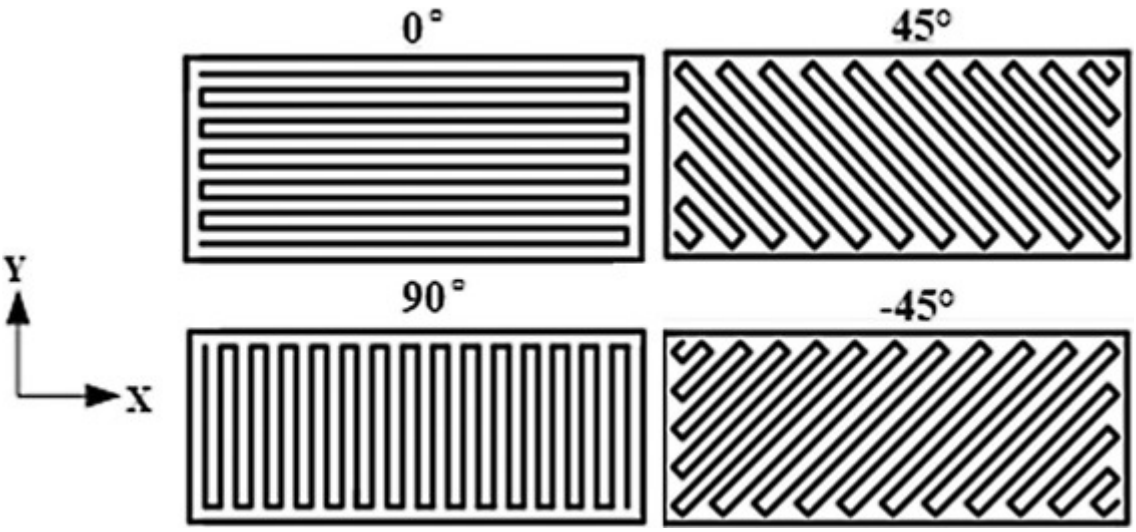
Slicer Documentation

Overview

An infill is a series of interior walls for maintaining the structure of the object while it gets printed layer by layer.

A Slicer program usually translates a 3D model into G-Code instructions, but because our 3D printer has some unique criteria and constraints, we must write our own.

Previous senior design groups provided us with various `.gcode` files to work with, but they all use diagonal rasterization, whereas Dr. Bigelow has asked us to do horizontal and vertical rasterization only:



This website: https://nraynaud.github.io/webgcode/ lets us visualize where the motors are moving the laser step by step.

`GCodeParser.cs` shows us that the previous SD groups used 3 custom defined `M` codes:

```
// Custom defined M codes for our 3D printer:
// M200: Execute layer change
// M201: Laser on
// M202: Laser off
```

The primary gcode type they use is the `G1` code, which tells the motors to move the laser at a constant velocity to the coordinates.

For example, this `.gcode` starts the laser at (0,0), turns it on, draws a line to (0,1), then turns it off.

```
G1 X0.0000 Y0.0000
M201
G1 X0.0000 Y1.0000
M202
```

Directories

All slicer files live in the directory `stl_to_gcode/Slicer/`.

```
stl_to_gcode
|
└──Slicer
        config.json
        gen_cube.m
        gen_voxel.m
        gen_voxel_90_degrees.m
        insert_defect.m
        slicer_ctrl.m
        slicer_gui.mlapp
```

Configuration

The `config.json` file is used to pass cube parameters to the `slicer_ctrl.m` file. Parameters in this file are listed bellow.

`filename`: specifies filename path for the gcode file

`length`: total length of the cube in cm

`width`: total width of the cube in cm

`height`: total height of the cueb in cm

`layer_height`: the height of each layer in the cube in cm

`voxels_per_length`: amount of voxels in each layer by length

`voxels_per_width`: amount of voxels in each layer by width

`voxel_margin`: the margin between voxels in cm

`voxel_padding`: the padding beteen voxels in cm

`defects`: 2D array of defects where each defect is an array coordinate points that describe where each defect is in the cube. If any of the arrays have -1 as a value in the array, that defect array is ignored

`defect_x_origin`: the origin x coordinate point of the defect

`defect_y_origin`: the origin y coordinate point of the defect

`defect_z_origin`: the origin z coordinate point of the defect

`defect_width`: the defect's total width

`defect_length`: the defect's total length

`defect_height`: the defect's total height

Source Code

gen_cube.m

Main function that generates the gcode file. Creates the overall cube and calls other functions to insert the voxels.

Does not insert defects (that happens after this function if needed).

gen_voxel.m

Helper function to the gen_cube.m file that inserts 0 degree rasterization pattern voxels into the cube.

gen_voxels_90_degrees.m

Helper function to the gen_cube.m file that inserts 90 degree rasterization pattern voxels into the cube.

insert_defect.m

Helper function in slicer_ctrl.m and slicer_gui.mlapp files that insert defects into a gcode file and overwrites the output file in the process.

slicer_ctrl.m

Master function that accepts a configuration JSON file and calls gen_cube.m file and insert_defect.m file if needed. The default configuration is located in `/stl_to_gcode/Slicer/config.json`, but any config location can be specified.

One of two ways (better way) to generate a gcode file of the cube.

slicer_gui.mlapp

MatLab GUI where you can manually input cube parameters and generate a gcode file of the cube. One downside of this method is that it can only insert one defect in the cube, not multiple.

References:

https://www.sciencedirect.com/science/article/pii/S2238785419301905