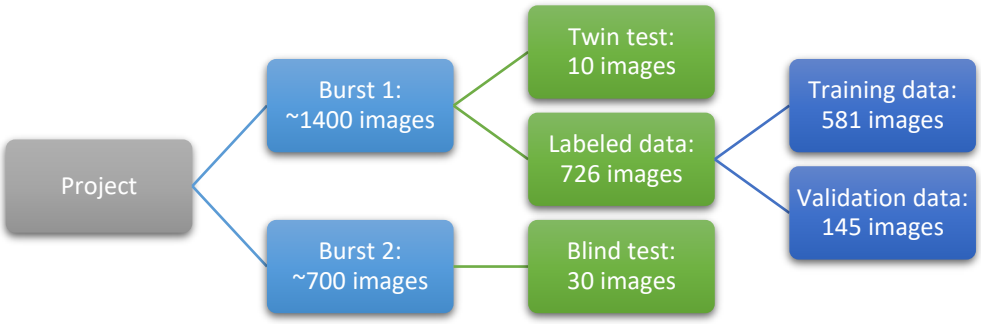

Figure 17: Allocation of images from two separate burst of images

The network reaches a training and validation loss in the $x10^{-4}$ after 50 epochs with accuracies of 1.0000 for both training and validation.

The following results in Fig. 18 are obtained after running the 10 images from the ‘twin’ dataset through the network:

```
* * * * *  Twin dataset  * * * * *
Image 026 is 0.02 percent false and 99.98 percent true.
Image 029 is 100.00 percent false and 0.00 percent true.
Image 120 is 0.01 percent false and 99.99 percent true.
Image 171 is 100.00 percent false and 0.00 percent true.
Image 504 is 100.00 percent false and 0.00 percent true.
Image 725 is 0.00 percent false and 100.00 percent true.
Image 796 is 0.10 percent false and 99.90 percent true.
Image 1040 is 100.00 percent false and 0.00 percent true.
Image 1326 is 0.05 percent false and 99.95 percent true.
Image 1483 is 100.00 percent false and 0.00 percent true.
```

Figure 18: Predictions from ‘twin’ test

The F_1-score described in Eq. 2 is a statistical analysis to measure the accuracy of a test's accuracy.

$$F_1\text{−score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \qquad (2)$$

The test results from the 'twin' set yield a F_1-score of 1.00. Lastly, Fig. 19 shows the results after running the 30 images from the 'blind' test through the network:

```
* * * * *  Blind dataset  * * * * *
Image 082 is 0.51 percent false and 99.49 percent true.
Image 085 is 0.31 percent false and 99.69 percent true.
Image 112 is 66.36 percent false and 33.64 percent true.
Image 121 is 71.33 percent false and 28.67 percent true.
Image 140 is 16.80 percent false and 83.20 percent true.
Image 142 is 19.35 percent false and 80.65 percent true.
Image 165 is 0.08 percent false and 99.92 percent true.
Image 167 is 0.28 percent false and 99.72 percent true.
Image 210 is 0.06 percent false and 99.94 percent true.
Image 212 is 0.06 percent false and 99.94 percent true.
Image 233 is 66.72 percent false and 33.28 percent true.
Image 234 is 28.79 percent false and 71.21 percent true.
Image 263 is 0.06 percent false and 99.94 percent true.
Image 283 is 0.27 percent false and 99.73 percent true.
Image 285 is 0.63 percent false and 99.37 percent true.
Image 364 is 100.00 percent false and 0.00 percent true.
Image 365 is 100.00 percent false and 0.00 percent true.
Image 366 is 100.00 percent false and 0.00 percent true.
Image 390 is 100.00 percent false and 0.00 percent true.
Image 391 is 100.00 percent false and 0.00 percent true.
Image 420 is 99.52 percent false and 0.48 percent true.
Image 424 is 99.90 percent false and 0.10 percent true.
Image 447 is 100.00 percent false and 0.00 percent true.
Image 485 is 96.40 percent false and 3.60 percent true.
Image 504 is 100.00 percent false and 0.00 percent true.
Image 506 is 100.00 percent false and 0.00 percent true.
Image 534 is 96.37 percent false and 3.63 percent true.
Image 536 is 98.15 percent false and 1.85 percent true.
Image 565 is 97.09 percent false and 2.91 percent true.
Image 581 is 52.68 percent false and 47.32 percent true.
```

Figure 19: Predictions from 'blind' test

The test results from the 'blind' test yield a F_1-score of 0.86.

Chapter 4: Summary

4.1 Discussion

As mentioned during the problem overview, a case study for a potential implementation of the robotic system consists of the fully integrated system operating within a 6U CubeSat orbiting in Low-Earth Orbit (LEO). Thus, the assumption of a free-floating object applies as it assumes conservation of linear and angular momentum. The free-floating case is modeled in a graphical programming environment to obtain the torques caused by the robotic arm by deploying for 2.5 seconds. See Fig. 20. The simulation assumes the CubeSat, as a free-floating object, only has 3-DOF and no other disturbance torques are considered other than the effect of the robotic arm's motion.

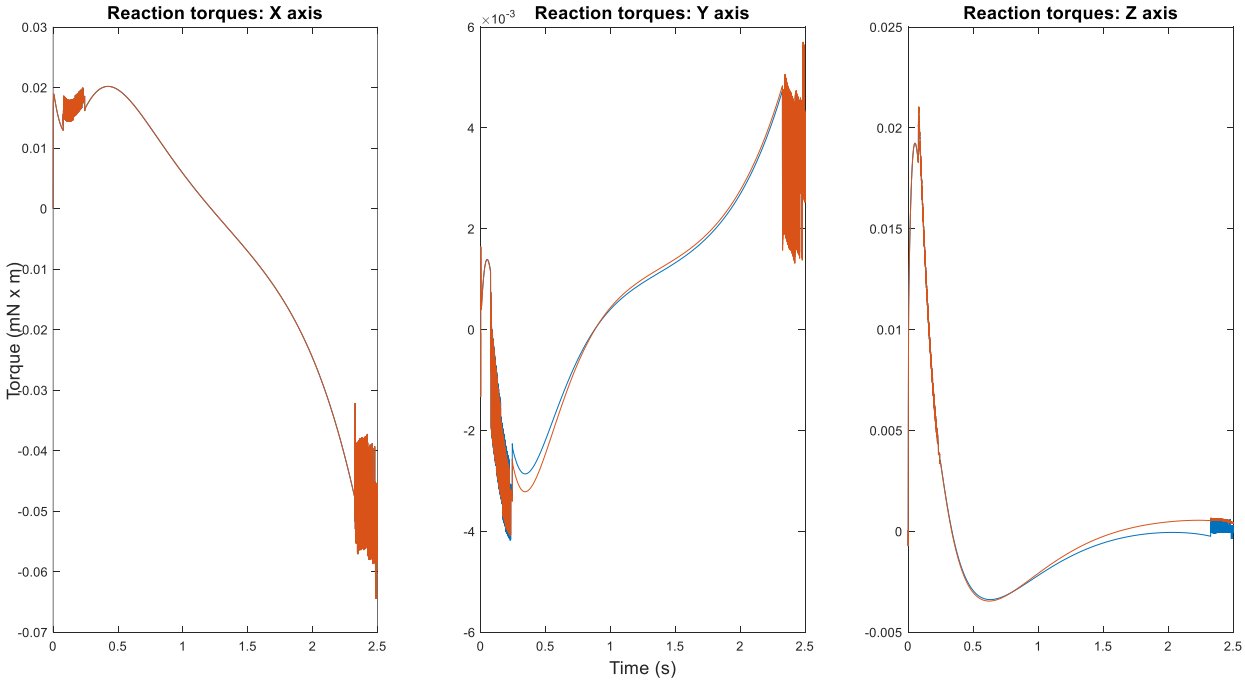

Figure 20: CubeSat torques originated from the motion of the robotic arm payload

Although the expected torques are within the range of what an average nanosatellite attitude control subsystem can achieve, other solutions could be explored to minimize the expected

torques. If the electrical power system (EPS) can provide enough power, the three motors in the robotic arm could be substituted for motors with a higher gear ratio to provide higher torques and slower angular velocities. In addition, the motion of the motors could benefit from a more thorough design of the controller or a trajectory planning that yields a smoother motion between waypoints in the trajectory. It is important to highlight that the simulation in Fig. 20 includes the torques created by the motion of the link but it also includes the reactions forces reacting at the base the robotic arm is attached to. The base is assumed to be at 7.5 cm in the X-axis of the CubeSat's frame. The other two reaction forces are estimated to cause no effect as torques when the base strategically aligns with the Y-axis and Z-axis. Thus, a strategic positioning of the robotic arm within the CubeSat could further reduce the torques presented in the graph. Furthermore, the simulation assumes a uniformly distributed mass of 12kg with a center of mass that aligns with the geometric center of the structure. Although 12kg is the maximum allowable mass for a 6U CubeSat and it is a reasonable starting point for the simulation, it is expected by the team that the final mass of the CubeSat would be only a fraction of the assumed mass thus resulting in further smaller torque values. Lastly, an additional factor of safety in how the motion of the robotic arm affects the CubeSat, although not calculated, results from the difference between the simulated results and the true response from hardware as demonstrated in Fig. 7.

4.2 Conclusion

The simulation that calculates the effect the robotic arm has on the 6U CubeSat makes conservative assumptions and the results should be refined through an iterative process of calculations as more information becomes available from the CubeSat team. However, with the implementation of the control design and hardware described in the methodology, potentially better outcomes will result from defining the assumptions stated in the discussion. The controller design has potential for increasing the performance if a quadrature encoder can provide higher resolution than the calculated ±0.2988 degrees. The factor of safety discovered from hardware testing could be calculated from running a hardware-in-the-loop test and compare an updated profile from Fig. 20 and the true reactions of deploying the arm inside a 6U CubeSat prototype. The payload requirements state a success criteria for the accuracy of the end effector to be of ±1mm in the cartesian coordinate system. With the joint positions shown in Fig. 20, the final distance between the expected position from the simulation and the actual position obtained from the hardware testing is calculated using Eq. 3.

$$d(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}. \quad (3)$$

Resulting in a distance of 1.083 mm, for a random test, which is estimated to be a repeatable test result given the final joint positions shown in Fig. 20.

The positive results obtained from the deep learning feedback concluding a 0.86 F_1-score for the most representative test conditions could be further improved by iterating with minimal changes in the model yet increasing the number of images in the dataset. In its current state, the CNN model could be trained to identify when the robotic arm is successfully grabbing a tool if a different tool or end effector is selected for flight. Although the training and testing of the model benefited from having a solid color background and favorable lighting, such conditions could be

emulated within the CubeSat envelope with low power light emitting diodes. The python model could be included in the flight microprocessor, use the camera to capture an image(s), and run it through the model to verify a successful task completion. Ultimately, the architecture of the network could be more robust and richer in training data to be able to classify multiple objects for on-orbit activities and a pre-designed variety of end effectors to perform such activities.

References

[1] Ansley N. Knight, Trent R. Tetterton, Alec J. Engl, Peter M. Sinkovitz, Bailee J. Ward, Jin S. Kang. 2020. “Design and Development of On-orbit Servicing CubeSat-class Satellite,” *Proceedings of the Small Satellite Conference*. https://digitalcommons.usu.edu/smallsat/2020/all2020/124/.

[2] John M. Gregory, Jin S. Kang, Michael Sanders, Dakota Wenberg. 2019. “Characterization of Semi-autonomous On-orbit Assembly CubeSat Constellation,” *Proceedings of the Small Satellite Conference*. https://digitalcommons.usu.edu/smallsat/2019/all2019/30/

[3] Sonawani, Shubham & Alimo, Ryan & Detry, Renaud & Jeong, Daniel & Hess, Andrew & Ben Amor, Heni. (2020). Assistive Relative Pose Estimation for On-orbit Assembly using Convolutional Neural Networks.

[4] 2020 NASA Technology Taxonomy. (2020, October 27). Retrieved November 11, 2020, from https://www.nasa.gov/offices/oct/taxonomy/index.html.

[5] Sugihara, Tomomichi. "Solvability-Unconcerned Inverse Kinematics by the Levenberg–Marquardt Method." *IEEE Transactions on Robotics* Vol. 27, No. 5 (2011): 984–91. doi:10.1109/tro.2011.2148230.

[6] J. Virgili-Llop et al., “SPART: an open-source modeling and control toolkit for mobile-base robotic multibody systems with kinematic tree topologies,” https://github.com/NPS-SRL/SPART.

[7] Flores Abad, Angel & Zhang, Lin & Wei, Zheng & Ma, Ou. (2016). Optimal Capture of a Tumbling Object in Orbit Using a Space Manipulator. Journal of Intelligent & Robotic Systems. 86. 1-13. 10.1007/s10846-016-0417-1.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. Deep Learning. The MIT Press.

[9] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 1800-1807, doi: 10.1109/CVPR.2017.195.

Curriculum Vita

Brian O. Valdez obtained a Bachelor of Science in Mechanical Engineering in December 2018 with academic experiences in national and international projects such as a Multidisciplinary Faculty-Led Study Abroad Program, a 1U CubeSat project, NASA's 2018 Breakthrough, Innovative & Game-changing Idea Challenge, and a 12U Robotic CubeSat mission research. Throughout his academic studies Brian carried out different roles such as front-end developer, integration and testing assistant, mechanical designer, and kinematics computational analyst as well as commitments as a member and tutor for the Pi Tau Sigma Mechanical Engineering Honor Society and Chi Epsilon Civil Engineering Honor Society.

Outside the academia Brian has had industry experience at the MIRO Center for Space Exploration and Technology Research working with multiple teams as well as assisting in different capacities of Orbital Factory II, a nanosatellite that was launched aboard the Cygnus NG-12 mission to the International Space Station in November 2019. Within the government sector Brian has interned three instances at The National Aeronautics and Space Administration in the capacities of structural dynamics, optics testing, and control systems and design where Brian worked in the development of a CIF technology.

Other non-academical efforts in STEAM engagement can be found in the Youtube's channel "DAME MI ESPACIO", at www.linkedin.com/in/bovaldez or via email at bovaldez@miners.utep.edu.