­­­­­­

論文・ノート・速報・解説・総説

G-coordinator: G-code生成の新たな手法とその可能性

谷口　朝洋

G-coordinator: A New Way To Generate G-code And Its Possibility

Tomohiro TANIGUCHI\*

大阪府立大学　工学域　機械系学類　機械工学課程

機械力学研究グループ

【Abstract】

This study focuses on the development of a novel method for generating G-code, the program read by a 3D printer. We have developed an open-source software called G-coordinator, which utilizes Python and mathematical functions to create G-code directly and construct three-dimensional shapes. This alternative path generation technique, which is distinct from traditional slicing methods, offers the potential to tackle complex shapes that were previously challenging to produce and enables the formation of more precise printing conditions and paths. It also facilitates the straightforward realization of mathematical shapes.

【Keywords】

3D printer, G-code, Software, Modeling

【要旨】

本研究は，MEX型3Dプリンタを用いて造形を行う際に，機械に読み込ませるプログラムであるG-codeの新たな生成方法に関するものである．Pythonと数学的な関数を用いてG-codeそのものを直接作成し，3次元形状を構築することできるG-coordinator ( <https://github.com/tomohiron907/G-coordinator> )というオープンソースソフトウェアを開発した．従来の「スライス」とは異なる手法によるパス生成により，より精密な印刷条件の調整が可能になるとともに，いままでは難しかった形状や，数理的な形状の容易な実現が期待できる．

【キーワード】

3Dプリンタ，G-code, ソフトウェア，モデリング

# はじめに

## G-coordinatorとは

G-coordinatorは，Pythonを用いて，3Dプリンタ用G-codeを生成できるソフトウェアである．従来のスライスソフトによるG-code生成と異なり，造形から印刷条件の調整まで3Dプリントに必要な工程の全てをG-coordinatorひとつで完結させることができる．Windows, Macに対応したアプリケーションに加えて，Python用のライブラリも公開している．Figure1にG-coordinator（GUIアプリ）のスクリーンショットを示す．G-codeを書き出す手順は主に４つのステップから構成される．

1. コードを書いて，造形を決定
2. グラフィックスビューで詳細を確認
3. 印刷条件の設定
4. G-codeの出力

## 背景

一般的に，3Dプリンタを用いる際にはCAD等で3Dデータを作成し，その3Dデータをスライスソフトで処理し，G-codeを生成する．スライスソフトは，3Dプリンタのノズルパスを自動で算出することができ，多くのユーザに取って，より簡単にG-codeを作成することができる．しかし，より細かい制御やマニュアルでの制御を求めるユーザにとっては，スライスソフトでは３Dプリンタの挙動や印刷設定のマニュアル制御ができない．そこで，近年，3Dモデルを介さず直接G-codeを作成するG-code Modelingの手法が台頭している．G-coordinatorは，G-code Modelingの新たな手法として位置付けられる．

## 目的

本研究は，広範なユーザに向けた高い自由度を有するG-codeモデリングを実現し，これにより従来のCADやスライサでは到達できなかった造形を具現化することを目的としている．G-coordinatorにより，3Dプリントにおける新たな可能性が開かれ，現行の技術では実現が難しい形状や構造の物体を容易に製造することが可能となる．従って，本研究は3Dプリンティング技術の進化と革新に寄与することが期待される．

## G-codeとは

G-codeは，コンピュータ制御された機械操作を指示するための標準的なプログラミング言語であり，その本質はツール（MEX型3Dプリンタにおいてはノズル）の位置を記述する座標列にある．

G-codeの基本的な構造は，各行においてツールの移動先の座標が指定され，これによってツールの位置が制御される．たとえば，以下は一般的なG-codeの例である．

G1 F1000 X100 Y80 Z0.4 E0.5

このコマンドで，機械はノズルを(X, Y, Z) = (100, 80, 0.4)の座標に1000mm/minの速さで移動させながらフィラメントを0.5mm送る．これらのコマンドが，G-codeの内部では，数百~数万行繰り返される．G-codeには印刷速度やフィラメント送り量など様々な要素があるものの，これらは，ソフトウェアで自動的に決定できるため，G-code Modelingにおいてユーザはノズルの移動する座標列のみを考慮すれば良い．

　上記のG-codeの性質から，G-coordinatorでのモデリングに不可欠な二つの用語を定義する．

セグメント：ノズルが移動する最小単位の線分．

パス：樹脂が連続的に押し出されてできる一本の経路．

Figire2は，６角形のパスが下から積層されており，一番上のパスの最初のセグメントまでが表示されている．このように，セグメントが集まってパスになり，パスが集まってひとつの造形物になる．

# 関連研究

## G-code Modelingの手法

現代の3Dプリントにおいて，G-code Modeling の手法としてRhino+GrasshopperやFullcontrolGCodeDesignerなどが広く利用されている．本節では，これらの手法とG-coordinatorとの比較に焦点を当て，それらの特徴と差異を明らかにする．

## Rhino+Grasshopper

RhinocerosとそのプラグインGrasshopperの組み合わせは，G-code Modelingの代表的な手法の一つであり，その特徴は以下の通りである．

オープンソース: 非公開

拡張性: 有り

多軸制御: 有り

インフィル: 一部制限あり

参入障壁: 低い

Rhino+Grasshopperはオープンソースではないが，アドオンなどの拡張性があり，多軸制御やインフィルに関してはアドオンで実現できるものも多い．また，プログラミングの形式がビジュアルプログラミングであることから，プログラミングに慣れていないユーザにも扱いやすいソフトウェアと言える．しかし，ソフトウェアの値段が比較的高価であり，一般ユーザが気軽にソフトウェアを試すことができない

## FullContolGcodeDesinger

FullContolGcodeDesingerは，Excel上で動作するマクロとしてのソフトウェアである．

オープンソース: 公開

拡張性: 制限あり

多軸制御: 非対応

インフィル: 非対応

参入障壁: 低い

FullContolGcodeDesingerはオープンソースであるが，拡張性には限りがあり，多軸制御やインフィルの生成には対応していない．Excelシートであるため，新たなソフトをインストールする必要がなく，初めG-code Modelingを行うユーザに向いているが，Excelのみで造形を完成させる性質上，複雑な形状や関数の扱いには限界がある．また，その点を克服したpythonライブラリ，fullcontrolも登場しているが，python環境をユーザが整える必要があるなどの弱点もある．

## G-coordinator

G-coordinatorは，G-code Modelingの新しい手法であり，以下がその特徴である．

オープンソース: 公開

拡張性: 有り

多軸制御: 有り

インフィル: 有り

参入障壁: 高い

G-coordinatorはオープンソースであり，あらゆるコードやライブラリを公開している．拡張性に優れ，Pythonを用いた造形により，様々なデータソースを活用可能である．多軸制御やインフィルの生成にも対応している．しかしながら，G-coordinatorでの造形には多少の数学的知識とコードのスキルを要求してしまう側面がある．これらの特徴により，G-coordinatorはG-code Modelingの新たな展開を可能にしており，従来の手法と比較して優れた柔軟性と拡張性を提供している．

# 実装

## G-coordinatorの造形方法

G-coordinatorで造形を行うための方法を，最も簡易な円柱モデルを用いて述べる．なお，円柱壁の造形コードを補足資料に掲載している．いかに複雑な形状であっても，基本的な造形のプロセスは同じである．G-coordinatorでは，パスの*x, y, z*座標列を計算して，それらの座標列からPathオブジェクトを作成する．より詳細な造形コードの作成に関してはG-coordinator公式ドキュメントを参照されたい．

### 円形パスの作成

import gcoordinator as gc

full\_object = []

まず，gcoordinatorのライブラリをインポートして，full\_objectというリストを用意する．

このリストには，作成したパスを印刷順番に格納していく．

arg = np.linspace(0, 2\*np.pi, 100)

0から2πまでの項数100の角度についての等差数列を作成し，

x = 10 \* np.cos(arg)

y = 10 \* np.sin(arg)

z = np.full\_like(arg, (height+1) \* 0.2)

その角度のcos, sinがそれぞれ*x, y*座標列になる．*ｚ*座表列に関しては，argと同じ項数で，値がレイヤーごとに0.2（レイヤー厚さ）大きくなる数列を用意すればいい．

wall = gc.Path(x, y, z)

full\_object.append(wall)

その後，*x, y, z*座標列を引数として，Pathオブジェクトのインスタンスを生成する．生成したオブジェクトをfull\_objectに追加する．

gc.gui\_export(full\_object)

その後，印刷するパスが格納されたリストをgui\_exportに渡す．円形パスでは，full\_obejctの中には，パスはひとつだけ格納されている．

### 円柱の造形

円柱を造形するには，前節で述べた円形のパスを縦に積み上げていけば良い．この処理はfor ループを用いて用意に記述できる．

for height in range(100):

このforループの中で，heightは0から99まで1ずつ増加する．X, Y座標列は円形パスと同じで，Z座標列をheightの値に応じて0.2ずる大きくすれば良い．

z = np.full\_like(arg, (height+1) \* 0.2)

ここで，heightに1を足してから0.2をかけているのは，ループの始めにおいて，heightの値が0の時に，ノズルの高さが0.2にしたいからである．そして，forループごとにパスオブジェクトを作成し，full\_objectに追加すれば良い．円柱を造形するための造形コード全体を付録に載せる．

## 個別印刷設定

G-coordinatorでは，パラメータツリーで設定したグローバルな印刷設定に対して，個別のパスへの詳細設定をコードから指定することができる．例として，ベッド定着のためにファーストレイヤにおいてのみ，印刷速度を遅くしたいという状況を考える．

この時，以下のコードをforループの中に追加すれば，ファーストレイヤの印刷速度をマニュアルで指定できる．

if height == 0:

wall.print\_speed = 5000

操作画面右側のパラメータツリーでのprint\_speedの値が10000の時，上のコードを実行すれば，第0層目（ファーストレイヤ）では印刷則と5000mm/min, 他の層では，10000mm/minとなり，印刷速度の個別制御が実現できる．印刷速度に限らず，射出係数，リトラクション，z hop，などグローバル設定で指定できる設定値は，全て個別に値をマニュアル調整することができる．

# ユースケース

以下に，G-coordinatorを用いて造形したユースケースを示す．

## 網目テクスチャ

京都の新工芸舎にインスパイアを受けて作成した作成をFig.3に示す．ノズルをジグザグに蛇行させ，一層ごとに位相を反転させることで，積層痕を意匠化させた．スライスソフトでは，直接ノズルパスを編集できないため，テクスチャを付与することが困難であった．

## 数理的形状

G-coordinatorではコードと関数を用いてモデリングを行うため，数理的な形状の実現が極めて容易にできる．Figure4は，フラクタル形状を3次元に押し出したもの．他にも，一葉双曲面や，リサージュ曲線を造形するためのコードも公開している．

## 振動による図柄表示

Pythonの持つ豊富なライブラリを用いれば，あらゆるものをモデリングのために活用できる．Figure5は，OpenCVのライブラリを用いて，モノクロ写真から，ある座標でのピクセルの濃さを抽出，その値を振動の振幅に重みとしてかけている．白のプレートを印刷後，黒の樹脂で振動パターンを印刷している．その結果，ミクロでは不規則な振動が，マクロでは写真に見える状況を実現している．

## インフィル

他のG-code Modelingと比較して，G-coordinatorの大きな特徴のひとつとして，Fig.6に示すようなインフィルを利用できることが挙げられる．インフィルを利用できるようになると，実用性を意識したものを作成することができる．現在は，line infiil, gyroid infillの二種類を利用可能である．G-coordinatorでは，外枠のパスを関数に渡せば，その内部を埋めるインフィルパスが関数の返り値として渡される．以下のコードでは，最初の20層のみにインフィルが発生する．

if height<20:

gyroid = gc.gyroid\_infill(wall, infill\_distance = 2)

full\_object.append(gyroid)

# 将来展望

## テスト機能

G-coordinatorを用いて様々な造形を可能にするためにいくつかの機能を開発中である．

### 数式処理

G-coordinatorの得意な点として，数理的な形状の実現について述べた．現在は，さらにその点を生かすべく，様々な関数やクラスを開発中である．具体的には陰関数で定義される曲面や曲線をより簡便な形式で描画することをめざしている．

### 多軸制御

多軸3Dプリンタの制御は造形の際に考慮すべき要素が座標に加えてノズルの傾きも造形に関わりその難易度は格段に高くなる．そこでよりユーザが多軸造形を簡単に行うことができるような枠組みを開発中である．

### スライス機能

G-coordinatorは基本的にすべてコードを用いて造形を行う．しかし，工業的な形状の造形をコードのみで行うことは難しい．そこで，G-coordinatorにはstlデータをスライスする関数も用意している．純粋なスライスソフトとは異なり，スライスしたパスに対して，個別の印刷設定を付与することができる．将来的には，前節で述べた多軸制御と絡めて，多軸マシンのためのスライス機能もサポートする予定である．

## 今後の展望

G-coordinatorは，数学とコードにより，いままで出来なかった形状を実現する一方で，同時に，ユーザには多少の数学的知識と，プログラムに関するスキルを要求するものであり，参入のための障壁は比較的高いと言わざるをえない．そこで，現在想定しているG-coordinatorの将来像は大きく分けて二つある．

一つは，Processingなどを代表とするビジュアルアートを実物化させるためのソフトウェアとしての活用である．これにより，アーティストやデザイナーは数学的なアルゴリズムを用いて，従来不可能だった美しい形状を3Dプリンティングで具現化することが可能になる．

もう一つは，G-coordinatorが数理的な形状に得意な性質を生かし，物理シミュレーションや計算力学を応用して形状を実物化させるソフトウェアとしての利用である．これにより，エンジニアや科学者は複雑な物理的な現象をシミュレートし，それを具現化するのにG-coordinatorを活用できる．プログラムを使った3D形状の生成と制御が主眼となる点で，どちらの展望も共通している．

# 結論

FFF 式 3Dプリンタユーザにとって，CAD などの造形手段と，印刷のための調整を行うスライサを同時に置き換える新たな手段としてのG-coordinatorのポテンシャルについて述べた.

スライスソフトに比べて，全部を機械任せにできない欠点はあるものの，裏を返せば，3D プリンタの挙動の細部まで，マニュアル操作可能になるということでもある.

G-coordinatorは，数学とプログラムの力を駆使して新しい形状を実現するための強力なツールとして，今後の3D プリント分野でさらなる進化が期待される.

# 参考文献

# 付録

以下にG-coordinatorを用いて円柱壁を造形するサンプルコードを示す．なお，以下のコードは，G-coordiantor ver3以降のバージョンで動作する．Ver2系列の造形コードとは，細部が少し異なることを注意されたい．

import gcoordinator as gc

import numpy as np

full\_object=[]

for height in range(100):

arg = np.linspace(0, 2\*np.pi, 100)

x = 10 \* np.cos(arg)

y = 10 \* np.sin(arg)

z = np.full\_like(arg,(height+1)\*0.2)

wall = gc.Path(x, y, z)

# if height == 0:

# wall.print\_speed = 5000 # ファーストレイヤの印刷速度を5000mm/min

　　　# infill = gc.gyroid\_infill(wall) # インフィルパスの計算

# full\_objet.append(infill) # インフィルパスを印刷リストに追加

full\_object.append(wall)

gc.gui\_export(full\_object) # for G-coordinator(GUI Application)

# gc.show(full\_obejct) # for gcoordinator(python library)

グラフ, じょうごグラフ

自動的に生成された説明

Fig. 1

建物, 小さい, 暗い, 座る が含まれている画像

自動的に生成された説明

Fig. 2

暗い背景に浮かぶ月

低い精度で自動的に生成された説明

Fig. 3

Fig. 4

屋内, 座る, テーブル, 食品 が含まれている画像

自動的に生成された説明

Fig. 5(a)

写真, 探す, 記号, 立つ が含まれている画像

自動的に生成された説明

Fig. 5(b)

窓, スピーカー が含まれている画像

自動的に生成された説明

キーボードとマウス

中程度の精度で自動的に生成された説明

Fig. 6

Table 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | オープンソース | 拡張性 | 多軸印刷 | インフィル | スキル要求 |
| Rhino+GH | × | ○ | △ | △ | 低 |
| FullControl | ○ | △ | × | × | 低 |
| G-coordinator | ○ | ○ | ○ | ○ | 高 |