WORCESTER POLYTECHNIC INSTITUTE

ROBOTICS ENGINEERING PROGRAM

Using Airbrushes to Paint a Sphere with an ABB Robotic arm:

SUBMITTED BY:
Connor Craigie
Peter Guelmino

Date Submitted : December 6th, 2019
Date Completed: December 6th, 2019
Course Instructor: Dr. Craig Putnam

Executive Summary

Our project explored the use of industrial robotics in painting three dimensional objects. The ABB robot was used to paint a two dimensional image onto a three dimensional spherical surface. Using a mapping technique we generated colored models of the Earth. A foam sphere was mounted on a PID controlled rotary table with defined work object in the robots workspace. End of arm tooling was developed that could paint up to 4 different colors depending on what color was needed for the given data point. The robot pose for the desired data point was communicated to the robot through TCP communication running from a MATLAB script. External actuation was integrated with the system through serial communication. These external actuators both actuated the air brushes in the end of arm tooling, and rotated the sphere on a rotary table to easily access the entire surface. This project met all of its goals and produced easily identifiable globes.

Table of Contents

Executive Summary **2**

Table of Contents **3**

Introduction **4**

Goal Statement **5**

Task Specification **5**

Workspace Setup **5**

End of Arm Tooling Design **6**

Programming **8**

Results and Discussions **11**

Conclusions **12**

Introduction

Robots are commonly used for painting in the industry, with applications ranging from painting cars to wooden furniture. Because of this, a project was chosen that involved painting with the ABB arm. After deciding to paint with the robot arm, the painting method was chosen through research and following restrictions from the lab management. The final decision was to use an airbrush in order to paint our desired image.

The next step was to decide the actual image that will be painted, as well as what will be painted on. Originally, a flat surface was considered, but it was decided to move to a three dimensional surface in order to take advantage of the robots capabilities. The simplest three dimensional surface is a sphere, so it was chosen as our base surface, and then an image was required to paint onto this sphere. The earth was chosen as the desired image because it can be distilled down to two simple colors, and is easily recognizable.

After deciding the target image and surface, a data structure was generated in order to parse the image data and generate color values for individual points, as well as determining the rotation of each of these points on the sphere of a given radius. The goal is to pass these values to the ABB robot arm so it will move to the desired point around the sphere, and then spray the appropriate color onto the sphere. After generating this data structure, a color system needed to be chosen, and the chosen format was CMYK values, because this is the color format used for standard printing, so it may be usable for the robotic arm.

The end of arm tooling attached to the robot will need to be designed to actually paint the desired image onto the sphere. Since the chosen painting method is an airbrush, an airbrush was purchased to examine how it is actuated. An airbrush has two forms of actuation, one that controls the airflow, and one that controls the area of the paint spray. Because of limitations, only one degree of freedom could be controlled, so the end of arm tooling was designed to actuate from off to fully on at the widest setting. While designing the end of arm tooling for the robot, there was a concern about the workspace of the robot, and how it would reach all parts of the sphere. To address this problem, a rotary table was designed that would allow the foam

sphere to spin along a single access, which greatly increases the effective workspace of the robotic arm because it no longer has to reach around the object to paint all sides.

The final question that was addressed was how all of this data would be communicated with the robot. It was decided that a MATLAB script would act as a central hub, and it would communicate with the robot arm through TCP, and the rotary table and end of arm tooling through serial communication. The end of arm tooling and rotary table are on a different voltage and communication network because they cannot handle the 24 volt logic of the PLC.

Goal Statement

The goal of this project is to program an ABB 1600 robotic arm to use multiple airbrushes to paint the earth onto a foam sphere of a given radius.

Task Specification

This project is expected to do the following:

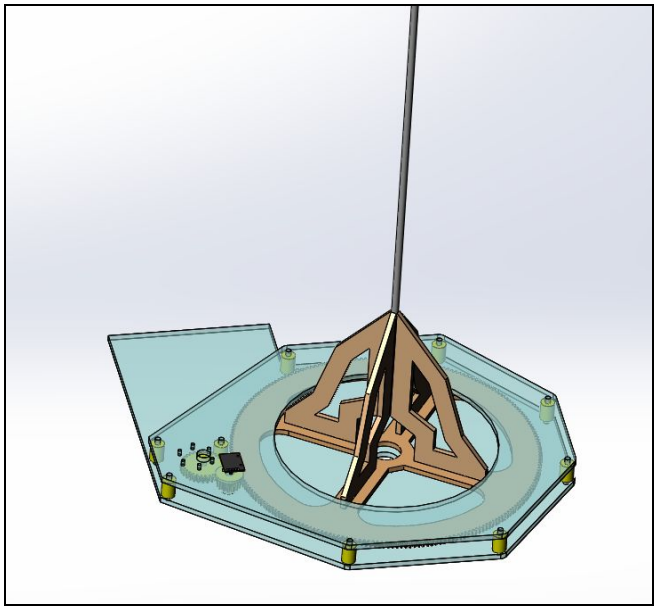
- Take an uploaded image of the earth and generate color values as well as positions for a set number of points on the sphere.
- Send these points and color values to the ABB robot so it can move to the required position to paint the sphere.
- Communicate to the rotary table so it can rotate to the desired angle to aid the robot arm.
- Effectively paint the desired colors in such a way that the image is clearly understood.

Workspace Setup

In addition to the MATLAB script we have been developing mechanical solutions to painting 360 degrees around a spherical object. Our solution is to develop a rotating table. The table will be driven using a DC motor and quadrature encoder. The DC motor is attached to a gear system with an idling gear fixed to the quadrature encoder. Using an ESP32 a PID controller has already been pre programmed to deal with the current hardware. The rotary table

will communicate with a laptop running the primary MATLAB script. The communication will be through a single SPI communication line. It will indicate the angular displacement of the rotary table as a 12 bit Integer value.

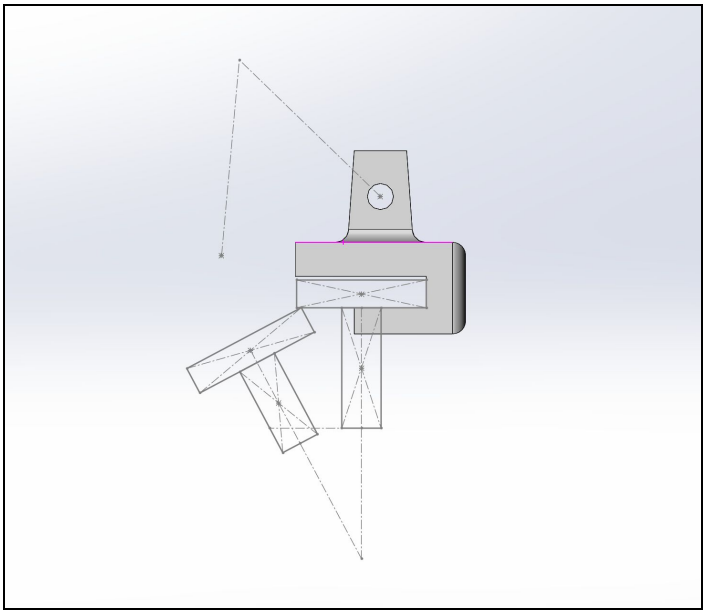
The design has been fully developed in Solidworks. This week the primary rotary table components will be cut from quarter inch plywood. Secondary components such as the drive and idling gears will be 3D printed on a personal machine. The team has scheduled manufacturing time this wednesday in washburn shops. A photo of the current design is depicted below.



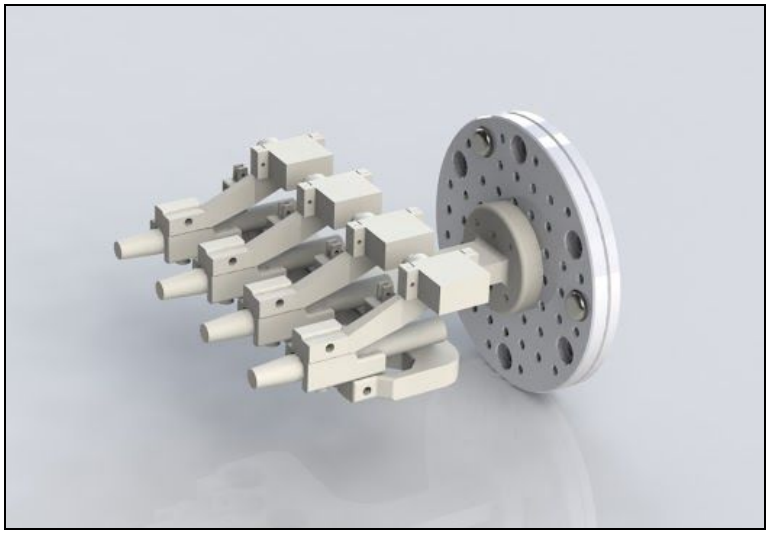
End of Arm Tooling Design

Knowing that the chosen color format is CMYK, four airbrushed are required, one for each of those colors. These airbrushes need to be individually actuated, and as mentioned in the introduction, they will be controlled with one servo that will reduce the degrees of freedom from two to one, the location for the servo to actuate the airbrush was chosen from a linkage synthesis shown in the following image. The two positions of the airbrush were measured on the physical airbrush, and those dimensions were used in the solidworks sketch to generate the desired

positions. From manual tests, there was no concern for torque requirements on the servos for actuating the airbrush.



Once the servo location was found, a holder was designed to hold the airbrush, making sure clearances were provided for the air hose and paint holder. Four of these servo holders were assembled into the full end of arm tooling with a tool holder that will provide enough clearance for the ends of the airbrushes. This tooling was mounted to a provided plate that was pre-tapped with holes that the airbrush holder would mount to. This full end of arm tooling is shown in the image below.



All components of the end of arm tooling were 3d printed. Initial prints were made to test tolerancing, and some of the tolerances were too small, so larger tolerances were made and the final prints were produced. No compliance was necessary in the tooling because the air brushes would not make contact with any part of the environment, they would only be held at a set distance away from the foam sphere. Overall, the EOAT successfully accomplished its goals.

Programming

Using mathematical projection, our team has used matlab to generate a dataset of coordinate points directly related to a RGB color value. The following plots prove the projection is accurate enough for painting purposes.

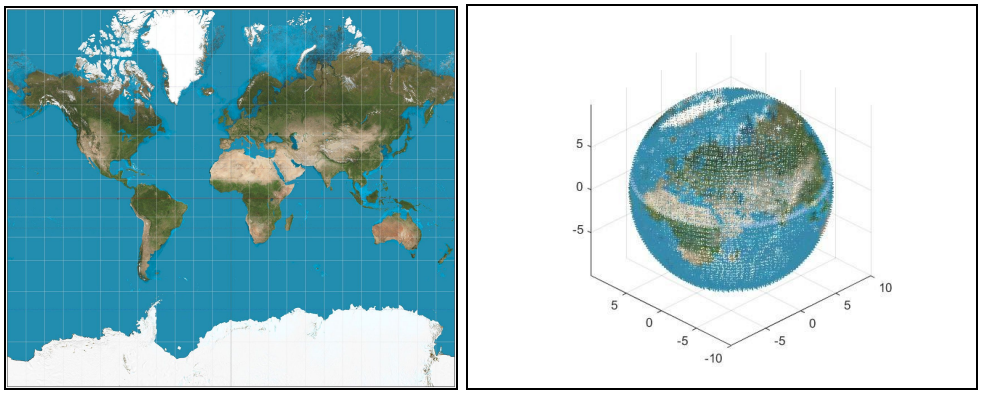


Figure 1: Comparison between input 2D image and projected 3D sphere

The data structure's composition is an "n X 6" matrix of values. Using a constant which defines points per inch, the sphere is defined by evenly distributed point values. The projection is a construction of many horizontally oriented cross sections. As the spheres radii changes along the z-axis, the number of points projected on the surface is calculated dependant on the circumference of the immediate cross section. This is easily calculated with our set parameter of points per inch. The number of vertical cross sections is also calculated as one half the product of circumference and the points per inch constant.

The data set is constructed in a very linear way to allow for the generation of trajectory planning. By iterating down the list of coordinate locations, the data set develops a natural progression from point to point. Where each cross sectional circumference begins and ends at the same line of latitude. This can bee seen as a stepping effect in the plot below.

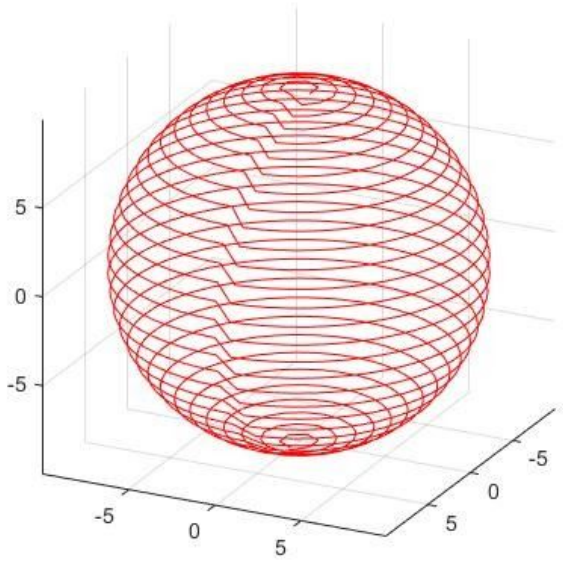


Figure 2: Connected dataset iterations from 1 to N

The system uses a variety of communication protocols. Matlab interfaces with the end of arm tooling, the turntable, and the painting end of arm tooling. Communicating with the end of arm tooling required the construction of an arduino object in Matlab. This object uses serial communication through a logically high level library which controls the servo actuators of the paint sprayers.

The external 7th axis turntable is also communicated through an onboard serial port. However, the turntable used an ESP32 microcontroller rather than a standard arduino board. This required us to generate our own communication protocol to represent an angular displacement sent from matlab. This low level logic was developed around the transmission of 8-bit characters. Due to the nature of serial communication, characters representing the integer setpoint were sent

in reverse order from the central Matlab script. The communication line only handled integer setpoints to eliminate the complexity of writing floating point data.

Communication with the robot was done using TCP data packets. The data packet was built as an array of six 8-bit characters. The first transmitted bit acted as a boolean which represented the running state of the robot. The second bit was a discrete horizontal theta displacement. Once again floating bits were avoided to simplify communication protocol. This theta displacement ranged from 0 to 180 giving us the potential for 180 discrete discrete steps for painting. The remaining four values represented the analog spray values for cyan, magenta, yellow, and black. These values were on a scale from 0 to 255. Although the analog values were sent to the robot, they were never used to modify spraying position as intended. The analog control was done in the form of a time delay when actuating the end of arm tooling. The rapid code still uses logic to check whether the bit received is lacking of value. If a character is received as zero the robot will immediately continue the program. This saves a considerable amount of run time. With this internal check, the robot can assure itself that a given coordinate frame will not be in use during a spray cycle.

TCP communication was later revised to handle only blue and green elements as it provided higher quality painting effects. In this case, only two of the four CMYK bits were used. In matlab, the green and blue values were directly compared. If the analog value for blue was larger than green, blue would be painted and vice versa. The third condition was if the summation of analog blue and green was less than fifty percent of 255, no color would be sprayed with the assumption that that cell is primarily white.

On the RAPID side of the code the first step was to define the work space and work object for the rotary table, then objects were generated for the end of arm tooling, and finally a program structure was generated to take in the values from the TCP communication and control the position of the EOAT. The workspace was defined with a single work object that would represent the bottom right corner of the rotary table, and a robot target was generated that represents the top of the all thread rod that holds the foam sphere. This top target will be used to

generate a target at the center of the sphere by translating it along the z-axis by a value determined by the radius

After the work object was generated the tool objects needed to be generated for each of the four airbrushes. This was done by importing the solidworks assembly into RAPID, and a reference frame was assigned to the end of each tool. One reference frame was assigned for each color: cyan, magenta, yellow, and black.

The main RAPID code was relatively simple, the first step was to open a socket with the MATLAB code, this socket would transmit a data packet that contained values for which color to use, and what angle along the sphere the robot should move to. The data would also contain a value to tell the robot when to stop painting. Once the robot get the value for the color, it will enter a nested reltool for the specific tool. This nested reltool would first rotate the tool around the center target, and then translate the tool along the tool z axis to the radius of the sphere, plus an offset. At this point, the tooling is in the correct position, and the robot will send a confirmation packet to the MATLAB script, which will actuate the servos and spray a desired color.

Results and Discussions

As expected, CMYK subtractive color printing is challenging in practice. Colors did not mix as intended to produce the required results. Multiple tests were run with the CMYK color spectrum. The first test was just CMY, fearing that the black component would potentially over power the combination of the other three. The CMY spectrum produced relatively accurate results. Continents on the painted globe were relatively distinguishable even on the small scale low resolution sphere. The lack of black coloration forced the landmasses to shift primarily yellow in color. With the addition of black coloration it became clear that it would be too overpowering. The calculated analog value for the black component was scaled down multiple times to achieve a different effect, however no scale of the black threshold was able to produce the correct colorization.

The secondary option was a blue green threshold system. This bi-color implementation worked considerably well. With a larger diameter foam sphere the resolution of the painting was

greatly increased. Continents became easily distinguishable. It was confirmed that by increasing the diameter of a sphere, a much higher resolution image can be made. In further implementation larger spheres would be used and the points per inch constant would be increased to produce more crisp images.

In the future, there are many changes that could be made, one of the biggest improvements that could be made is to machine the tool out of metal so that it can have better tolerancing and will not flex as much when actuated. A cage would also be made to hold the paint cans to the airbrush, this was previously done with zip ties. The servos could also be updated to utilize two servos which would allow for actuating both degrees of freedom of the airbrush.

In terms of expanding the project, analog control could be added to the servos instead of purely time based binary control. The tooling could also be redesigned to mix colors before spraying them, this would allow the CMYK method of painting to work better. The communication system could also be updated to utilize the PLC on the robot station, however this would require managing the 24 volt logic on the PLC with the control system for the end of arm tooling.

This project could also be expanded with continuous path planning, instead of painting point by point. This would rapidly increase the speed at which the spheres could be painted, although there may be issues with resolution. An automatic loader and unloader could also be

implemented to change out the spheres in the system. This would increase the speed and throughput of the system.

Conclusions

This project enforced and enhanced the team's understanding of multiple robotic disciplines. The application of Robotstudio simulation and work object frame definition became important in the success of the project. The course covered these important topics well and prepared the team considerably to perform the task of painting a spherical surface. The openness and flexibility of the final project allowed teams to develop creative solutions for creative problems. With a self defined project, some problems exist which could be considered out of the scope of the course. This ambiguity allows engineers to develop problem solving skills along with the sill of technical implementation. Teams have the ability to apply knowledge beyond the scope of the course or learn something new. For example, out implementation required us to research communication protocol for both TCP and serial communication. In short, it could be said that this exercise was one of the most valuable the team has experienced here at WPI.