2017 年度 学士論文

「多様なマテリアルに応じた、3D プリンタ用の快適な制御ソフトウェアの構築」

平成 25 年度秋入学
慶應義塾大学　総合政策学部
田中浩也研究室
71335213 番　千葉真英

目次

[5] おわりに

[6] 謝辞

[7] 参考文献・引用

論文要旨

2017 時点、デスクトップ 3D プリンタの活用は、個人的な趣味の範囲をこえ、プロダクト（特に耐久性に重きをおかないもの）や医療用品、そしてアート作品といった、第三者に影響を及ぼすような分野にまで広がりをみせている。個人では購入できない価格帯（数百〜数千万クラス）の 3D プリンタを、メーカーがプロトタイプの過程で導入するケースは以前から存在したが、それと比べて安価（数十万クラス）な 3D プリンタも、最近は寸法精度や操作性が向上し、ものづくりベンチャーや研究の現場で、実験用の筐体外装や部品の製造に利用されることがメジャーになりつつある。このように少しずつ着実に、デスクトップ 3D プリンタはプロユースと呼べる工作機械への歩みを進めてきた。

また素材に関しては、特に医療分野で活躍が期待される、生体適合性があり柔軟性質な素材や、形状記憶性質の素材など、特異な物性をもつ材料が、3D プリンタ用にセットアップされ出し、単物性のプラスチック製品を生み出すだけが 3D プリンタの役割ではなくなっている。筆者の研究室でも、そのような素材を用いた介護用品の試作や、社会実装のための品質・安全評価方法について検証がなされているが、その様子からして、物性の豊かな素材ほど扱いは難しく、単純な形を出力するだけでも、汎用素材として普及している ABS や PLA より多くの時間とトライが必要であるのは確かである。

本研究では、このような背景を考慮した上で、既存の 3D プリンタの利用環境を向上させることを目的としたソフトウェアとハードウェアの開発を行った。具体的には以下の 2 つである。

第一に、多様な素材の登場に伴い複雑化したスライス（3D データを 3D プリンタで解釈可能な形式に変換する行為）時のパラメータの調整を補助し、素材毎の最適な出力値を探し易くすることを目的としたソフトウェアを製作した。

第二に、今後登場する可能性のある、フィードバック制御形式の 3 D プリンタの学習素材の収集方法に貢献するため、3D プリントの失敗の瞬間を撮影し、これまで曖昧にされてきた失敗原因の分析と観察手法の提案を行った。

また、論文の構成は以下の通りである。第 1 章では、研究動機とその背景、関連研究について述べる。第 2 章では、本研究で提案するソフトウェアと観察手法の提案及び、実装プロセスについて記述する。第 3 章では、それらを用いたユーザー実験の結果を記録し、第 5 章では、これまでの研究についての振り返りと、今後の展望について述べる。

本研究での具体的な検証内容は次のようになる。

・3D プリント中に変数を即時変更するソフトウェアの製作とその実用性について

・ノズル付近からの接写による、3D プリントの失敗原因の分析とその観察手法について

1. はじめに

1.1 研究動機とその背景

はじめに以下の写真を見ていただきたい。

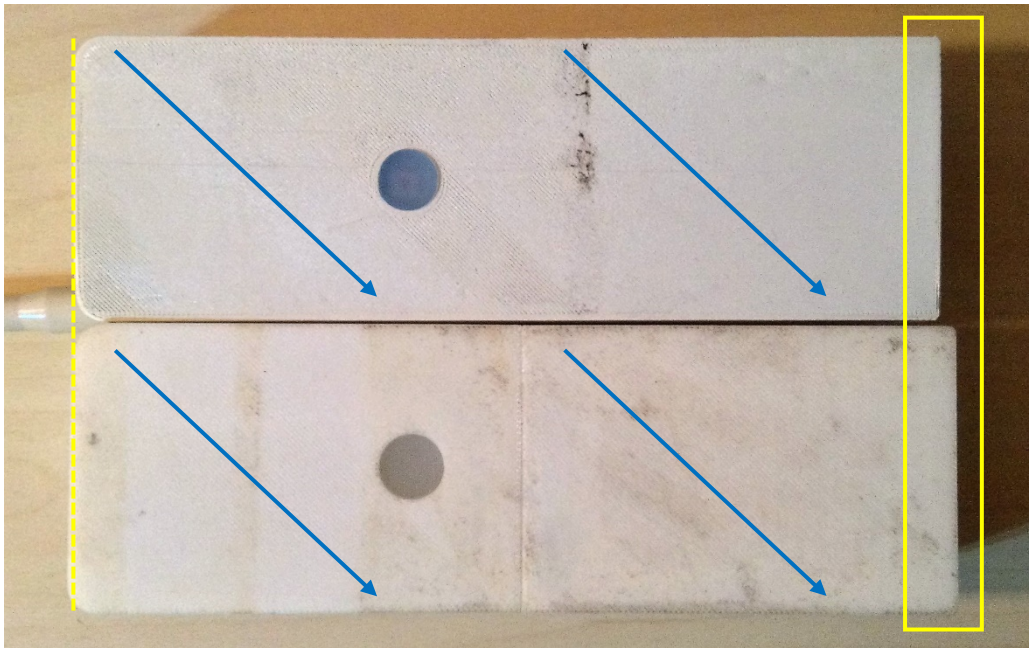

図 1　同じ 3D データを、使用頻度の異なる 3D プリンタで出力した写真。材料は両方とも ABS。

上下それぞれの筐体は、両者とも ABS を用い、同じ 3D データを使用頻度が異なる 3D プリンタで出力したものである。また、青色の矢印がツールパスの方向を表している。（上）の筐体が 1 年間駆使（同様の筐体を 150 個ほど出力している）した MakerBot Replicator 2x で、（下）の筐体が購入して間もない QIDI Tech(MakerBot Replicator 2x の互換機)により出力されたものである。また、黄色線内を拡大した写真が以下となる。

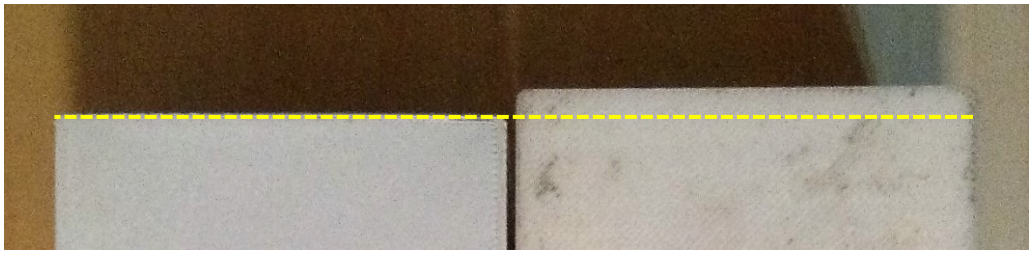

図 2　図 1 の黄色線内を拡大し、左に 90°回転させた写真

図 2 から、（右）の長辺の方が、（左）の長辺より少し長いのが分かる。ソースの 3D モデルの長辺は 203mm であり、（右）の長辺が 203.2mm であるのに対し、（左）の長辺は、201.3mm であった。つまり、MakerBot Replicator 2x で出力した筐体の長辺は、ソースより約 1 %縮小され出力されていたことになる。樹脂の収縮は、ツールパスの方向に向かい大きくなるが、図 1 から分かるように、パスの方向は同じであり、2 つの筐体の寸法が異なる主因は、材料の収縮ではないと言える。また、ここでは他辺に縮小は見られなかった。

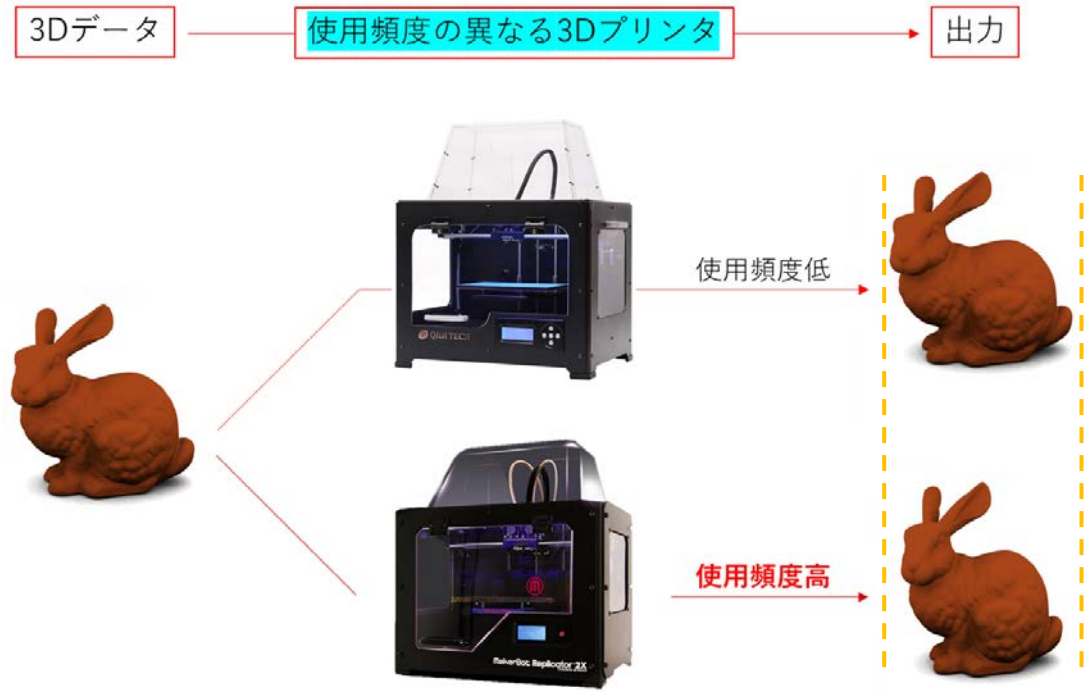

図 3　使用頻度の違いが出力物の縮尺に影響する下りの図解。

今回のケースは、筐体中に組込む基盤の収まりが、印刷回数を重ねる内に悪くなりだしたことで発覚し、その原因は X 軸側のゴムベルトの劣化が主であったが、各軸のモータやシャフトなど、他のパーツが原因となることも十分考えられた。3D プリンタの利用目的が、一品ものを作ることや、製作者本人の趣味であった時代は、使用頻度による寸法精度（それも、縮小率 1%に満たない今回のケースのようなもの）の低下は問題になりづらかったが、論文趣旨でも述べたように、造形精度の向上と造形時間の短縮を始めとした 3D プリンタの進化によって、製品製造や小ロット生産の現場へ展開されることを考えると、解決すべき課題の 1 つになる。具体的には、これによりアセンブリの過程でパーツ同士が噛み合いにくいないし、完全に噛み合わない時、製品の品質を均一に保障していく時に難しくなる。

上記の経験から、3D プリンタの経年劣化（例：ゴムベルトの伸び）によって生じる寸法誤差を検出し、補正する仕組みが必要であると考えた。このように、指定した位置にヘッドが移動しているのかをプリンタで確認する方法には、フルクローズド制御とセミクローズド制御の 2 種類がある。3D プリンタに使用される現状のステッピングモータを想定した場合、セミクローズド制御ではエンコーダで脱調による位置偏差を検出し、ずれた分のパルスを補正する方法があるのに対し、フルクローズド制御では何かしらの方法（レーザー等）でエンドエフェクタ（=ノズル）の位置を検出し、位置フィードバックをかける方法があげられる。両者の違いは、セミクローズド制御はモータ自身が指定位置に達したと判断した場合でも、他の要因（ベルトの伸び）により最終的な位置偏差が生じる可能性があるが、フルクローズド制御では、その点も考慮して位置偏差をなくすことが可能である。一方、シカゴ大学では、ファームウェア側にフィードフォワード制御のアルゴリズムを組込み、造形時、高い加速度を与えた際に発生するモータの構造振動が引き起こす指令軌道への追従誤差を、FBS（filtered B-Spline）を用いたフィードフォワード制御で低減する研究がなされている。これは、前述した、セミクローズド制御及び、フルクローズド制御で行うものではなく、また、今回のゴムベルトの劣化による出力物の縮小の直接的解決にはならないが、これまで造形命令通り動くだけの機械だった 3D プリンタが、トラッキングエラーを自動制御により最小化する機械に変化していく未来を示唆しているとも捉えられる。自動制御がこれからのファームウェア側の拡張領域の 1 つになるだろうが、それに伴うソフトウェアの基礎的技術に興味を持ったのが、本研究で 3D プリンタ用ソフトを製作することになった動機である。

一方、本研究で製作してはいないが、全てのエラーを自動的に解決しようとするのではなく、製造に 3D プリンタを導入する際のマニュアルを作るというような、手作業でのアプローチの可能性も探求せねばならない。樹脂製の製品を量産するのに、現在は金型を利用するが、それにも耐久限度があり、大抵の金型が、CAD 通りに成形できる限度数（保証ショット数）を定めていている。それを 3D プリンタに置き換えて考えると、正しい寸法で出力ができる回数の目安や、消耗品（ヘッドが XY 方向に動くためのゴムベルトなど）を交換するタイミングのマニュアル化が最低限必要になるということになる。

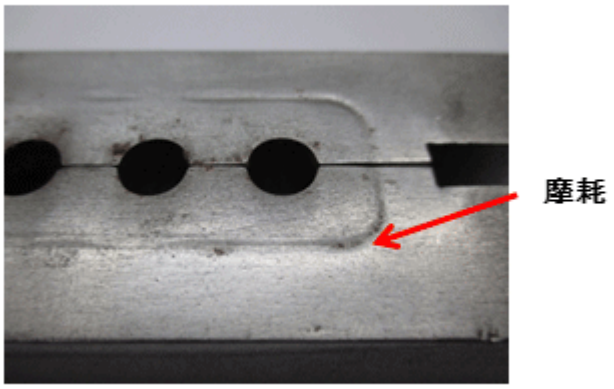

図 4　ショット数や成形材料により、物理的に金型が摩耗した写真。成形材料の中にガラス繊維などの硬い素材が含まれる場合、消耗は早くなり、メンテナンスや再度金型を作る必要が出てくる。

次に、デスクトップ 3D プリンタは本体だけでなく、利用可能な素材の分野も発展してきているので、それについても述べる。ABS や PLA など、従来のプラスチック成型品に使用されてきた素材だけでなく、生体適合性のある柔らかい素材(FABRIAL)や、温度帯の変化により形状を記憶する素材(SMP)など、新たな物性をもつフィラメントの登場は、医療やアート分野などでの 3D プリンタの可能性を更に拡大させることになった。ただし、各々の素材毎に、最も安定して出力できるパラメータは異なる。開発元の企業は、そのパラメータの参考値を提示してはいるが、3D プリンタは周囲の温度や湿度により造形精度が左右されることから、極端な話、日毎に最適値は変動するのである。よって、ある時は成功したパラメータが、別の日には失敗することも稀ではない。特にもこれら新しい素材の、印刷環境は一層デリケートで、一見造形が簡単にみえるシンプルな形状でも、数えきれない失敗の上にようやく成立している。そして、その日毎に変わる最適なパラメータの発見は人力で行われ、かなりの時間を要するのである。

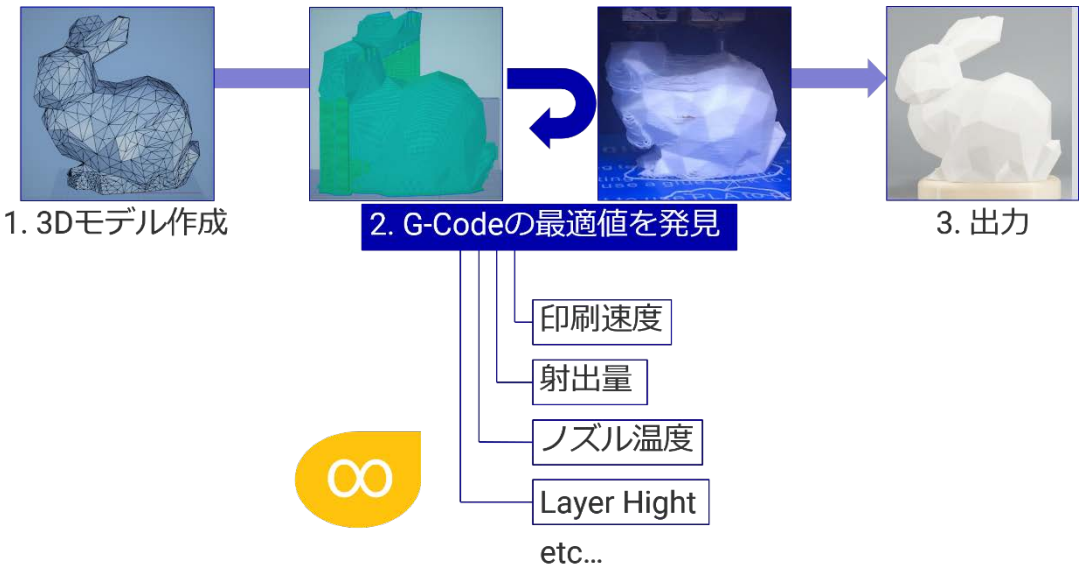

図 5　3D プリンタと聞くと[1]の作業から[3]の結果にすぐ結びつくと想像されがちだが、それは機能や素材を制限したプリンタに限られる。巨大もしくは極小の物の製作ないし、新素材を使うとなれば、最重要なのは[2]の過程であり、最も多くの時間を要する。

筆者の研究室でも「図 5」にある素材を用いた製作が行われており、ユーザー実験用、或いは展示用の一品を生み出すための、大量の試行錯誤が観察できる。対象ユーザーが身近に存在することはユーザーインサイトの発見に繋がりやすくなると考え、今回開発したソフトの主な機能は、パラメータの調整⇄出力間におけるイテレーション速度の向上に定めることにした。

図 6　（左）がキョーラク(株)の SMP55　（右）が JSR(株)の FABRIAL。

1.2 関連研究

1.2.1 既存のソフトウェアについて

最初に 3D プリンタで出力を行うとき、使われるソフトウェアを整理する。

図 7　3D モデル作成から、出力が行われるまでの流れと、各々の過程で使われるソフトについて。

「1」で、オリジナルのモデルを作る時に必要なのが CAD である。機構設計などに用いられるソリッドモデリングや、外装などデザインに用いられるワイヤーフレームモデリング及びサーフェイスモデリング、フィギュアの作成などに用いられ、粘土のようにモデルをこねたり削ったりして形を作るスカルプトモデリングと手法は様々で、代表的な CAD には、「SOLIDWORKS」、「Rhinoceros」があり、スカルプトモデリングソフトでは「ZBrush」がメジャーである。「2」にデータを渡す際は、.stl 形式に変換する必要がある。

「2」のホスト App は、3D プリンタのメイン基板に、シリアル通信で造形命令（G Code）を送信したり、「1」で作成した STL データを、3D プリンタで解釈可能な形式に変換（以下スライス）したりする役割を担う。3D データのビューアーや GUI が搭載されている上、3D プリンタ本体の X,Y,Z 各軸を別々に動かせ、ノズルやベッドの温度のみ上昇させることも可能なので、造形時のみならず、メンテナンスや故障個所の特定で使用されることも多い。また、フリーソフトも充実しており、代表的なものに「Repetier Host」、「Cura」、「Slic3r」などがある。有料ソフトでは「Simplify3D」が代表的で、通信したい 3D プリンタの造形エリアの設定が容易である他、造形部分毎に異なるスライス設定を割り当てることが可能であるなど、フリーソフトにはない機能が多く、ユーザビリティが高い。また、MakerBot(株)や Zortrax(株)など、自社の 3D プリンタ専用のホスト App を提供しているケースもある。これらのソフトで変換されたデータは専用のプリンタでのみ解釈可能であるが、スライサーの設定条件を限定しているため、造形精度が

高く失敗しにくいのが特徴である。各々のホスト App については後に詳しく記述する。

「3」のファームウェアは 3D プリンタを制御するプログラムであり、通常、購入時にメイン基板に書き込まれている。「2」作成したスライスデータを元に実際に造形を行うのが主な枠割である。また、RepRap 系統の 3D プリンタなら、GitHub（https://github.com/）上などで公開されているオープンソースのファームウェアをダウンロードし、ユーザー自身の目的に応じて書き換え、書き込み直すこともできる。主なオープンソースのファームウェアとして、「Marlin」、「Repetier firmware」などがあり、各々のファームウェア毎に仕様が異なるため、ユーザーは自分の目的にあったものを選ぶ必要がある。

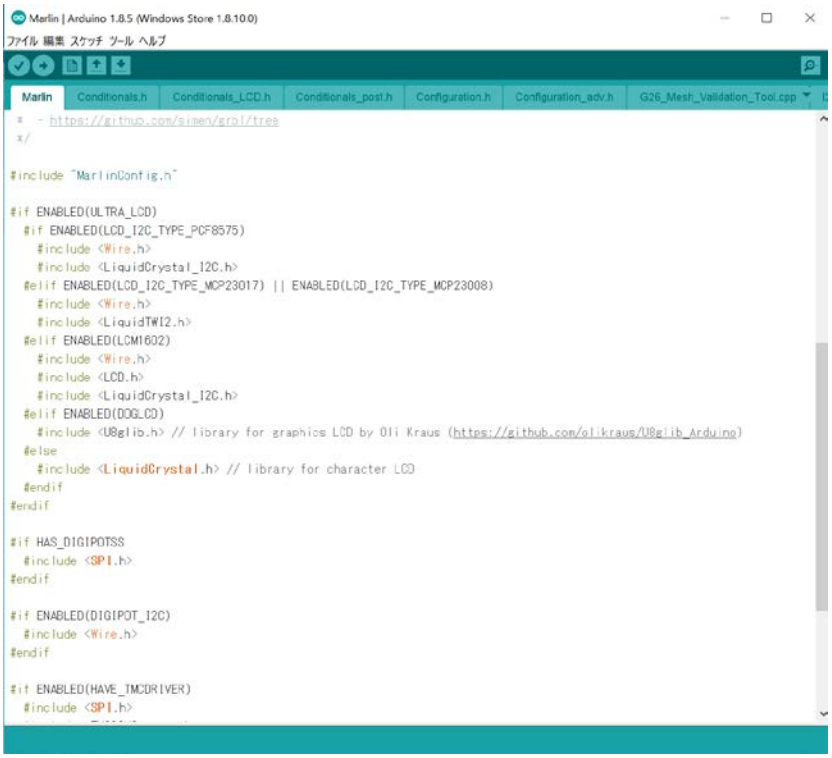

図 8　Marlin のソースコードを Arduino IDE で開いた様子。

なお、今回製作したソフトの立ち位置は「2」のホスト App に近いため、以後は既存のホスト App の特徴について記述してゆく。

Repetier Host　version2.0.1（フリーソフト）　RepRap 系統の 3D プリンタとシリアル通信できる。スライサーの役割も内包しており、その機能は Cura、Slic3r に依存する。GUI 上のアイコ

ンにマウスオーバーすると、何の機能を持つコマンドか説明されるため、初心者でも使い易い。シリアル通信で GCode を送信する場合、最新版では、「送りレート」、「ノズル温度」、「ヒートベッドの温度」、「ファンの回転数」をリアルタイムで変更できる。送りレートとは、印刷速度のことで、出力されていない時のヘッドの移動速度もこれに含まれる。ただし、速度を変えても、吐出量が同期して自動的変更される訳ではないので、基本的に印刷速度を早くすることは推奨しない。というのも、例えば速度を 100%から 150%に上昇させた場合、吐出量は 100%のままなので、射出幅が設定した値よりも狭くなり、造形物の強度を低下させるからである。(現行版へのメジャーアップデート前のバージョンでは、吐出レートも変更できた。)

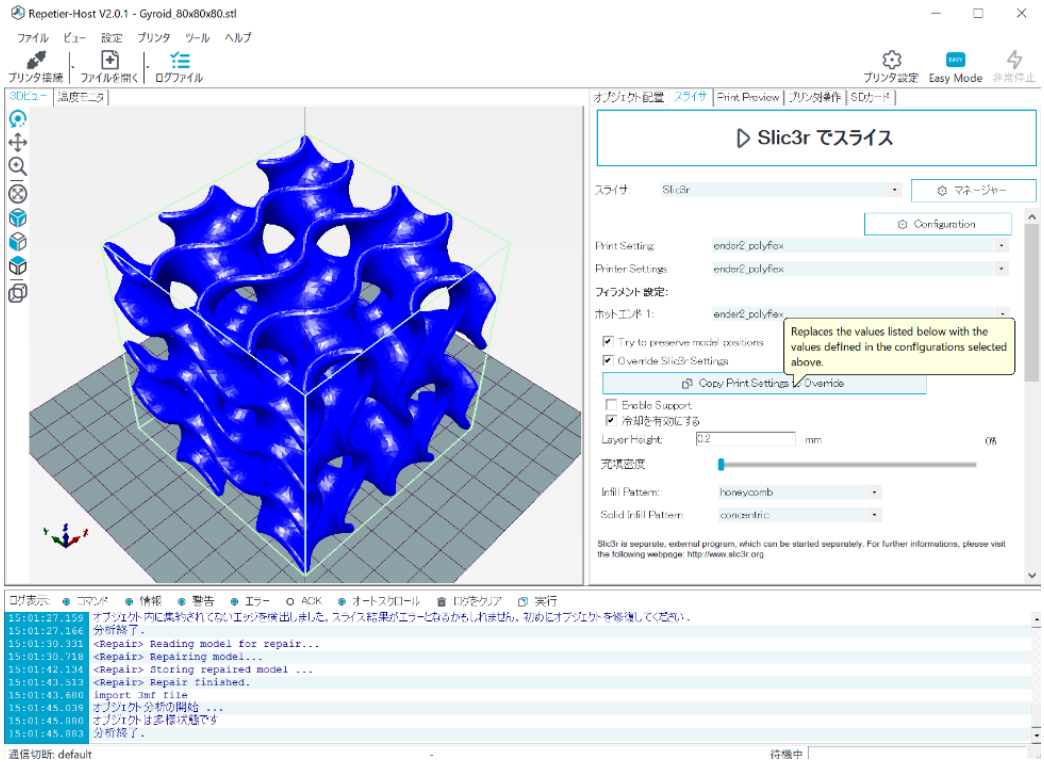

図 9　Repetier Host にモデルをインポートした様子

Cura　version3.1（フリーソフト）　こちらも RepRap 系統の 3D プリンタとシリアル通信できる。スライサーも搭載されている上、セットアップ時に Ultimaker(株)の 3D プリンタ専用のホスト App に設定することもでき、Ultimaker と併用すれば、造形品質はかなり安定する。特徴は、スライス設定時に参照できるマテリアルデータセットの数である。45 種類の異なる素材のスライス参考値（推奨印刷温度、リトラクトの距離と速度、フィラメントの密度など）が提供されており、利用したい素材がその中に存在する場合、その参考値をベースに最適値を模索す

ればよい。

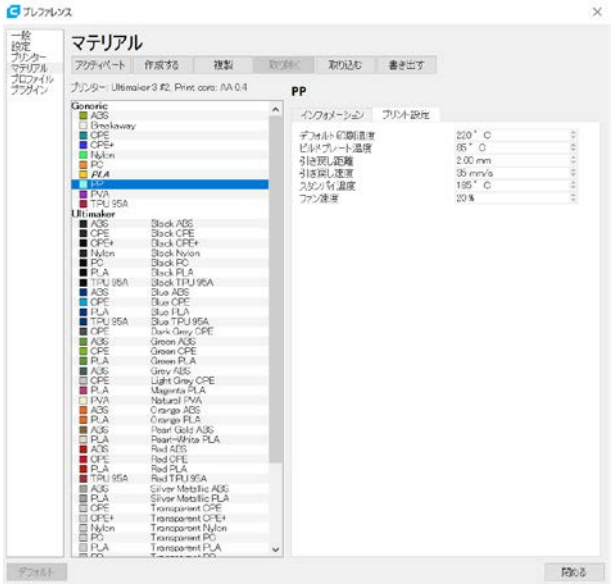

図 10　Cura のマテリアルデータセット画面。

Simplyfy3D　version4.0.0（有料ソフト）　有料ソフトであり、RepRap 系統に加え、MakerBot 社の 3D プリンタともシリアル通信が可能である。利用したい 3D プリンタが、Simplyfy3D の校正アシスタント内に存在する場合、自動的に造形エリアや、そのプリンタにあった推奨印刷速度の設定に変更してくれる。現存するホスト App の中で、スライスで設定できる範囲が最も広く、造形モデルの各部分に異なるスライス設定を割り当て可能な他、ノズル径よりも細い箇所が存在する部分に対して、ダイナミックに射出を絞り、また必要であれば解放し、形状に沿った自動射出コントロールをするなど、フリーソフトにはない機能が豊富である。3D プリンタで出力することをあまり考慮せず 3D モデルを作成した場合（多くのガイドラインでは、3D プリンタで出力するモデルは、どの部分も 1mm 以上の厚みを必要するとされている）も、限りなく 3D モデルに近い形状で造形できる上、小さい物や、内部に細かい形状を必要とする物をより正確に造形可能にしている。また、サポートを生成する箇所を手動で追加でき、FDM 3D プリンタの性能を最大限に活用することができるソフトである。シリアル通信で造形する時は、印刷速度と射出量をリアルタイムで変更できるが、Repetier Host 同様、2 つは独立的している。

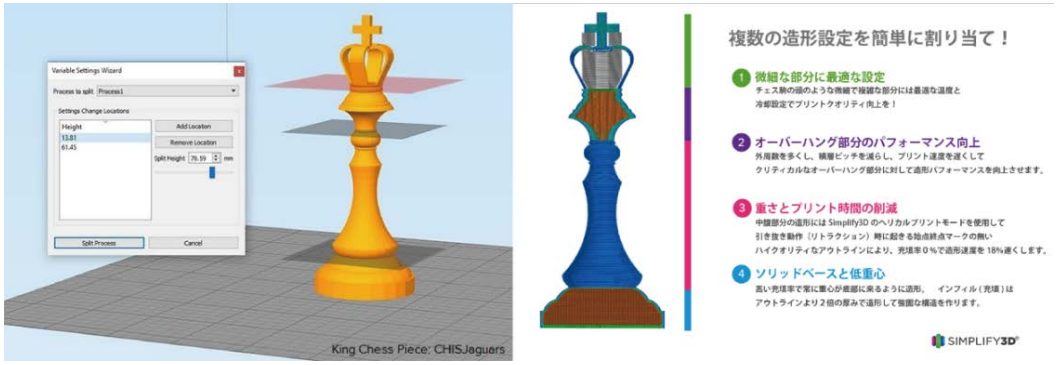

図 11　（左）が Simplyfy3D で造形物の部分別にスライス設定を変えている様子。（右）が各々どのような目的でスライスしたかの説明。造形物の上方は形状が細かく、中空部分も多いため、綺麗に造形するために印刷速度を落としている。また下方は物体としての安定性を考慮し、内部充填率を上げることで、他部分より重みをつけている。

MakerBot Print　version2.7（.x3g の拡張子のみ生成可）　.x3g 形式のファイルを解釈可能な 3D プリンタのみ対象にしている。シリアル通信で印刷することも可能。スライスのインフィル設定時、「MinFill」（構造を維持し、内部充填率を限りなく少なくする）を選択すると、造形時間を通常より 30%短縮でき、大きいモデルを少しでも早く形にしたい人に適している。MakerBot 社が販売している素材と併用することで、造形質を安定させることができる。

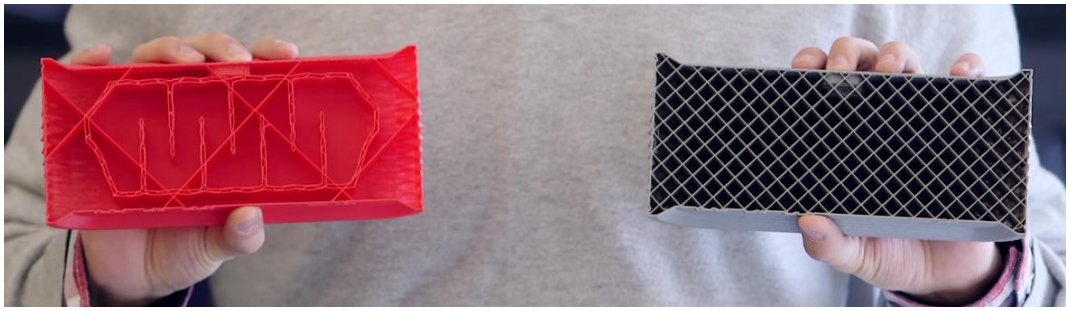

図 12　（右）が通常の設定、（左）が MinFill。内部充電率が少ないことにより、造形時間も短縮されるのに加え、材料も節約できる。

Zortrax　version1.12.2（Zortrax 専用ソフト）　Zortrax 社の 3D プリンタ専用のスライスソフト、以前はスライス設定時、Zortrax 社が販売している素材のみ選択できたが、現在「External Materials」が追加され、自社以外の素材用にスライス設定をカスタマイズ可能になった。先に説明した 3 つのソフトと違い、スライスの設定可能範囲は狭いが、Layer Hight で選択できる数値をあらかじめ絞り、造形スピードを初期値から何%増減させるのかをスライダーで選択させる UI 設計

など、造形の質を担保しつつ、ユーザーの自由度も拡張させている。シリアル通信で印刷することはできない。

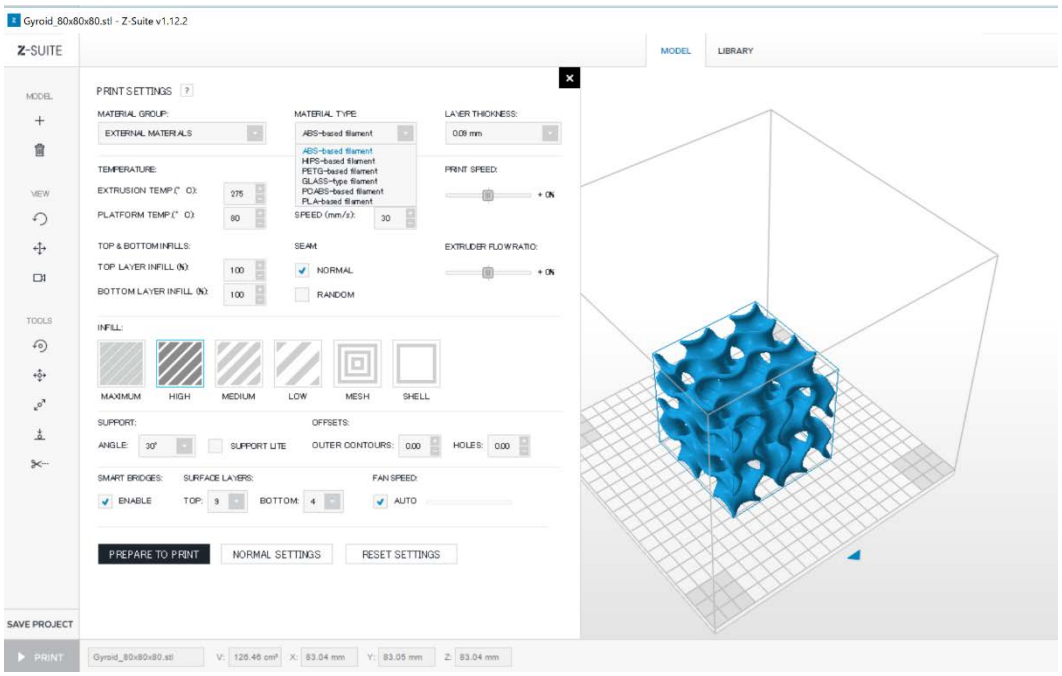

図 13　Zortrax 社のホスト App に 3D モデルをインポートした様子。他ホスト App と比べ、グラフィカルで洗練された UI が特徴的。

1.2.2 自動制御について

ここで取り上げるのは、フィードフォワード制御を活用した 3D プリンタの先行研究である。まず、フィードバック制御とフィードフォワード制御について説明する。フィードバック制御は、外乱が生み出す出力誤差をゼロにすることができるが、外乱がシステムに入力され、外乱によって乱されたシステム出力が観測された際にはじめてフィードバックが可能になる。よって、外乱を迅速に抑制する応答性能が本質的に低いことになる（=つまり環境の変化に受動的に反応する）。これに対し、フィードバック制御器にフィードフォワード制御器を前置した場合、外乱に対する抑制効果が高まる。「図 13」は Honeywell(株)のプロセスシュミレータ UniSim Design で作られた、50℃の温水タンクに突如冷水が入った場合の、タンク内の温度制御分布の様子である。

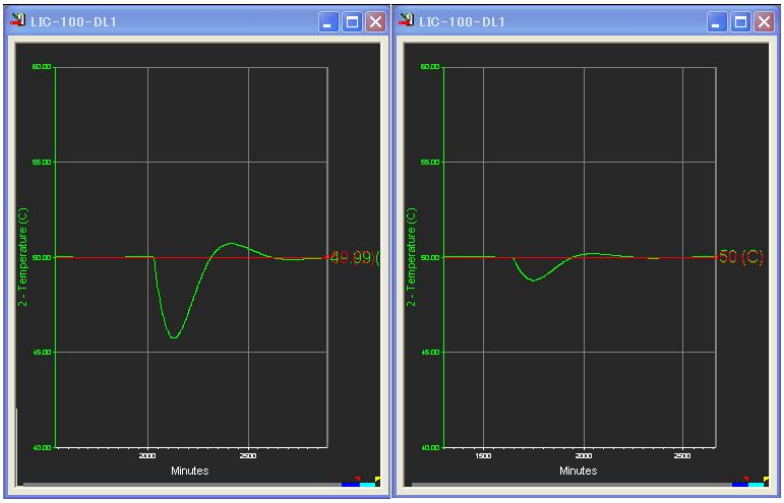

図 14　（左）がフィードバック制御　（右）がフィードフォワード制御。右の方が変化の振れ幅が小さく、タンク内の急激な温度変化に対し、温度帯を左よりも早く設定値に戻せている。

次に、自動制御を活用した 3D プリンタの先行事例についてである。

アメリカのシカゴ大学機械工学科の学生と Chinedum Okwudire 准教授は、論文、「Algorithm allows 3-D printers to “read ahead” of their programming to boost speeds.」内にて、造形時間を短縮しても、造形精度が落ちないソフトウェアアルゴリズムを開発するために、フィードフォワード制御を用いたと述べている。印刷速度を上げた時に造形精度が落ちる理由の 1 つに、速度が上がるだけ、ヘッドの振動が大きくなることがあげられるが、このアルゴリズムでは、ヘッドの「軌道」

を予測し、構造的な振動が起こすトラッキングエラーを最小にすることで、造形品質を保ちつつ、印刷速度を上げることに成功したと述べられている。先の説明では、フィードバック制御に組合せて用いる（=補完するための）フィードフォワード制御器について述べたが、こちらの研究では、フィードバック制御を用いずに、フィードフォワード制御のみを用いて、追従誤差を抑制している。彼らがフィードフォワード制御を用いている理由は、システムへの外乱に対する応答性能・抑制性能を高めるというよりは、本質的にステッピングモータがオープンループの構造であることに起因している。またオープンループのみで制御器を構築できれば、外部に出力を監視するセンサを設ける必要もないため、コストも低く、制御自体も簡単になるという理由も考えられる。

図 15　印刷速度を上げても造形物が形状を保ったまま出力されている様子。アルゴリズムは既存の3D プリンタのファームウェアに組込むだけで利用でき、ハードウェア側のアップグレードも必要ない。

1.3 研究目的

既存のソフトウェアの機能をまとめていく中で、シリアル通信で GCode を送っている最中に変数を変更できる機能が搭載されているホスト App は既に存在していることが分かった。しかし、マテリアルの最適値を探したり、3D プリンタの機構的に、造形が難しい部分の最適なスライス値を探したりすることに特化したホストはまだ存在していない。そこで本研究では、リアルタイムに GCode 内の変数を変更でき、且つ、新しい素材の調整や、通常よりもはるかに高い造形精度を出すために時間を費やすユーザーに向けた機能を実装したホストの開発を行うことにした。同時に、ただ実用的なソフトだけを目指すのではなく、3D プリンタを利用した表現領域の拡張に貢献できるような、今回製作したソフトウェアだからこそ生み出せる造形物の形やテクスチャーについての可能性の検討も行うことにした。

また、3D プリンタの失敗の瞬間をノズル付近から撮影し、観察と分析を行うことで、今後、プリントの失敗時に、具体的にどのような対策を行えばそれが解決されるのかを明らかにしようと試みた。

ホストの開発と、失敗の瞬間の撮影の 2 つは、連続性がない研究に思われるかもしれないが、1 章.でも述べたように、今後自己学習型の 3D プリンタや、ファームウェアにコードを追加するだけでフィードバック制御が可能になるようなプログラムが公開された際、特に後者において、2 つの技術は大きな役割を担うと考えている。

今回の研究内容だけで、3D プリンタに関する何か大きな問題を解決することができなくても、今後、この中の記録が何かしらの問題解決の糸口になってくれればと切に願う。

2. 提案・実装

2.1 3D プリンタの制御ソフトウェアの拡張のために

2.1.1 対象となる 3D プリンタ

ターゲットとする 3D プリンタは、「RepRap 系統」のプリンタとした。RepRap とは、自己を構成する大半の部品を自身で製造できる 3D プリンタを作るプロジェクトで生まれた 3D プリンタである。また、その進化の過程で派生型として生まれ、個人向けにメーカーから販売される 3D プリンタのことを、ここでは「RepRap 系統」と呼ぶ。これらを対象に選んだ理由は、世の中には既に大量の RepRap 系統の 3D プリンタが販売ないし複製されており、仮に、今後私の作ったホスト App が有用性を持ってきた場合、恩恵を与えられるユーザーが最も多いと考えたからである。また、G Code を解釈可能で、状況によりファームウェアを書き換えることができるので、ソフトウェアがエラーを起こした時など、解決の手段に縛らず、柔軟に対応できる点も考慮した。

.x3g の命令文で動く 3D プリンタ（MakerBot や QIDI Tech）も一度は視野に入れたが、バイナリデータの.x3g を.gcode として変換できても、シリアル通信で送っている命令文を即時的に修正するには、再度.gcode に.x3g に変換してから送信する必要があり、入力と出力間にタイムラグが発生する可能性を考慮し、RepRap 系統の 3D プリンタに対象を絞ることにした。

本研究においては、HICTOP Prusa i3 シリーズの「3dp-17bk」と「3dp-21」を使用した。

2.1.2 射出量、速度、レイヤーハイトの即時変更

G Code を即時的に変更しながら出力する時、まず、どの変数を動かすのが最適かを考えた。例えば失敗の瞬間をカメラで観測しつつ出力すると仮定し、失敗映像から、G Code のどの部分を修正すれば良いかを推測できるとした場合、Simplyfy3D の 3D Print Quality Troubleshooting Guide が参考になった。ここでは頻発しやすい 3D プリントの失敗例の写真と、それが一体何が原因で起きているのかについて解説されている。「図 16」では出力後にレイヤーから亀裂が入るケースについて言及している。この場合は、レイヤーピッチを狭めるか、造形時のヘッドの温度を上げることで解決できると書かれている。他にも 27 種類の、3D プリンタユーザーが直面する失敗についての解決策が記載されており、随時更新もされている。「図 17」はその失敗のパターンを、ソフト側とハード側の原因別に分類した図になる。1 つの失敗は複合的原因により形成されているケースが多いので、今回は主因になりやすい原因がハードとソフト、ど

ちらに存在しやすいかで分類した。図をみれば分かるように、ソフト側で解決できる問題の方が多い。

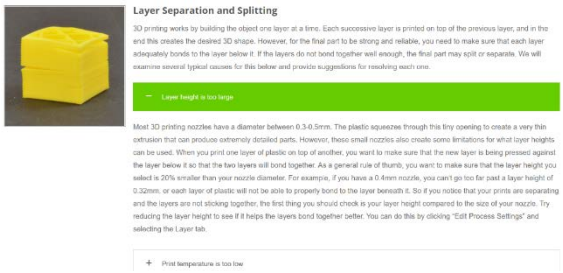

図 16　Simplyfy3D が公開している Print Quality Troubleshooting Guide の一例。

図 17　（上）Simplyfy3D が公開している、3D プリントの失敗群。（下）（上）の失敗群を、筆者がソフト側とハード側のどちらに主因がある場合が多いかを整列した図。ソフト解決できる問題の方が多いことが分かる。

更に、ソフトウェア側で解決できる問題が、具体的に何が原因で発生しているのかに分類し、それぞれ、スライス時にどのパラメータを変更することで改善できるのかを図示したのが「図18」である。サポートは変数というよりも 3D モデルに応じて自動的に生成されるものなので、今回対象とする変数からは外している、またノズル温度もファームによっては印刷を一時停止して温度が上昇するまで待つ必要があり、変更した際の動作を一律に保証できないので、対象から外している。その 2 つを除くと、プリントスピードと射出量/幅、レイヤーハイトが原因の多くを占めていることが分かる。

このことを考慮し、上記 3 つの変数をリアルタイムで変更できることを第一目標とした。

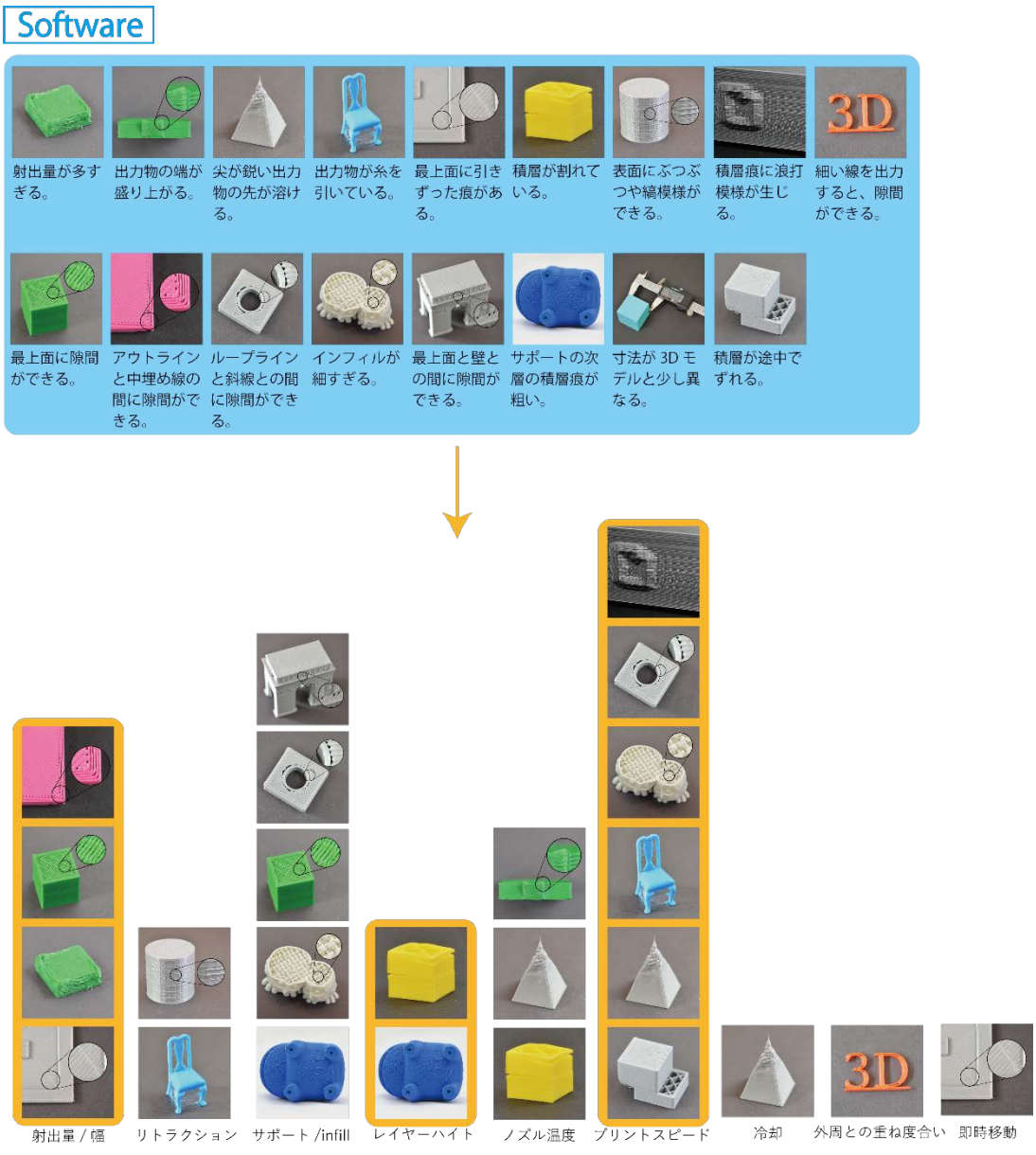

図 18　筆者が、ソフト側で解決できる問題を、失敗原因別に整列した図。

2.1.3 機能と UI の実装

リアルタイムで G Code を変更する以前に、G Code の構造がどうなっているかを確かめる必要があった。「図 18」は 1 レイヤーをプリントする際の、実際の G Code（右）と、それぞれの行がどのような役割をもっているかを解説したもの（左）である。基本的に同じ、「図 19」（右）にあるようなパターンの繰り返しであることが理解できた。また、今回対象とする変数であるプリントスピードと射出量は、G Code 上ではそれぞれ、「F」と「E」として定義され、F(Feed Rate)は印刷に関わる全ての速度を、E(Extrusion)は射出量と幅をコントロールしている。E は大きな変動は無いがほぼ毎行指定されており、F は大きく変動するが指定される間隔に少し間があるのもみてとれる。

また、1 行目で F 値を設定し、s 行目でも F 値の設定が行われるが、3 行目ではなされず、値の指定がない場合は、前の値を継承する仕組みであることも分かった。

G Code についての大枠の理解ができた所で、具体的な即時変換の手法を考えたが、こちらは変更したい値を上書きすることで実現できると考えた。例えば速度現状の 50%にしたいとする、その時は命令を下した後、G Code 中の「F」以降の数字を、0.5 を乗算した値に変更してあげればよい。E 値、レイヤーハイト（図中には記されていないが「Z」で表される）についても同じ考え方で実装できると考えた。

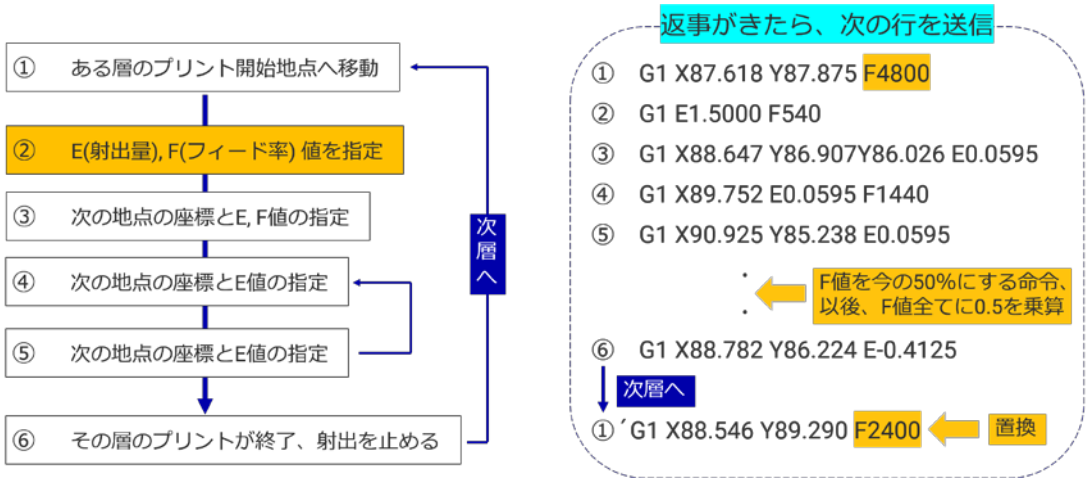

図 19（右）G Code を即時変更するための仕組み。（左）右の G Code の各行の命令内容について。

ただ、この仕組みへの懸念点もあった。Simplyfy3D では造形物の部分毎に個別のスライス設定

を割り当てられることは説明したが、例えば仮に絶対にその速度で印刷する必要のある部分が存在したら、その部分のF値まで変更されてしまう点である。筆者はそのような場面に遭遇したことはなかったが、細部まで考慮したスライス設定に対し、例外なく変更を加える仕様に、些か短絡的すぎるのではと感じたことは事実である。ただ、概念実証もしていない段階で、まだ起こっていない不具合に怯え手を止めることは避けたかったので、F・E・Z値を即時変更する仕組みを一度実装することにした。

次に、GCodeを3Dプリンタに送るプログラムの実装に移った。ユーザビリティのことまで考えると、UIを作り込む可能性があったため、Processingを開発環境とした。第一に、Processingからシリアル通信で3Dプリンタを動かすことからはじめた。
Processingから直接RepRap機を動かすソースはシンプルで、最低限、以下のソースコードだけで駆動させることができるが、6行目、myPort~からはじまるコードの、「COM3」と「250000」の2つの部分は、各自のプリンタのCOMポートとシリアルポートに変更する必要がある。

```
import processing.serial.*;

Serial myPort;
public void setup() {
   size(   800, 600, OPENGL   );
   String portName = Serial.list()[0]; // This gets the first port on your computer.
   myPort = new Serial(   this, "COM3", 250000   );
}

public void draw() {
}

void keyPressed() {
   if ( key == 'h' ) {   myPort.write("G28¥n"); println("here");   }
   if ( key == 'x' ) {   myPort.write("G1 X10 Y0 Z0 F1000¥n");   }
   if ( key == 'y' ) {   myPort.write("G1 X0 Y10 Z0 F1000¥n");   }
   if ( key == 'z' ) {   myPort.write("G1 X0 Y0 Z10 F1000¥n");   }
}
```

図 20　Processingから RepRap機を動かすために、最小限必要なソースコード。

次にUIの製作に移った。

確認になるが、本ホスト App にはスライサーの機能は搭載していないため、他ホスト App でスライスし生成した G Code を使用する。それをシリアル通信で 3D プリンタに送るまでのフローは「図 21」に示した。

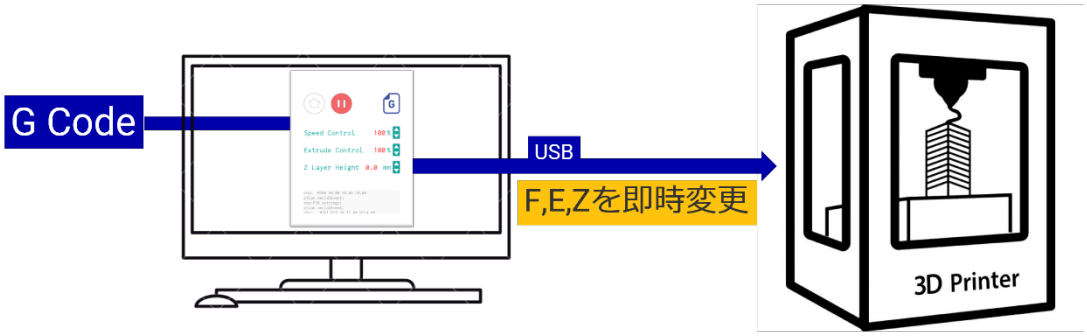

図 21　G Code を本ソフト App 経由で 3D プリンタに送るまでのフロー。

はじめに実装した UI が「図 22」である。左上にある家のマークのボタンが、X,Y,Z 軸をホームポジションに戻す命令で、その右隣りの赤いボタンが、G Code の送信を一時に停止する命令である。右上の「Import G-code」で、.gcode のファイルを 3D プリンタに送信できる。また送信した G Code は下の「start up Direct Code」と書いてあるテキストエリア内に表示される。

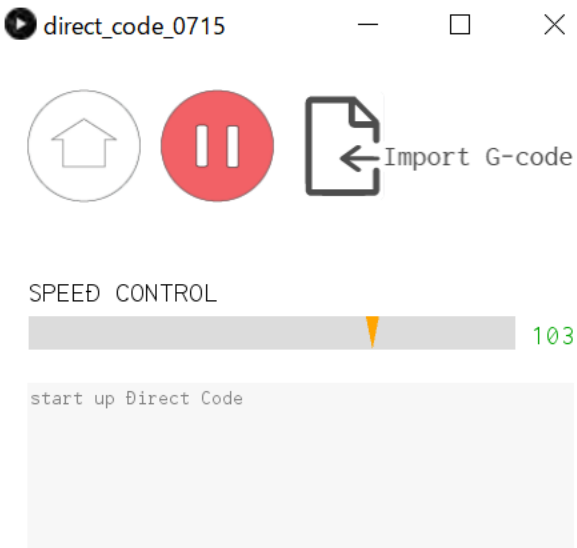

図 22　今回製作するホストのファーストプロトタイプ。

即時的に G Code を変更するためには、プログラム側が、送信する G Code を常に監視する必要があった。USB 経由で 3D プリントをした経験のある人ならわかるだろうが、造形中に USB を抜くと印刷がとまる 3D プリンタがほとんどである。この点から、ホストが一方的に G Code

を送りつけているのではないことがわかる。もし、ホストがファーム側からの応対を待たずして、G Code を送っているなら、それらはバッファに格納されるため、造形中に USB を抜いても印刷は継続されるはずだからである。ならば、シリアル通信時、ファームとホスト間では何かしらの対話が行われており、本プログラムでも G Code を一行送った時に、ファームからの返答が帰ってきたら、次の行を送信する仕様にしてある。これによりホスト側で G Code を常に監視する仕組みを実現した。

「図 23」が実際に造形ファイルを送信した際に、ホスト-ファーム間で送受信されている命令を表したものである。1・3 行目に「ok」の文字がみえるが、これがファームからの応答が来たことを示しており、返答が来次第、ホストが次の行を送信している。

```
ok
G1 X77.285 Y79.234 E0.0373
action serialEvent:
ok
G1 X77.372 Y78.975 E0.0383
```

図 23　実際にテキストエリア内に送信した G Code と、ファームからの返答の有無が表示されている様子。

ただ、G Code 中にはコメント文と呼ばれる「;」からはじまる行があり、コメント文は、G Code 生成時のスライス条件などを人が再確認するためにかかれているので、ファーム側が解釈することができない。故に命令文としての機能はなく、ファームも応答することができない。本ポストの、ファームから応答が来次第、次行を送信する仕様上、このままではコメント文にさしかかると印刷が継続されないので、「図 24」にあるよう、正規表現で、「;」からはじまる行を「」（空）に置換し、行の内容が「」の場合、次行を送信する仕様にすることにした。

```
gcode[i] = gcode[i].replaceAll(";.*$","");

if (gcode[i].trim().length() == 0) i++;
```

図 24　1 行目が、「；」を「」に置換する命令で、2 行目が、「」の場合、次行を送信する命令。

F 値を変化させる方法については、前述した通りで動いたので割愛する。同じ原理で E 値、Z 値（レイヤーハイト）も変更できるようにし、最終的に完成した UI が「図 25」である。

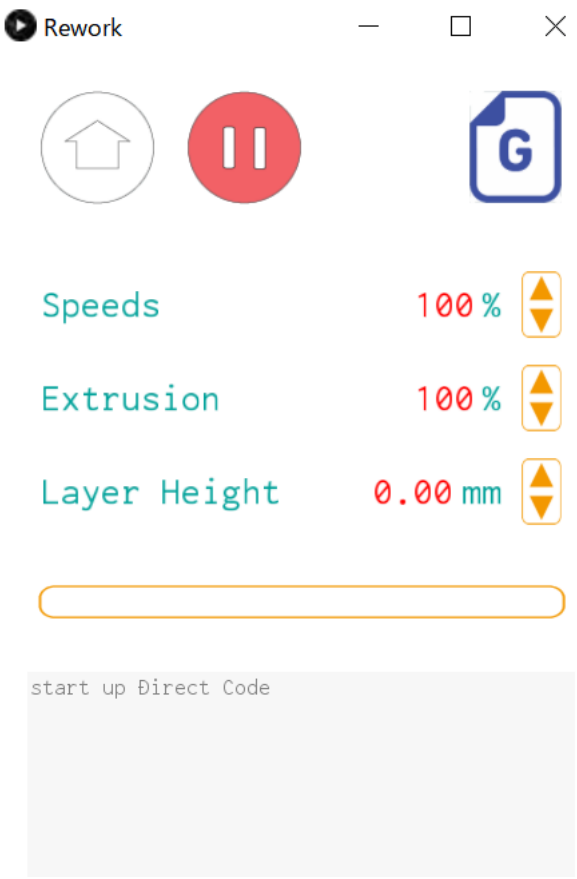

図 25　最終的に完成したホストの UI。現在の印刷の進行度合いが分かるよう、Progress Bar を追加した。

「Speeds」が F 値、「Extrusion」が E 値、「Layer Hight」が Z 値をそれぞれ変更できる。Speeds と Extrusion が現在の値を基準とした「%」変更できるようにしたのは、本ポストの仕様上、直接 F 値を入力する形にした場合、全ての F 値が等しくなってしまい、スライス時、意図的に設計した印刷速度の振れ幅を消滅させてしまうからである。他の印刷速度を変更することができるホストでも、変更単位には「%」を使用しており、同様の理由からであると考えられる。

また Layer Hight については、乗算にすると、レイヤーが進むにつれ、Z 値が 2 次関数的に増えることになり、層と層との間を接着することが不可能になるため、加算する形で置換を行った。

2.1.4 周期的制御の追加

ここまで、本研究の目的に沿ったアプリケーションの開発を進めてきたが、この章では、他ホストではできない、本ホストならではの制御について述べる。そもそも、1 章では、従来の樹脂製のプロダクトや介護用品など、「使われるもの」を作るための工作機械としての 3D プリンタ

の発展について述べてきたが、一方で、ベッドを振動させながら造形することでテクスチャーを付与したり、あえてレイヤーハイトを大きく設定したりして、積層を不規則なものに変えてしまうなど、想定外の使用方法で表現領域の拡張を試みる者もいる。結論から言うと、本ホストならではの制御というのはサーフェスのコントロールなのだが、はじめに、同じく 3D プリンタによるテクスチャーのコントロールの事例を述べる。またここで取り上げるのは、3D モデルや G Code 自体を変更してテクスチャーを付与した事例ではなく、元々スムームな表面に対し、スライス時や出力時の過程でテクスチャーを加えた例とする。

Expressive Fused Deposition Modeling　明治大学の宮下芳明氏と高橋治輝氏による Expressive Fused Deposition Modeling (CHI2017 Paper)は、あえてノズルの位置をベッドから離し、その分射出量を増加させることで、テクスチャーに変化をもたせた研究である。

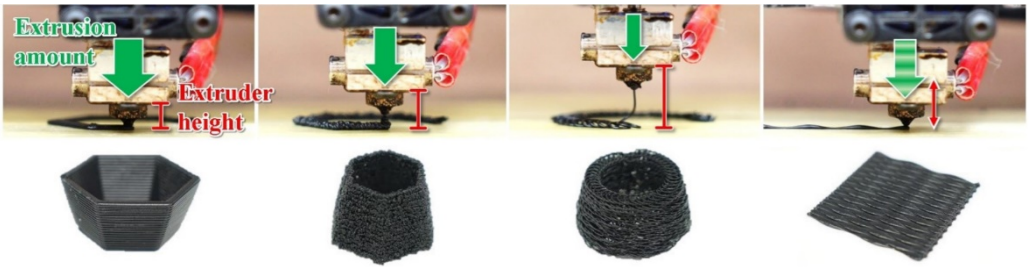

図 26　またこの結果から、レイヤーハイトと射出量の変化により、どのような射出がなされるかをグラフ化し、再現性をもたせる試みもなされている。

Solid Vibrations　Olivier van Herpt 氏と Studio Van Broekhoven により作られたもので、粘土を用いて陶器を 3D プリントする際、ベッドから音や音楽の振動を伝えることで、音波振動を模様として浮き上がらせた作品である。繰り返されるリズムが微妙に変わることで、模様にも変化が生まれるその様は、振動が固定化されたものとも言える。

図 27　Solid Vibrations の展示風景。

Velocity painting　1.2.1 章で紹介した Repetier Host 内に存在する機能である。スライス後、造形物の表面に任意の画像を UV マッピングすることができる。テクスチャーは印刷速度を変化させることにより再現されており、射出量は固定したまま、基準の印刷速度を 100%とした時、凹の部分を 150%、凸の部分を 50%で造形することで射出幅=厚みに変化をつけている。

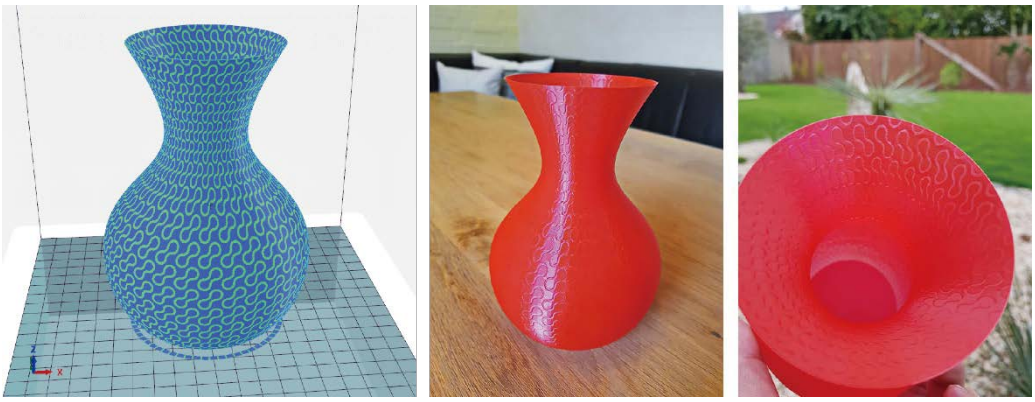

図 28（左）モデルに画像を UV マッピングしたものと、（右）出力されたもの。

筆者は上記の事例を参考に、ホスト側で周期的に射出量と造形速度をコントロールしてゆくことで、規則的な模様のパターンを生み出せるのではないかと考えた。「図 29」では「図 25」のホストに「Surface Control」という項目を増やした。その中の「interval [every]」は Speeds、Extrusion、Layer Height がそれぞれ指定された時、その変更条件を適応する間隔を指定する機能である。「図 29」で言えば、Speed_80%、Extrusion_400%を、① 5 行適応→②次の 5 行は不適応（両方 100%で造形）→①…というフローになる。

図 29　模様をつけるために開発したホストの UI の一部

「図 30」は本ホストを用い製作した造形サンプルの写真である。造形の際、3D プリンタには 3dp-21 を使い、フィラメントは T-grass を用いた。T-grass は PET が原材料だが、ガラスのような質感をもった造形物を作りだすことが可能で、ノズル径を太くしたり、射出量を増やしたりすることで、よりガラスに似せることができる。「図 31」で、内側から光を照らした様子は、一見するとそれがプラスチックであることを忘れてしまう。

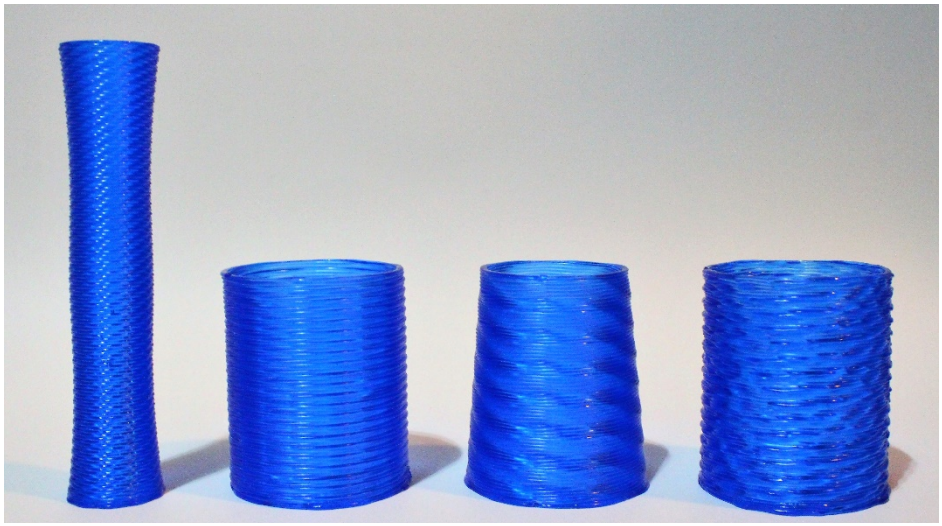

図 30　本ソフトで作った造形サンプル。ノズルは径が 0.8mm のものを使用した。

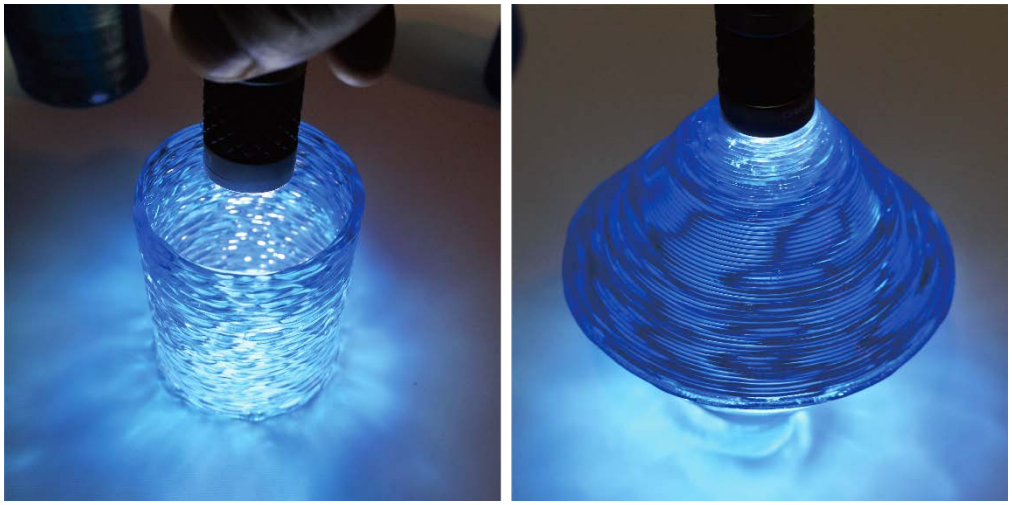

図 31　「図 30」のサンプルの内側を光で照らしたもの。

2.1.5 印刷終了時の G-code の保存

2.1.4 章でこれまでの話の流れから脱線し、表現の分野に内容が傾いたが、ここからはまた本来

の目的に軌道を修正する。既存のソフトウェアでシリアル通信を行い、速度や射出量を変化させることはできるが、変更後に G Code を記憶する機能はついておらず、自身がパラメータをどう変更したかはユーザーが記憶しておく他なかった。新しいマテリアルの最適なパラメータを発見する時に限ったことではないが、偶然見つかった良質な G Code の保存をソフト側が担ってくれれば、ユーザーの負担も軽くなると考え、本ホストを通じ 3D プリンタに送られた G Code を PC 内に自動的に保存できるようにした。2.1.6 章で記述するが、Github 上にある「Rework」と「Unevenness」の仕様では、デスクトップ上に「Overwrite_Gcode」と「Unevenness_Gcode」というフォルダが生成され、その中に G Code が保存されるようになっている。

2.1.6 オープンソース化と導入手順の共有

今回製作したホスト App は GitHub 上（https://github.com/mshdchb/Real-Time-Editing-Gcode）で公開されており、誰でもダウンロードすることができる。またその導入手順について Fabble（http://fabble.cc/masahide/realxtimexeditingxgcodexxxxx）に詳しく記述している。Fabble 内には、導入手順の他に、E 値、F 値、Z 値の変化量を変える方法についても書いてある。

2.2 接写を用いた 3D プリントの失敗の瞬間の観察

2.2.1 カメラを設置する 3D プリンタについて

今回、カメラの取り付け対象に選んだのは武藤工業(株)の MF-2200 であった。MF-2200 では研究室内で SMP の出力に頻繁に使用されており、SMP も失敗の多い素材であるという認識であったため、失敗の瞬間を入手しやすく本研究との相性が良いと考えたからである。

2.2.2 取り付けるカメラの選定

はじめに、プリント全体の様子を監視する Web カメラと、ノズル付近から射出の瞬間をとらえるファイバースコープの 2 台で撮影を行うことにした。遠隔から様子を監視、撮影した映像を自動でクラウド上に保存するなどしたかったため、2 台とも Raspberry Pi に接続する必要があり、USB 端子をもつ以下の 2 つのモデルを選んだ。

Web カメラには「Locicool C270」を、ファイバースコープには「SPYDERSX M-935」を使用した。2 台の簡単なスペックは以下の通りである。

メーカー / 機種名	SPYDERSX / M-935	Logicool / C270
用途	接写	広角撮影
固定場所	ノズル付近	3D プリンタの右上方
固定方法	冶具を自作（2.2.3 章にて詳細説明）	ゴリラアーム
動画の解像度	640 × 480	1280 × 720
最大 fps	~30 fps	~30 fps
最小焦点距離	3 cm	40 cm

図 32（左）M-935（右）C270

2.2.3 M-935 の固定用ジグの設計

まず、全体の形を作り込む前に、M-935 の各部分に対して、ジグがぴったり噛み合うような寸法を見つける。モデリングと 3D プリントを繰り返して、調整を行う。

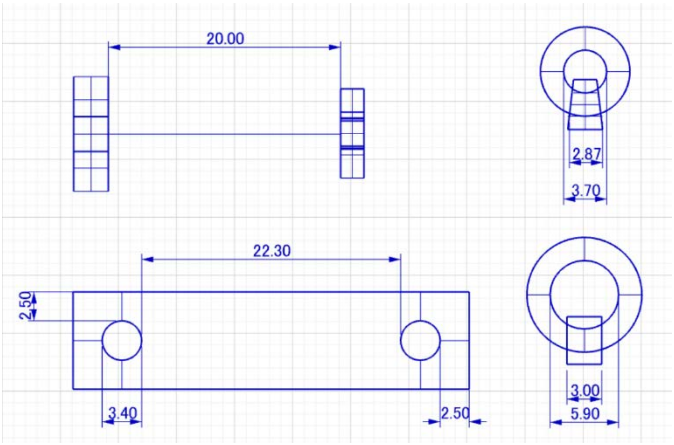

図 33　最適な寸法を模索中の CAD 画面。

次に、ある程度最適な寸法が定まったら、各々のパーツの配置を考える。今回はノズルの右斜め上より射角で撮影するため、パーツの配置は以下のようになる。こちらも、大まかな形が決まったら、サーフェスを結合し、実際に 3D プリントしてみる。

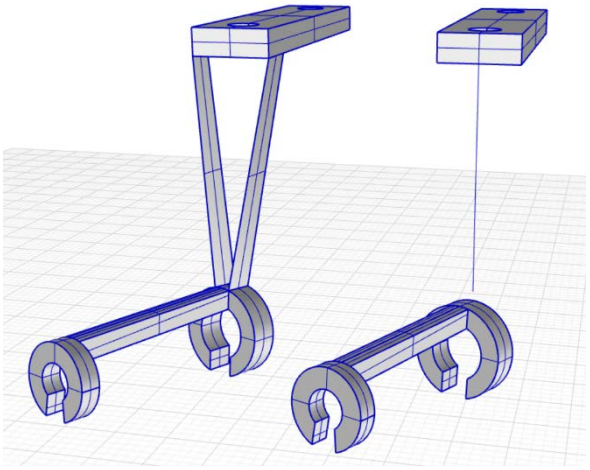

図 34　図 34 のパーツを配置している CAD 画面。

次に、配置し終わったパーツの強度に問題がないか、振動を増幅させないか（ヘッド付近に取り付けるため、その振動につられて揺れると映像が乱れる。）より最小限の要素でデザインを完

結させられないか、などを考慮しつつモデリングを進めていった。「図 35」の 3 つのモデルは、右から時系列になっている。一番右のモデルより、一番左のモデルの方が形状がよりシンプルになっていることが分かる。

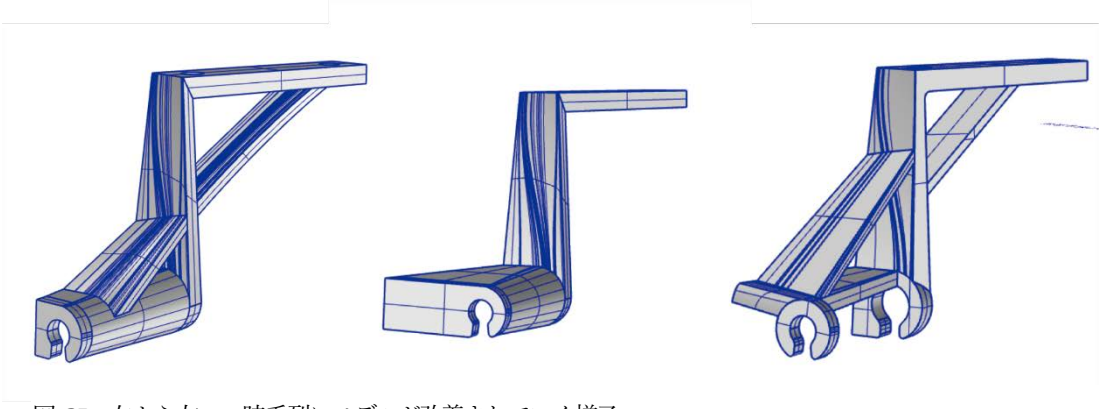

図 35　右から左へ、時系列にモデルが改善されていく様子。

最終的にデザインは一番左のモデルに落ち着き、それを実際に MF-2200 のヘッドに取り付けた様子が「図 36」である。また、この状態で撮影した映像のキャプチャーが「図 37」になる。

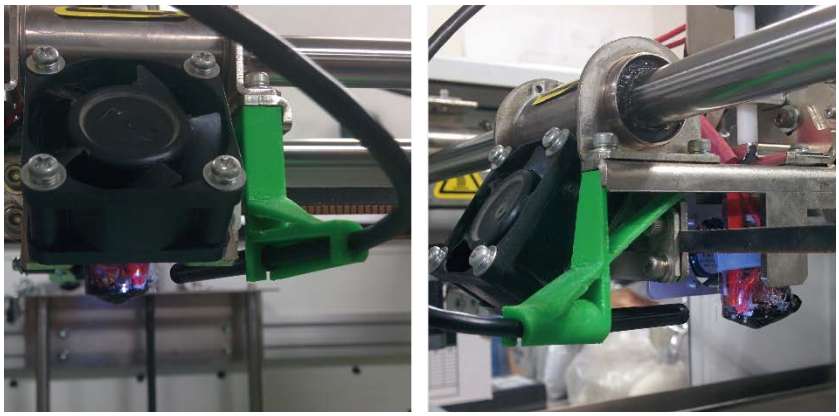

図 36　ヘッドに M-935 を取り付けた様子。

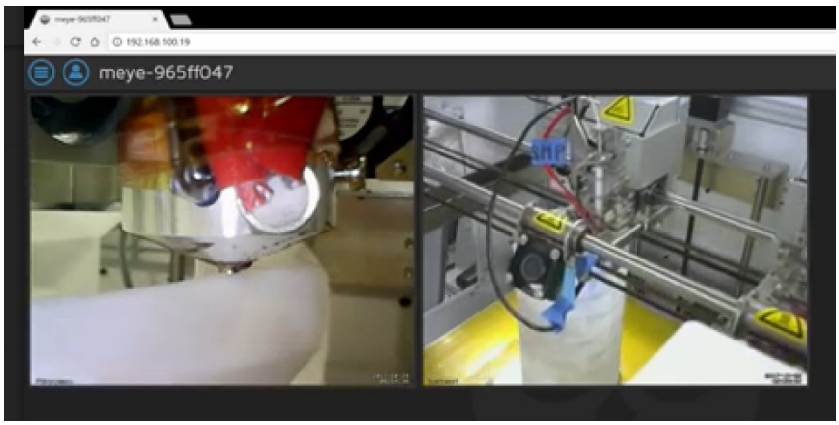

図 37　図 36 の状態で撮影した映像のキャプチャー。

ファイバースコープの接写の方は、時折、さらに寄りで撮影したい時と、引きで撮影したい時とがあることが分かった。そのため、ジグ側にズームイン/ズームアウトを行えるような機構を実装できるようにするとより良いと考えられる。

製作したジグは（https://www.thingiverse.com/thing:2756818）からダウンロードすることができる。

2.2.4 撮影時に使用したソフトウェアについて

カメラを接続するマイコンには Raspberry Pi 3 を使用した。また OS には motionEyeOS（https://github.com/ccrisan/motioneyeos/releases）を使用した。motionEyeOS はシングルボードコンピューターをビデオ監視システムに変える Linux ディストリビューションである。カスタマイズが非常に簡単で、カメラを何台でも接続できる他、録画した映像を Dropbox や Google Drive に自動アップロードしたり、ストリーミング URL の発行や、カメラ解像度の変更、動態検地をした場合のみ録画を開始するなどの機能を GUI 上から設定できる。

今回 Raspberry Pi に接続したカメラは全部で 2 台であったが、5V 2.5A の電源では落ちることがあり、Pi 3 負荷検証済みの電源セット（http://www.physical-computing.jp/product/1171）を使用することで電力不足を回避した。また遠隔から映像を監視するにあたっては、VPN で Raspberry Pi にアクセスすることで実現させた。

映像は（https://www.youtube.com/playlist?list=PLolc55LW0XS2INKy0JcVV84XWRsJQ5IRg）で見ることができる。

2.2.5 サーモカメラを用いた撮影

通常、3D プリントを行う際はヒートベッドの温度を上昇させるが、これは第一層のベッドへの吸着率を高める他に、既に造形された部分にも熱を伝わせることで、その上に射出されるフィラメントがよく接着することを助ける側面もある。故に、熱源が最下層だけにある熱溶解積層方式の多くの 3D プリンタでは、積層が高くなるにつれ、後者の恩恵は受けにくくなる。一方、既に積層された部分が熱も持ちすぎていてもいけない。「図 39-写真左」は、造形物の先端が形を成していないが、これは、積層が十分に冷える前（柔らかい内）に、次の層をのせた場合に発生する造形の失敗例である。前述したが、完全に冷却しきった上に次の層をのせても、十分に接着せず、最悪の場合「図 39-写真右」のように積層から亀裂が生じるケースに繋がる。つまり、冷えすぎず、温めすぎず、丁度良い温度で層を積み上げた時、層同士の接着力があり、形状も綺麗に維持された造形がなされるのである。

サーモカメラを併用し失敗の瞬間を撮影することはその基礎実験であり、どのような温度帯で印刷することが、失敗ないし最適化を検証する目的がある。

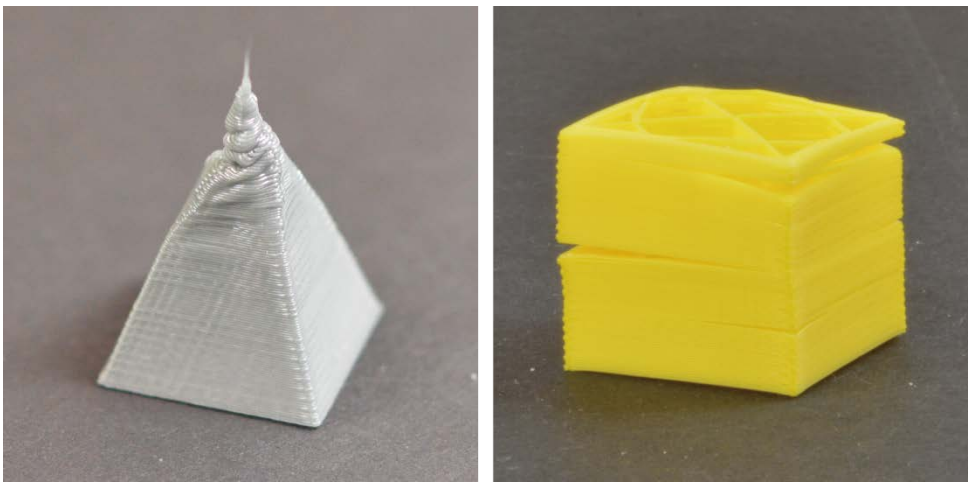

図 38（左）先端の方は、層が冷える前（柔らかい内）に次の層をのせた結果、形状が崩れている写真。（右）層が冷えすぎた所に次の層を乗せたため、層同士の接着力が弱く、いくつかの積層間から亀裂が生じた写真。（写真右の造形の失敗は、過度の冷却以外に、レイヤーハイトが大きすぎる時にも発生する）

図は、U5855A TrueIR 赤外線サーモグラフィ、350℃（https://www.keysight.com/ja/pd-2417816-pn-U5855A/trueir-thermal-imager-350c?cc=JP&lc=jpn)を用いて、印刷中の状態を観察したものである。ファイバースコープと違い、カメラ自体が大きく、ノズル付近に設置することができないため、定点撮影が望ましい。また、温度分布の解像度が出力物の構造までを分析するレベルに達しておらず、印刷に最適な温度帯を発見するには至らなかった

図 39　サーモカメラで造形中の様子を撮影した動画のキャプチャー。

3. 評価

3.1 ソフトウェア

3.1.1 対象ユーザーについて

今回製作したソフトを使っていただくために、素材メーカーで 3D プリンタ用のフィラメントを開発し、自身で出力の最適値を算出することもされている社会人Y さんと、同じ研究室内で、スライサーを開発している学生 T 君に協力をお願いした。前者からは、実用性のあるホストにするためのキメ細かい指摘が、後者からは、UI のユーザビリティと飛躍したアイディアが頂けると考えてのことである。

3.1.2 ユーザー実験

2 人にソフトウェアを GitHub からダウンロードいただき、それぞれから改善点や追加実装希望点をヒアリングした。Y さんの意見が「図 38」、T くんの意見が「図 39」である。

■インストールについて
問題なく動作させることができた。fabble の説明には COM ポートの変更の欄に追記で「ボーレート」も 3D プリンタや環境に合わせて変える必要がある点を記載した方がよい。

■使用感について
基本的に樹脂の最適出力条件を見つけ出す目線で使用。

・温度の項目の不在
ノズルの温度が一定だと、押し出し量（スピード×extrution）により、ノズルから出る樹脂の温度が変化するが、このバランスをとることが大事だと思うので、温度の項目は追加出来ればありがたい。

・extrution 変化によるバグ？
エクストルージョンの値を変更し、それがプリンタの動作に反映されるタイミングで、エクストルーダーのギアが変な挙動。

→値を低下させると「引き抜き」
→値を増加させると「ギア停止」

・デフォルトの増加量
Extrution は 10%となっていますが、通常の使い方（0.4mm ノズル、0.2mm ピッチ程度）だと、そんなに大きくは変化させないので 1%ずつの変化で良い。
当人は fabble の説明に従い、すぐに 1%に変更した。また、ボタン連打は厳しいので、直接数字を打ち込めるとありがたい。

・スピード変化に比例して押出量も増えてしまう。
→押出量は変えずにスピードだけ変化させたい場合もある。
→比例させるか、させないかチェックボックスによる選択肢があると嬉しい。

・反応時間
設定を変えてから、プリント結果に反映されるまでの時間が長い。
正直に言うと、このソフトウェアに一番期待していた部分である。通常のソフトウェア（Simplify3D や Repetier host）でも、造形中のパラメータ変更はできるが、パラメータ変更から実際の反応までの時間が長くて使いにくいので、改善を期待していたが結果は「同じくらい」という印象。

・G-code しか使えないことのデメリット
この「Rework」と通常のスライスソフトウェア（Simplify3D や Repetier host）は COM ポートを共有しており同時に使えないのは、結構不便だと感じた。
温度の変更や、その他のプリンタの状態を把握しながらの作業がやりにくいため、既に完璧に造形できることがわかっている G-code のみ、使用が許されているイメージ。

■その他
・出力後の overwrite ファイル
→なぜか出現せず。

・緊急停止ボタンの必要
安全のため押した瞬間に、その場で全てがストップする「緊急停止ボタン」は必須。画面上の「一時停止ボタン」も反応までに時間がかかるため、緊急停止には使用できない。

・レイヤーピッチの変更は面白そうなので、実装を期待している。
また、上記と同じ理由で「他のパラメータはいじらずに、レイヤーだけ変更したい」ので、比例した押し出し量の増加の有無は選べるようにできると便利。

■総評
使ってみた感じ、まだ既存のソフトウェア内の機能を凌駕しているわけではないので、それよりもスライスソフトと同時に接続できないことによるデメリットが目立つ形であった。

・パラメータ変更の反応速度の向上
・温度パラメータ項目の追加
・１つのパラメータだけ動かせるようにする（他を比例にて変動させない）

などで、実戦レベルになる。

図 40　Y さんからいただいたコメントの全文。

出力後に G Code を見直すことはしないので、ファイル生成の必要性を感じなかった。

BIQU（G-Code を解釈可能なデルタ式の 3D プリンタ）を使用する際、射出量とスピード、温度はプリンタ―側で変更できるため、ホストの優位性を確保するには不十分であると感じた。

テクスチャーを付与するのに、速さと射出量だけで満足な模様を生成できるのか？
→G Code の移動軌道自体を変更してあげることでより面白くなりそう。

ノズルがベッドにヒットしたり、逆に空間が空きすぎてフィラメントが定着しないことがあるため、全体的に Z の高さをオフセットする機能があると嬉しい。
（0.001 とか細かい単位で）

図 41　T さんからいただいたコメントの全文。

3.2 3D プリントの失敗の観察

3.2.1 観察結果とそのイラスト化

積層中に混じる、茶色い焦げの正体　白い造形物を印刷する時に、積層中で途中茶色くなっている箇所がある。これまではノズル中に溜まったゴミが排出されていたものだと考えていたが、接写で確認してみると、他にも理由がるように見えた。ノズルが造形物の端で折り返す際、よこは熱が溜まりやすく、他の箇所と比較しても既に造形されている部分が余計に溶けてしまう。そうして余分に溶けだして下の積層が、吐出中のフィラメントと混じった結果、体積が増え、射出口付近の焦げと融合し、積層されるのである。「図 42」ではそのフローをイラスト化した。

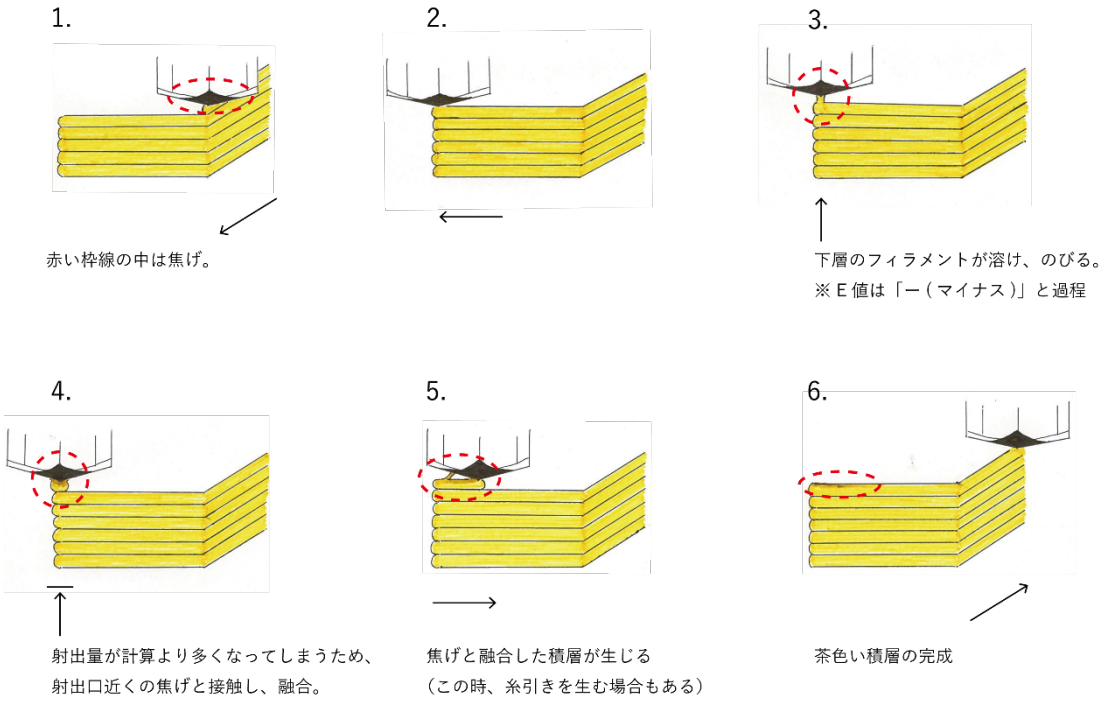

図 42　積層中に茶色い痕が残るまでのフローをイラスト化したもの。

造形物の端や、中空部分の印刷時におこる造形不良　サポートなしで中空状態の層を印刷すると造形不良がおきる。「図 43」はそのフローをイラスト化したものである。図中の[2]にあるように、ノズルの熱により下層が温められ柔らかくなると、射出されるフィラメントの勢いに柔らかくなった中空状態の層が耐えられず、変形してしまう（=[3]）。また、[6]にあるように、下層との接面が少なくなる造形物の端の部分は、ノズルの動きに端が引っ張られ、反りを引き起こすことになる。実際、スライサーによりサポート不要と判断された箇所であっても、印刷環境によっては以下のような造形不良を起こすケースがある。

このような場合は、ファンの回転数を上げ、下層の冷却効果を高めるか、ノズル温度を失敗時よりも下げることで改善されることもある。

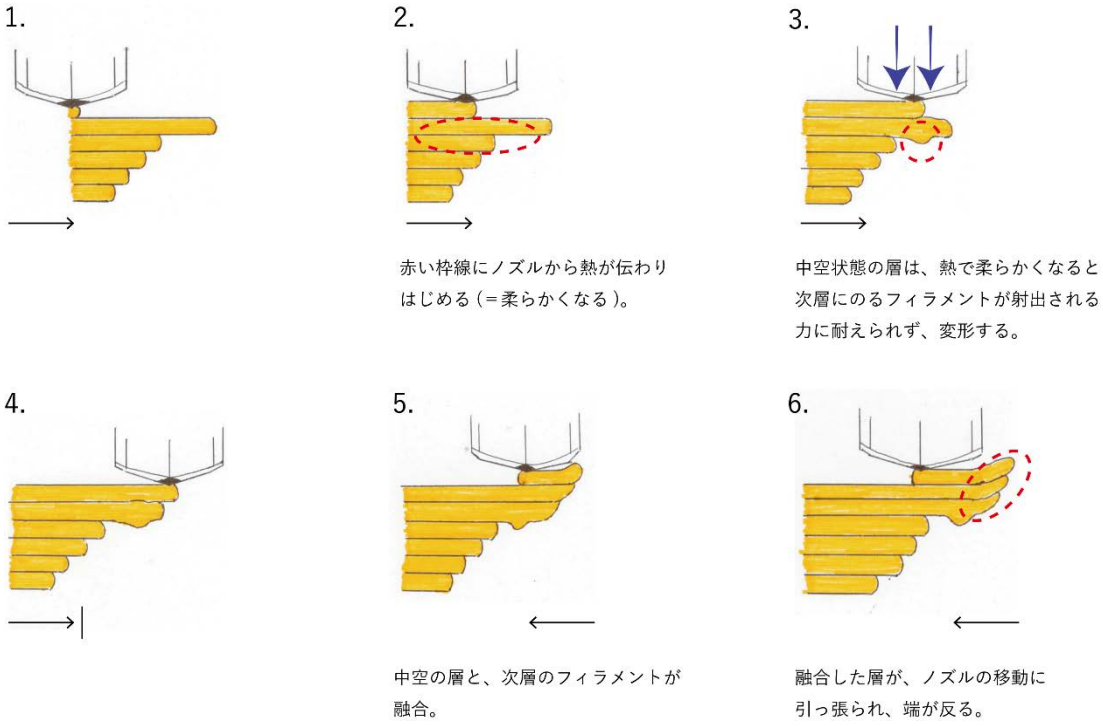

図 43　造形物の端や、中空部分の印刷時におこる造形不良のフローをイラスト化したもの。

4. 考察

4.1 結論

はじめに、ソフトウェアに関して述べる。当初多様なマテリアルを扱う人たち向への、キャリブレーションソフトとして開発をはじめたのだが、途中、G-Code の周期的制御により凹凸の付与を機能として実装したことにより、デザイン用途で使用する可能性が一時的に高まった。故にソフトウェアの評価も、キャリブレーションソフトとデザイン用途との 2 つに分けて述べたい。キャリブレーションソフトの機能としては、F・E を即時的に変更するという、現行ソフトウェアに搭載されている機能から実装しはじめた点は、同様の方法で、Z やその他あらゆるパラメータを変更するための技術的基礎になったので良かったと言える。加えて、出力後の G-Code の保存や、Z の値の即時的変更など、既存のソフトウェアにはない機能の実装も実現できた。ただし、ユーザー実験の結果からも分かるように、それが有用性を担保するためのクリティカルな点になりえてはおらず、結果的に現在 3D プリントユーザーの開発環境を快適にすることには十分に貢献できているとは言えない。原因は、開発当初、ユーザー調査を十分に行わず、場当たり的な実装をしていたためだと考えられる。最終的にソフトウェアが完成してからユーザー実験を行ったため、改善案や新規で実装を希望するような要望の回収がかなり後半になり、時間的に実装までこぎつけることができなかった。ソフトが自分の中で未完成な段階でユーザー実験をすることに躊躇があったのだが、もう少し早く、ユーザーとの接点や要望の洗い出しを行っていれば、今のものよりも有用性のあるソフトウェアを製作できたのではないかと考えた。次にデザイン用途としては、G-Code を即時的かつ周期的に変更しながら出力を行う事例がなかったため、試み自体は新規性があったと言える。ただ、結果的にできた製作物をみると、CAD 上や G-Code を直接編集して製作された、テクスチャーや凹凸のある造形物との差異を十分に生み出せてはいない。即時的に変更できることの必然性を考えるなら、Solid Vibrations のような、環境が造形物に対して変化をもたらすインタラクションの方向に振っていればアート性生まれたのではないかと考えた。

次に、カメラをノズルに付近に取り付け、接写にて観察を行ったことに関して述べる。まず、撮影されたノズル付近の映像には、前例がなく、それでしか観察できない瞬間を捉えられる可能性を提示はできたであろう。また、失敗の原因のうち、幾つかをより明確にすることができた上、造形ミスが起きるまでのフローを、映像媒体で提出できない論文において、イラスト化で表したのも、結果的に読者に直感的その状況を理解して貰うことに繋がったと考える。

4.2 展望

命令通りに動くだけの 3D プリンタの限界がきていることは、1 章の中でも話したが、ソフトウェアを作っている時に、GCode の仕様上、どうしても個別に変更することができない箇所などが存在した。元々CNC 用の言語であった G Code は、そのシンプルな記述方法が良点の一つであり、筆者のような学生でも、ハックすることができた。その点は素晴らしいといえるが、3D プリンタの領域が拡張子、性能が向上していく時に、果たして G Code は現在の形のままで良いのだろかと疑問に感じた。プログラム言語はその時々の状況や、時代に合わせ新しく生まれ変わるが、GCode もそのタイミングに差し掛かってきているのではないかと考えられた。

例えば、カメラにて層の幅を測定しながら、E 値をダイナミックに変更しつつ印刷をしていく方法や、周囲の温度を計測しながら、ノズル温度を微妙に変化させつつ印刷をするという方法である。

また、ユーザー実験の結果からある、G-Code を用いる以上発生する反応時間の遅れは、G-Code という高レイヤーではなく、ファームウェアレベルの低レイヤーレベルで修正をかけていく必要性を説いていた。

最期に、今回製作したソフトウェアは、既存のソフトウェア（Simplify3D や Repetier host）などと COM ポートを共有しており、同時に使えない点に指摘があったが、これに関しては、arduino 側で動くファームウェアにすることで解決できると考えられた。つまり、スライサーから G-Code が本ソフトを実装した arduino をバイパスし、最終的にコントロール基盤側の arduino に送るということである（arduino のシリアルを 2 つ使う）。この際、パラメータは即時的に変更するために、arduino 上で起動するウェブサーバなどで変更できるようにすれば良い。

5. おわりに

ソフトウェアの路線をキャリブレーション用途と、デザイン用途の 2 本建てにしたことや、複数種類でのカメラでの観察など、実験の内容が多岐に渡り、作るものや検証の内容が毎週違う経験を通し、筆者自身は技術的にも様々なことを学ぶことができた。一方、能力や時間が、分散してしまったことも確かで、どれか 1 つに絞って研究していたら、本論文の趣旨でもある、3D プリントユーザーの環境を快適にすることにも繋げられたのではないかと考えた。

場当たり的実装で様々なテーマに手を出せば、最初は同時並行でスピード感を落とさず進捗を出せるが、ある一定のライン（それだけに没頭しないと見えてこない、分析の結果や考察）に達すると、進捗が滞ることになり、どこかで、ウォーターホール的な開発スタイルを組込む必要があるように感じた。

6. 謝辞

まず、本研究を進めていくにあたり、ソフトウェアの実用的な側面とデザイン用途としての可能性を、一緒になり丹念に探求して下さった田中浩也教授に心より感謝致します。カメラでの失敗の観察はじめ、自分一人では息が詰まりそうなことばかりでしたが、先生とコミュニケーションをしていく内に、私から見ると無機質なものが、有機的な豊かさを帯びていく瞬間を幾度となく体験できたことは、一生の財産です。

また、工学部の存在しない本大学において、工学的な論文の書き方や、研究とは何かについて、説いて下さった青木翔平特任教授にも感謝の意を述べさせていただきます。失敗した結果も、書き方次第では成功と等価の価値をもつ可能性を感じられたのは、ポジティブに研究に向き合う一番のきっかけとなりました。

加えて、ユーザー実験や、添削などもご協力いただきました、田中浩也研究室の先輩・後輩・同期、そしてメーカーの皆様に心から感謝致します。

最後に、高校卒業から 6 年間に及ぶ大学生活を経済・精神的に支えてくれた父、祖父、弟、故母、故祖母に、感謝と敬意を表し、謝辞とさせていただきます。

7. 参考文献・引用

[図 4] 鋼材変更による金型の物理強度アップ・修理コスト削減

http://www.shashutsuseikeikanagata.com/index.php/author/shashutu/ (2018.1.14 閲覧)

[図 6] JSR、3D プリンター用フィラメント「FABRIAL（ファブリアル）R シリーズ」を発表-

http://www.jsr.co.jp/news/0000613.shtml (2018.1.14 閲覧)

[図 6] 業界初!造形後に形状カスタマイズ可能な 3D プリンター用「形状記憶ポリマー」発売

https://www.atpress.ne.jp/news/106523 (2018.1.14 閲覧)

[図 11] Simplify3D 最新版!FDM で光造形並みの厚さと、部分部分で異なる設定が可能に?!

http://makerslove.com/16793.html (2018.1.14 閲覧)

[図 12] MAKERBOT MINFILL 造形スピード 30%アップ、フィラメント 30%節約

http://pages.makerbot.com/jp-minfill.html (2018.1.14 閲覧)

[図 14] Honeywell ダイナミックシュミレーション 簡単なサンプルファイル (17)

http://hps-honeywell.co.jp/unisimdesign/usd109.htm (2018.1.14 閲覧)

[図 15] Algorithm allows 3-D printers to “read ahead” of their programming to boost speeds.

http://www.sciencedirect.com/science/article/pii/S0957415817301277 (2018.1.14 閲覧)

[図 16 , 38] SIMPLIFY3D Print Quality Troubleshooting Guide

https://www.simplify3d.com/support/print-quality-troubleshooting/ (2018.1.14 閲覧)

[図 26] Expressive Fused Deposition Modeling (CHI2017 Paper)

https://dl.acm.org/citation.cfm?doid=3025453.3025933 (2018.1.14 閲覧)

[図 27] Solid Vibrations 2015

http://oliviervanherpt.com/solid-vibrations/ (2018.1.14 閲覧)

[図 28] Velocity Painting

https://www.repetier.com/velocity-painting/ (2018.1.14 閲覧)

[図 32] M-935

https://www.amazon.co.jp/スパイダーズ X-ファイバースコープ-小型カメラ-スパイカメラ-M-935/dp/B071DNYRM4 (2018.1.14 閲覧)

[図 32] c270

https://www.logicool.co.jp/ja-jp/product/hd-webcam-c270 (2018.1.14 閲覧)