ILR04

11/13/15

TEAM F: ADD_IN

Dan Berman
Astha Prasad
Ihsane Debbache
Nikhil Baheti

Individual Progress

Since the last ILR presentation I have focused on formulating our path planning algorithm and implementing a proof of concept simulation of it. Early in the week of November 2nd, Astha and I met and created an overall concept for the path planning algorithm. The basic steps for the algorithm are as follows:

1. Orient the printer nozzle to be normal (or at some specified angle) to the velocity trajectory of the nozzle at all times (this yields two possible solutions for nozzle orientation).
2. Always orient the nozzle to be pointing towards the nearest COTS item (this selects between the two solutions in (1)).

In this design the path planning algorithm simplifies each COTS item to a 'keep out' zone centered at a specific point location and with a specified height. The assumption is made that the G-Code file for the printed part is designed to accurately represent the exterior profile of the COTS item, and thus no other information is needed about the COTS item's external geometry. Not relying on additional geometry information implies that the part designer avoided creating a design which violate the geometry constraints of the printer (i.e. parts are too close together, or have concave profiles that the nozzle can not access), since these potential collisions will not be detected by the path planning algorithm.

The path planning algorithm was implemented in MATLAB and is applied to a given G-Code file. The inputs to the algorithm are a G-Code file produced by Slic3r which encodes the XYZ locations the print nozzle must move to, and a list of COTS item locations and heights. Using this information, the algorithm computes the necessary angle of the rotation axis for each G-Code movement command, and produces a modified G-Code file. It was discovered that Slic3r only produces straight line commands (G00 and G01) which greatly simplifies the implementation algorithm because rotation along an arc doesn't need to be considered.

The actual implementation of the algorithm is as follows:

For each G00 or G01 (straight line movement) command:

1. Compute the vector ($\vec{n}$) in the XY plane from the current position to the center point of each COTS item ($\vec{c}_i$) which is at or below the current Z position

$$\vec{n}_i = \vec{c}_i - \overrightarrow{p_1}$$

2. Determine the COTS item which is nearest to the current position.

$$\vec{n} = min_i(\|\vec{n}_i\|)$$

3. Compute the vector for the current movement command

$$\overrightarrow{m} = \vec{c}_i - \overrightarrow{p_2}, \quad where,$$
$$\overrightarrow{p_2}\ is\ the\ desired\ (x,y)\ position\ of\ the\ extruder\ at\ the\ end\ of\ the\ movement\ command$$

4. Determine which side of the print path the COTS item is on

$$side = det([\vec{m}, \vec{n}]) \quad [1]$$

$$\begin{gathered} where\ for \\ side > 0:\ COTS\ item\ is\ to\ the\ left\ of\ \vec{m} \\ side = 0:\ COTS\ item\ is\ on\ \vec{m} \\ side < 0:\ COTS\ item\ is\ to\ the\ right\ of\ \vec{m} \end{gathered}$$

5. Compute the velocity of the nozzle in the coordinates of the print bed

$$\vec{v} = \frac{\overrightarrow{p_2} - \overrightarrow{p_1}}{\|\overrightarrow{p_2}\|\|\overrightarrow{p_1}\|}$$

6. Compute the angle between the velocity vector and the x-axis

$$\alpha = atan2\left(v_y, v_x\right), where\ v_y, v_x\ are\ the\ x, y\ components\ of\ \vec{v}$$

7. Based on which side of $\vec{m}$ the COTS item is, rotate the nozzle to be normal to the velocity vector and pointing towards the COTS item.

$$\begin{gathered} side > 0:\ r = \alpha + \frac{\pi}{2} \\ side = 0:\ Invalid - Collision\ will\ occur \\ side < 0:\ r = \alpha - \frac{\pi}{2} \end{gathered}$$

To evaluate the algorithm MATLAB scripts were written both to implement the algorithm and to plot the resulting G-Code file. A simple part (cylindrical tube) was designed and sliced. The algorithm was then applied on the resulting G-Code file with a COTS item positioned in the center of the cylinder and the result was visualized as shown in Figure 1 and Figure 2.

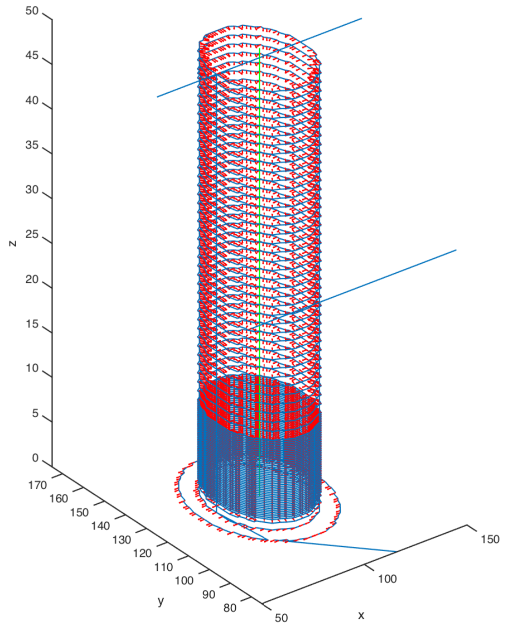

Figure 1: Plot of modified G-Code including R-axis commands. A COTS item (represented by the green line) was specified to be at the center of the cylinder.

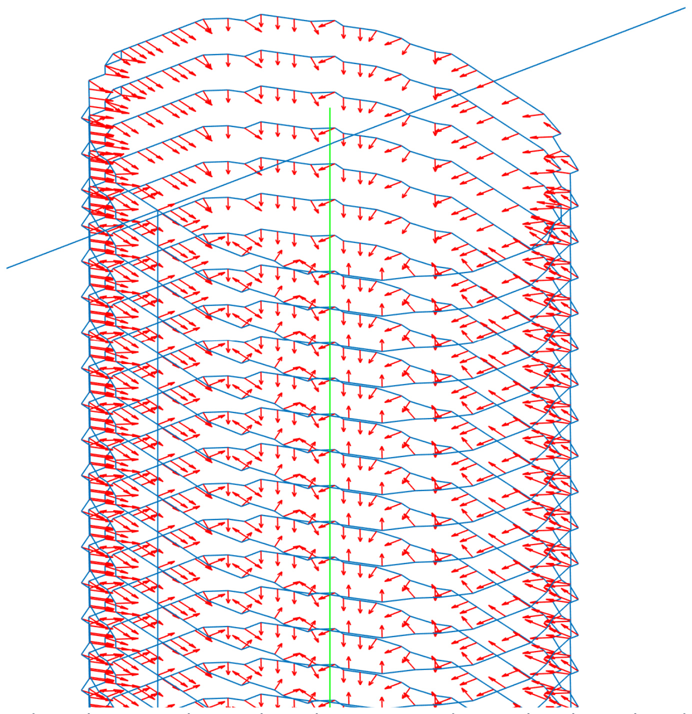

Figure 2: Close up of G-Code path. Notice the nozzle is always oriented normal to the path and towards the COTS item.

In addition to my primary work on the path planning algorithm, I also assisted Nikhil and Ihsane by machining our first nozzle design. Neither Nikhil nor Ihsane have shop experience, and since Ihsane has not yet completed the shop class I worked with Chuck Whittaker to get access to the RI shop, permission to use the lathe/mill, and machined the nozzle.

Challenges

The primary challenge was deriving the mathematical formulas to implement the path planning algorithm. Although conceptually the algorithm is simple, my initial implementation relied heavily on inverse trigonometric functions which due to range limitations would provide incorrect results when the extruder nozzle was in certain Cartesian quadrants relative to the COTS item. To solve this issue, I vectorized the equations, used equation [1] to determine which side of the path the COTS item was on, instead of my original approach which relied on computing the angle between the COTS item and nozzle trajectory.

Plotting the G-Code files also presented a challenge, since they typically contain $10^4 – 10^5$ movement commands which requires excessive memory usage to plot. To alleviate this, I modified the plotting algorithm to compute the distance between each movement command, and only generate a point on the plot when the distance of successive movement commands is greater than a defined threshold. Using a threshold of 0.1 mm reduced the number of plotted points from $\sim10^5$ to $\sim10^3$, producing much more manageable plots which still accurately portrayed the part.

Teamwork

Nikhil Baheti: Nikhil completed the nozzle design and produced a drawing which I used to machine the heat block (primary component in our custom nozzle). After the nozzle was machined, he worked with Ihsane to assemble and test it. In addition, Nikhil created a flowchart to help himself get up to speed on the layout of the 3D printer's firmware and then modified it to accept an additional G-Code command to control an additional stepper motor. Nikhil also completed the board layout and gerber file generation for our PCB.

Ihsane Debbache: Worked with Nikhil testing our custom nozzle and identifying changes to be made for the next design iteration. He did extensive research to select our slip ring and rotary stage, ultimately coming to the realization that we could meet our specifications for the rotation joint using a hollow-shaft stepper motor rather than a geared rotary stage. The hollow shaft stepper motor is much smaller than comparable rotary stages and (surprisingly), a similar price. In addition, Ihsane handled all purchasing for the team, including procurement of parts for our PCB design.

Astha Prasad: Worked with me on the conceptual development of the path planning algorithm, and refined the insertion layer selection algorithm to plot the input *.stl* file and display the insertion layer as a plane. She also took over from Nikhil as our project manager, taking responsibility for organizing our team meetings and ensuring we are meeting all deadlines and requirements.

Future Plans

The implementation of the path planning algorithm helped to verify our conceptual design, but there are numerous details that still need to be resolved before it can generate printable 4-axis G-Code. The two primary issues are as follows.

1. Direction of R-axis rotation:

   Currently, the algorithm analyzes each movement command individually and computes the rotation angle of the R joint in absolute coordinates. Because of this, the algorithm currently does not specify which direction the joint needs to rotate to reach each new angle, which could lead to a collision with the COTS item. To solve this, an additional step needs to be added to the path planning algorithm to determine which rotation direction maintains the nozzle orientation to always be pointing towards the COTS item.

2. Collision during long moves:

   Currently, the R axis angle is only calculated based on the distance to the nearest COTS item at the beginning of the movement command. In the case where a long movement command bringing the nozzle from far away up close to the COTS item, or a movement command passes near one COTS item and continues to a more distant location, there exists a possibility for the nozzle to collide during the movement. To remedy this, the path planning algorithm needs to be extended to compute the distance to the nearest COTS item at multiple locations along each movement command.

In addition to the the primary issues, there also exists numerous opportunities to improve the path planning algorithm to be more efficient (it currently takes ~45 seconds to parse ~30,000 movement commands) and more robust (it occasionally produces logical errors such as appending the r axis movement command into the comment section of a G-Code command).