GCodeutils Documentation

Release 1.3

Olivier Jolly

June 20, 2016

Contents

GCodeUtils is a collection of scripts to manipulate GCode, usually as filters, as in the Un*x philosophy.

CHAPTER 1

Installation

GCodeUtils is installable from PyPI with a single pip command:

```
pip install gcodeutils
```

Alternatively, GCodeUtils can be run directly from sources after a git pull:

```
git clone https://github.com/zeograd/gcodeutils.git
cd gcodeutils && python setup.py install
```

Once GCodeUtils is installed, the .py files located in the cura_plugins subdirectory can be copied into the *plugins* directory of Cura to be callable from within Cura itself.

CHAPTER 2

gcode_tempcal

gcode_tempcal modifies an existing gcode program to introduce temperature gradient along the Z axis.

When finetuning the extrusion temperature for an unknown filament, it is possible to perform an unattented calibration print where the temperature changes along with Z. Once the print is finished, you can observe which height the print looks better and determines the temperature at which this part was printed.

2.1 Use case

Finding out what is the ideal temperature to print with a new spool of filament can be achieved using the technique described on this video : https://www.youtube.com/watch?v=FSOPsRiiOZk

Basically, you create a tower that you'll slice to print only the outer shell.

Then you update manually the gcode program to insert temperature decrase as you print and you inspect the print quality to find out which temperature resulted in the best appearance.

gcode_tempcal comes with a suitable cuboid model and a script to perform this temperature alteration in the gcode produced by your slicer to get you ready to print in a shorter time.

You may also want to use a tower with vertical holes like this one : http://www.thingiverse.com/thing:826019 so that you'll both see perimeter appearance and bridge quality depending on the temperature.

2.2 Usage

Follow the directions in the video https://www.youtube.com/watch?v=FSOPsRiiOZk except that you can use the cuboid located at this location https://github.com/zeograd/gcodeutils/blob/master/models/temperature/temperature_hollow_tower.stl which is already hollowed out (you may want to check that the slicer will generate at least one outer shell for the thin walls. If not you can force the perimeter width or regenerate the mode out of the self documented openSCAD model at the same location).

Once sliced, instead of manually editing the gcode program, use **gcode_tempcal** to insert the temperature change instructions.

```
$ gcode_tempcal 220 210 temperate_hollow_tower.gcode temperature_grad220-210.gcode -v
[ ... heights snippet removed ...]
INFO:temperature gradient from 220.0°C, altitude 0.30mm to 210.0°C, altitude 99.95mm
DEBUG:target temp for layer #2 (height 0.30mm) is 220.0°C
DEBUG:target temp for layer #34 (height 9.65mm) is 219.0°C
DEBUG:target temp for layer #64 (height 18.65mm) is 218.0°C
DEBUG:target temp for layer #94 (height 27.65mm) is 217.0°C
```

```
DEBUG:target temp for layer #124 (height 36.65mm) is 216.0°C
DEBUG:target temp for layer #154 (height 45.65mm) is 215.0°C
DEBUG:target temp for layer #185 (height 54.95mm) is 214.0°C
DEBUG:target temp for layer #215 (height 63.95mm) is 213.0°C
DEBUG:target temp for layer #245 (height 72.95mm) is 212.0°C
DEBUG:target temp for layer #275 (height 81.95mm) is 211.0°C
DEBUG:target temp for layer #305 (height 90.95mm) is 210.0°C
```

```
usage: gcode_tempcal [-h] [--min_z_change MIN_Z_CHANGE] [--continuous]
                     [--steps STEPS] [--verbose] [--quiet]
                     start_temp end_temp [infile] [outfile]

Add temperature gradient to gcode program

positional arguments:
  start_temp            Initial temperature (best set to the default slicing
                        temperature). For instance, for ABS you may want 240
                        and 200 for PLA.
  end_temp              End temperature for the gcode program. Usually lower
                        than the initial temperature. Make sure that your
                        material can be still be extruded at this temperature
                        to avoid clogging your extruder.
  infile                Program filename to be modified. Defaults to standard
                        input.
  outfile               Modified program with temperature gradient. Defaults
                        to standard output.

optional arguments:
  -h, --help            show this help message and exit
  --min_z_change MIN_Z_CHANGE, -z MIN_Z_CHANGE
                        Minimum height above which temperature gradient is
                        created. If you have a special start sequence playing
                        with temperatures, you may want to raise this to avoid
                        overlapping of temperature. Defaults to 0.1mm which is
                        compatible with NopHead ooze free unattended start
                        sequence.
  --verbose, -v         Verbose mode. It notably outputs the mapping between
                        temperature and height if you have troubles figuring
                        it out.
  --quiet, -q           Quiet mode

temperature control:
  --continuous, -c      Switch to a continuous gradient generation where
                        temperature is recomputed for every layer. You may
                        want this in the case of very precise and fast hotend.
                        Defaults to discrete temperature gradient divided in X
                        steps.
  --steps STEPS, -s STEPS
                        Number of steps used to create a discrete gradient
                        when using the default gradient generation model.
                        Defaults to 10 steps. This setting is not used when
                        using the continuous gradient generation model.
```

CHAPTER 3

gcode_mod

gcode_mod modifies an existing GCode program to translate all moves along X and Y by some amount or to convert all extrusion distance to relative.

3.1 Use case

As it stands now, its can be used to reprint at another place a part for which you only have the GCode program. If you have the original stl, slicer program and settings and want to print at another location on your bed, you better generate a new GCode program.

This program can also be used to convert absolute extrusion to relative extrusion so that further GCode processing is eased. It is used for instance in the **gcode_stretch** as preprocessor before performing toolpath changes required for stretching.

3.2 Usage

Call **gcode_mod** with the GCode either in plain text (or piped) or by giving its filename. Depending on the required modification, you have to pass either X and Y amount to translate and/or -e to enable relative extrusion.

gcode_mod attempts to handle relative and absolute moves as well as position setting (G92) but you better double check the generated GCode until more feedback have been factored into polishing the translation algorithm.

```
usage: gcode_mod [-h] [-x amount] [-y amount] [-e] [--verbose] [--quiet]
                 [infile] [outfile]

Modify gcode program

positional arguments:
  infile         Program filename to be modified. Defaults to standard input.
  outfile        Modified program. Defaults to standard output.

optional arguments:
  -h, --help     show this help message and exit
  -x amount      Move all gcode program by <amount> units in the X axis.
  -y amount      Move all gcode program by <amount> units in the Y axis.
  -e             Convert all extrusion to relative
  --verbose, -v  Verbose mode
  --quiet, -q    Quiet mode
```

CHAPTER 4

gcode_stretch

gcode_stretch modifies existing GCode program to change the size of "printed" holes.

It is a port of skeinforge stretch module, made to work with other modern libre slicers.

> **Warning:** This filter isn't extensively tested yet. Please apply an appropriate amount of scepticism and report both failures and successes.

4.1 Use case

When designing technical 3D parts for printing, you often need to make sure that the hole made in your part will accomodate another part of a known size. Typically, you want to be able to fit a M8 screw or rod.

Due to several reasons, the printed part may end up with a smaller size than expected.

> **Note:** To achieve the right round hole size, when using OpenSCAD, you should use polyholes.

If you have no way to improve the original design to increase the hole size (not having access to the source files, not having round holes or just feeling very lazy), **gcode_stretch** can help increase it.

4.2 Usage

The GCode to modify must be prepared so that **gcode_stretch** knows about the theorical edge width and loop types. This step is specific to each slicer.

4.2.1 Preparation of Slic3r GCode

Upstream slic3r doesn't provide any distinction of external and internal perimeters in its GCode. Since the 2 July 2015, slic3r doesn't generate usable comments to distinguish properly all loops type.

There is an open ticket regarding this GCode verbosity. If a solution is found, regular slic3r versions will be compatible with **gcode_stretch**.

Once done, you need to set 2 settings to produce compatible GCode :

- "External perimeters first" (in *Print Settings* > *Layers and perimeters* > *Advanced*)
- "Verbose G-code" (in *Print Settings* > *Output options* > *Output file*)

At this point, you can generate GCode and apply **gcode_stretch** manually or you should be able to use **gcode_stretch** as post processing script (settable in *Print Settings* > *Output options* > *Post-processing scripts*, but untested).

4.2.2 Preparation of Cura GCode

Cura generates usable GCode for stretching by default.

You can either manually postprocess generated GCode or copy the gcode_stretch.py file to your Cura plugins directory to enable stretching from within Cura itself.

4.2.3 Command line

GCode can be provided in standard input and will be output in standard output by default. You can set the GCode filename on command line and set the output filename too.

If you need to change the sizing, you're advised to play with the stretch_strength parameter first (it defaults to 1, increase it to increase hole size [1.1, 1.2, ...], decrease it to decrease hole size [0.9, 0.8, ...]) . Changing other parameters imply knowledge of the inner working.

```
usage: gcode_stretch [-h]
                     [--cross_limit_distance_over_edge_width CROSS_LIMIT_DISTANCE_OVER_EDGE_WIDTH]
                     [--stretch_from_distance_over_edge_width STRETCH_FROM_DISTANCE_OVER_EDGE_WIDTH]
                     [--loop_stretch_over_edge_width LOOP_STRETCH_OVER_EDGE_WIDTH]
                     [--edge_inside_stretch_over_edge_width EDGE_INSIDE_STRETCH_OVER_EDGE_WIDTH]
                     [--edge_outside_stretch_over_edge_width EDGE_OUTSIDE_STRETCH_OVER_EDGE_WIDTH]
                     [--stretch_strength STRETCH_STRENGTH] [--verbose]
                     [--quiet]
                     [infile] [outfile]

Modify GCode program to account for stretch and improve hole size

positional arguments:
  infile                Program filename to be modified. Defaults to standard
                        input.
  outfile               Modified program. Defaults to standard output.

optional arguments:
  -h, --help            show this help message and exit
  --cross_limit_distance_over_edge_width CROSS_LIMIT_DISTANCE_OVER_EDGE_WIDTH
                        Distance after and before a point where moves are
                        considered to determine the local normal. Defaults to
                        5.0. Set too low or too high, it might cause points no
                        to be moved in the right direction.
  --stretch_from_distance_over_edge_width STRETCH_FROM_DISTANCE_OVER_EDGE_WIDTH
                        Distance after and before a point where moves are
                        considered to determine the local normal. Defaults to
                        2.0. Set too low or too high, it might cause points no
                        to be moved in the right direction.
  --loop_stretch_over_edge_width LOOP_STRETCH_OVER_EDGE_WIDTH
                        Stretching strength for "loop" (extra shells),
                        defaults to 0.11
  --edge_inside_stretch_over_edge_width EDGE_INSIDE_STRETCH_OVER_EDGE_WIDTH
                        Stretching strength for "inner perimeter", defaults to
                        0.32
  --edge_outside_stretch_over_edge_width EDGE_OUTSIDE_STRETCH_OVER_EDGE_WIDTH
                        Stretching strength for "outer perimeter", defaults to
```

```
                        0.1
  --stretch_strength STRETCH_STRENGTH
                        Stretching stretch factor. This is the first setting
                        you'll want to change to modify the hole size
  --verbose, -v         Verbose mode
  --quiet, -q           Quiet mode
```

4.3 Inner working

This filter uses metadata provided by the slicer to determine the type of path to which a point belong. Only points being part of the visible shell are affected (infill, skirt, brim and so on aren't related to hole sizes).

When a loop is flagged in the GCode (external perimeter being the outer shell, external perimeter being the inner shell or internal perimeter), all points in this loop are considered for streching.

For each point, the normal vector is estimated by looking at the next and previous points of this loop. Once the normal vector is found, the point is moved away proportionally to the edge width, normal strength and loop type stretching strength. Extrusion is also adapted (quite empirically at this moment) to limit overextrusion in the shell / infill boundary.

CHAPTER 5

gcode_optimize_arcs

gcode_optimize_arcs modifies an existing GCode program to translate all moves along a circular are into G2/G3 commands.

5.1 Use case

As it stands now, its can be used to post-process gcode generated from STL. As STL does not have means to describe arcs the resulting gcode does not contain arcs. Especially if there are small thru-holes with a high segmentation to be printed it might kill the print as many small movements happen a the same place.

5.2 Usage

Call **gcode_optimize_arcs** with the GCode either in plain text (or piped) or by giving its filename.

gcode_optimize_arcs attempts to handle relative and absolute moves as well as position setting (G92) but you better double check the generated GCode until more feedback have been factored into polishing the translation algorithm. Moreover there is currently no extrusion correction so you better us a high segmentation. In OpenSCAD this is controlled by $fn so a good starting point (for small radius) is $fn=64;

```
usage: gcode_optimize_arcs [-h] [--inplace] [--verbose] [--quiet]
                  [infile] [outfile]

Modify GCode program to account arcs and replace the G1 with G2/G3

positional arguments:
  infile         Program filename to be modified. Defaults to standard input.
  outfile        Modified program. Defaults to standard output.

optional arguments:
  -h, --help     show this help message and exit
  --inplace, -i  modify the code in-place, usefull if gcode_optimize_arcs is used as post processor
  --verbose, -v  Verbose mode
  --quiet, -q    Quiet mode
```

5.2.1 Inner working

Arcs are detected using a moving window over the gcode commands. The minimum size of the window is 9 so that at least 8 segments are needed to constitute an arc. The validity is checked using the following criterias: * all segments

are printed using the same speed (F parameter in gcode) * if segments extrude they have to have the same ration between extrusion (E param) and the length of th epath * the endpoints of the segments need to be equidistant on the arc