Smart Digital Image Correlation Patterns via 3D Printing

DIC_pat_gcode Code Manual (v1.0)

Jin Yang[1,*,†], Jialiang Tao [1,*], Christian Franck[1,‡]

[1] Department of Mechanical Engineering, University of Wisconsin-Madison, Madison, WI, USA
* These authors contributed equally to this work

Email: †jyang526@wisc.edu; ‡cfranck@wisc.edu

Github page: https://github.com/FranckLab/DIC_pat_gcode
MATLAB FileExchange page: https://www.mathworks.com/matlabcentral/fileexchange/90431-dic_pat_gcode-generate-g-code-files-to-control-a-3d-printer

Last updated on April 14, 2021

Contents

1 Introduction: 3D Printing Control via customized G-codes 2

2 Function "funWriteGcode" to generate G-code scripts 2
2.1 G-code start part 3
2.2 G-code main body part 3
2.3 G-code end part 4

3 Example 4

Acknowledgements 7

References 7

1 Introduction: 3D Printing Control via customized G-codes

The significant advantage of our proposed DIC patterning technique [1] is that each dot location and size can be precisely controlled via a customized G-code, a common language used for controlling 3D printers [2]. In general, a G-code file is composed of three parts: *start part*, *main body* and *end part*. In the *start part*, the initial coordinates of the needle's position (x, y, z) and extrusion are set to a reference state after a prime extrusion is set. The *main body* part of G-code follows the steps to control the position of the needle and the extrusion volume of the ink transferred to the sample surface. During the actual printing cycle the needle tip is in close proximity to the top surface of the sample but is typically programmed to avoid coming in direct contact with the surface. In the *end part*, the extruder is moved away from the sample surface and its motor turned off.

A list of G-code commands used to control 3D printer motion are summarized in Table 1.

Table 1: List of G-code commands used in controlling motor motion [2]

Command	Usage	Parameters	Example
G1	Linear move	Position (X, Y and Z), extrusion (E) and move speed (F)	G1 F200
G21	Set units to mm		G21
G28	Move to the origin		M83
G90	Use absolute coordinates		G90
G92	Set position	Position (X, Y and Z), extrusion (E)	G92 E0
M83	Set to relative mode		M83
M84	Motors off		M84
M302	Allow cold extrudes	State (P)	302
T	Select tool	Tool number	T0

2 Function "funWriteGcode" to generate G-code scripts

We provide a function called `funWriteGcode(x0,extrusionVol,filename)` to generate G-code scripts to control the movement of a 3D printer to print dots at coordinates `x0` with extrusion volume as `extrusionVol` .

```
function [Gcode] = funWriteGcode(x0,extrusionVol,filename)
%FUNCTION funWriteGcode(x0,extrusionVol,filename)
% to output a G-code script to control motions of 3D-printer to paint a
% designated pattern on the sample surface
% -------------------------------------------------------------
% Inputs:
%
%   x0          [required] A 3-column matrix to store dot  coordinates.
%                          The row number equals the number of the beads,
%                          and the x,y z coordinates are stored in the
```

```
%                          first, second and third column, respectively
%   extrusion  [required] Extrusion volume (unit: mL) of the ink to print
%                          each dot. It needs to be a single scalar or a
%                          vector of the same length as the number of dots
%   filename   [optioanl] Name of the written G-code file, needs to be a
%                          string ended with 'gcode'. It is called
%                          'Patten.gcode' by default.
%
% -------------------------------------------------------------
% Output:
%
%   Gcode       G-code lines are written into an output "*.gcode" file.
```

2.1 G-code start part

At the beginning of the G-code start part, we set the initial position of the extruder as (`X0 Y0 Z0`). We allow cold extrusion and set units to millimeters. We use absolute coordinates to control the movement of the extruder. The extrusion amount will be transferred to the extruding distance of the motor and we use relative distances to extrude the filled ink.

```
% prepare motor for printing
Gcodebegin=strcat('T0 ; select extruder0\n' ...
                 ,'G92 X0 Y0 Z0 E0\n '...
                 ,'G1 E0.0001\n' ...
                 ,'G92 E0\n' ...
                 ,'T0\n' ...
                 ,'M302 P1\n' ...
                 ,'G21 ; set units to millimeters\n' ...
                 ,'G90 ; use absolute coordinates\n' ...
                 ,'M83 ; use relative distances for extrusion\n');
```

2.2 G-code main body part

We complete the body part of the G code by printing all the dots.

```
fileID = fopen(filename,'w');
x= x0(:,1);
y= x0(:,2);
z= x0(:,3);
Gbody=sprintf('; start pattern\nG1 Z2 \n');
G3=sprintf('G1 Z6 F800');
extrusionVol=extrusionVol/(1.61e-7)*0.0001; % transfer the extrusion volume to extrusion distance of motor

% Go through all the dots
for i=1:nbeads
    G1=sprintf('G1 X%.4f Y%.4f E0 F500',x(i),y(i)); % go to the designed position
    G2=sprintf('G1 Z%.4f E%.5f F200',z(i),extrusionVol(i)); % extrude a certain volume of ink
```

```
    G3=sprintf('G1 Z6 F800');
    Gbody=strcat(Gbody,'\n',G1,'\n',G2,'\n',G3);
end
```

2.3 G-code end part

Finally, we retreat and home the extruder head, and set motors off.

```
Gcodeend=strcat('G1 E-0.075 F60 ; retreat extruder\n'...
    ,'G1 Z20 F500 ; retreat head\n'...
    ,'G28 ; home the exruder\n'...
    ,'G92 X0 Y0 Z0 E0; set the current position to be(0,0,0)\n'...
    ,'M84 ;Motors off');

Gcode=strcat(Gcodebegin,Gbody,Gcodeend);
fprintf(fileID,Gcode);
```

3 Example

In the `main_example.m` file, we generate a G-code script that acts as input for a 3d printer to print random speckle patterns. Here, the spatial distribution of all the speckle dots follows the Poisson-disc sampling rule, as shown in Fig. 1 [3, 4], whose parameters are defined as follows:

```
%% Initialize parameters

% Total number of speckle dots
nDots = 500;

% DIC print pattern ROI size (mm)
ROISize = [12.7, 12.7];

% Resolution of camera: pixels of a single image
pxSize = [1024 1024];

% Generate a set of dots following a random Poisson distribution
spacing = 15; showIter = 0;
[x0] = poissonDisc(pxSize,spacing,nDots,showIter); % Coordinates of pattern dots

% Transform the units of each pattern dot's position from "pixel" to "mm"
transRatio = ROISize./pxSize;

x0_pw = x0*diag(transRatio); % x0 in physical world with unit "mm"

% Z-motions are set to be zeros for flat, planar sample top surface
x0_pw(:,3) = 0;

% Output G-code script file name
fileName = 'Pattern.gcode';
```

```

% Set uniform extrusion volume % could be non-uniform values
extrusionVol = 1.61e-7; % Unit: mL
```

[1] `nDots = 500;` This line defines total number of dots. Each single speckle dot can occupy 5 ∼ 7 pixels in typical DIC patterns. Based on the size of the entire region of interest (ROI), `nDots` can be approximated as

$$\text{nDots} = \frac{\text{ROI area}}{\pi r^2} \tag{1}$$

where r is the single speckle dot radius.

[2] `ROISize = [12.7 12.7];` This line defines the sample size in the physical world (unit: mm).

[3] `pxSize = [1024 1024];` This line defines the image size of the DIC image (unit: px).

[4] In this example, we apply the Poisson-disc sampling rule to specify the location of dots, where the minimum distance between each pair of dots is defined as `spacing = 15;` and the parameter "showIter" is set as `showIter = 0;` to not show details during the Poisson-disc sampling procedure.

[5] `x0_pw(:,3) = 0;` We only test a planar sample in this example where all the dot z-coordinates are set as zeros. For a non-planar sample, `x0_pw(:,3)` can be non-zeros.

[6] `fileName = 'Pattern.gcode';` File name of the generated G-code script.

[7] `extrusionVol = 1.61e-7;` is the amount of ink extrusion to print each single dot (unit: mL). It can be a single value or a vector of the same length as the number of dots to allocate different extrusion volumes.

After executing " `main_example.m` " file, a designed G-code script is generated automatically, as shown in Fig. 2.

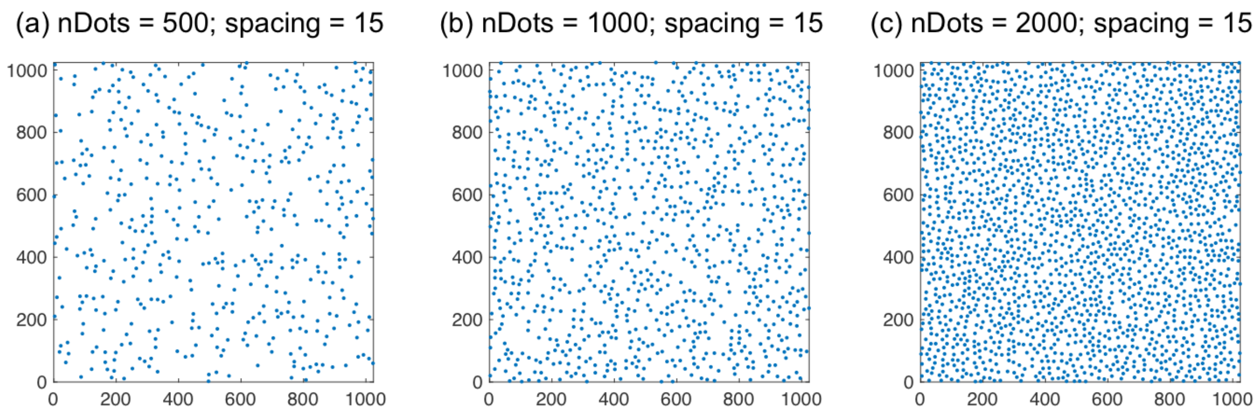

Figure 1: Examples of Poisson disc sampling.

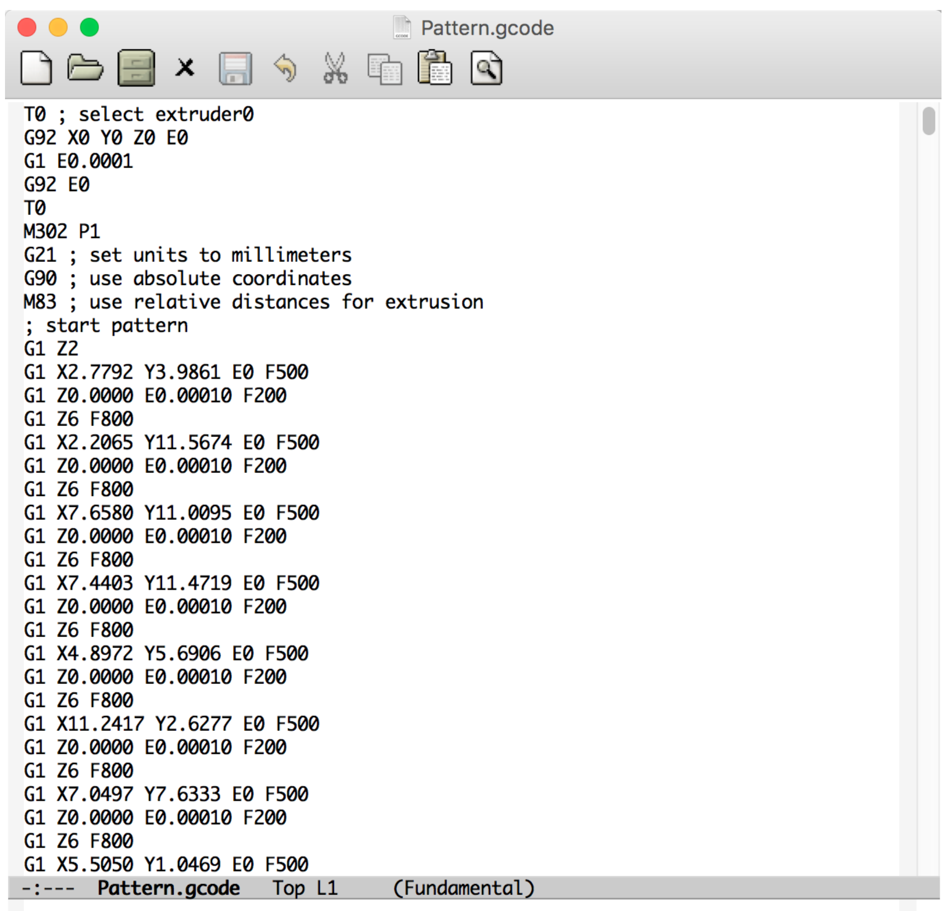

Figure 2: Generated G-code script after executing “ `main_example.m` ” file.

Acknowledgements

We gratefully acknowledge the funding support from the Office of Naval Research (Dr. Timothy Bentley) under grants N000141712058 and N000142112044.

References

[1] J Yang, Tao JL, and C Franck. Smart digital image correlation patterns via 3d printing. *Experimental Mechanics*, 2021.

[2] Automation systems and integration — Numerical control of machines — Program format and definitions of address words — Part 1: Data format for positioning, line motion and contouring control systems. Standard ISO 6983-1:2009, International Organization for Standardization, Geneva, CH, 2009.

[3] *Poisson Disc Sampling Reference 1*. http://extremelearning.com.au/an-improved-version-of-bridsons-algorithm-n-for-poisson-disc-sampling.

[4] *Poisson Disc Sampling Reference 2*. https://github.com/mohakpatel/Poisson-Disc-Sampling.