# A Roadmap Towards Parallel Printing for Desktop 3D Printers

Molly Aubrey Carton**, Chandrakana Nandi*, Adam Anderson*, Haisen Zhao*, Eva Darulova***, Dan Grossman*, Jeffrey Ian Lipton**, Adriana Schulz*, Zachary Tatlock*

* Paul G. Allen School of Computer Science & Engineering, University of Washington, USA
** Mechanical Engineering Department, University of Washington, USA
*** MPI-SWS, Germany

## Abstract

3D printers with multiple extruders (or multi-headed printers) are common in the desktop fabrication community, but are primarily used for multi-color or multi-material printing, using only one extruder at a time. What if these multi-headed desktop printers could also be used for simultaneous parallel printing? While this is a relatively unexplored direction, we argue that it deserves further investigation: a flexible, robust, and affordable parallel printing ecosystem could significantly reduce fabrication time for many applications and further enhance the value of desktop rapid prototyping.

We propose a research agenda to explore the development of a parallel printing pipeline, and summarize our observations from a preliminary investigation of simultaneous extrusion. We hope this vision will encourage and guide future research in developing hardware, firmware, and slicers to facilitate parallel 3D printing.

## 1 Introduction

Desktop-class 3D printers are democratizing manufacturing. They are increasingly used for rapid prototyping, customized manufacturing, casual making, medicine, and education. While there are numerous styles of affordable printer that are now available, Fused Deposition Modeling (FDM) printers are perhaps most popular and commonly used. Even though the primary use of FDM printers is *rapid* prototyping, in practice, unfortunately, they are often much *slower* than one would expect — it can take tens of hours to multiple days to print large models. One way to expedite the printing process would be to add more extruders that can simultaneously print parts of a large model. In fact, several desktop 3D printers are already equipped with multiple extruders [7, 34, 9, 3, 26] but their pipelines (CAD tools, slicers, firmware) support only multi-material or multi-color printing. Little prior work on desktop printers has demonstrated the use of multiple extruders to simultaneously print the *same* object.

This paper motivates further exploration of parallel 3D printing opportunities on desktop-class machines. We present a research vision for parallel-printing-capable fabrication pipelines by first identifying the challenges associated with parallelizing each stage of the pipeline and proposing strategies to address them, and then discussing an early proof-of-concept we developed that suggests the promise of parallel 3D printing. We view parallelized desktop 3D printing as analogous to parallel computing — the latter revolutionized computer science and led to new algorithms, parallelizing compilers, and numerous advances in high-performance computing. More importantly, even though not all algorithms can be parallelized, the benefits of parallel computing are nevertheless indisputable. Similarly, we argue that even though not all designs can be fabricated in

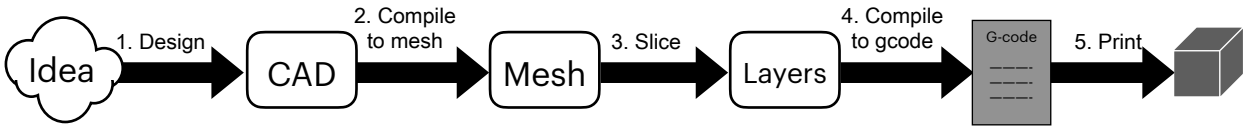


Figure 1: *A typical 3D Printing Workflow. Once a model is designed using CAD tools, it is compiled down to a polygon mesh. The mesh is then sliced to generate 2D layers. The layers are then compiled to G-code. Ultimately, the G-code is sent to a printer which has firmware that can interpret the G-code to print the model.*

parallel, parallel printing still has the potential to significantly reduce print time for a broad class of designs thereby making desktop 3D printing an even more appealing fabrication option for rapid-prototyping. We hope this paper will encourage researchers to pursue this direction further.

The rest of the paper is organized as follows. Section 2 first provides necessary background on how typical desktop 3D printing pipelines work. Section 3 then lays out a bottom-up research vision that highlights the challenges for enabling parallel 3D printing at each stage of the pipeline. Section 4 proposes strategies to mitigate the bootstrapping problem due to the current lack of both suitable hardware and software. Section 5 discusses preliminary experiments we conducted that indicate the promise of simultaneous extrusion for parallel printing, Section 6 discussed relevant related work, and Section 7 concludes our discussion.

## 2 Background

This section provides a high-level overview of a typical desktop 3D printing process (shown in Figure 1). A designer first develops a CAD model for a design [30, 18, 25, 24], then compiles it down to a polygon mesh. Alternatively, it is also common to download polygon meshes directly from online repositories [31]. The mesh is then sliced into horizontal layers / slices (each layer is a set of 2D polygons) using a slicer [1, 29]. Each slice is then compiled to G-code (perimeter and infill of each 2D polygon in each layer) which is sent to the printer. The firmware [22, 13] in the printer interprets the G-code sequentially to actuate motors and heaters for printing the object.

We argue that researchers must rethink (and potentially redesign) *each* phase of this traditional pipeline to enable parallel fabrication in desktop devices, akin to recent developments in industrial scale machines [5]. The availability of modular [28] and easily customizable [35] machines together with open-source firmware packages, slicers, and CAD tools can help realize this vision as demonstrated by our own early exploration (Section 5) — we show that it is possible to parallel print basic designs by simple modifications to various stages of the traditional desktop fabrication pipeline shown in Figure 1. We envision a future where parallel fabrication machines will be just as common as their sequential counterparts and will be supported by domain specific languages [33, 32] and frameworks [8] for representing and building next-generation fabrication machines that have recently been proposed.

# 3 Vision: Parallel Printing Capable Pipelines

Given $N$ extruders, successfully incorporating parallelism in the desktop 3D printing pipeline can ideally offer an $N$ times speedup in printing which can dramatically improve the utility of desktop manufacturing for rapid prototyping. This section proposes a bottom-up approach for solving the challenges involved in each phase of the pipeline for enabling parallel fabrication.

- **Machine design.** Printer configurations vary in terms of factors like number of extruders, the dimensions of the print bed, the distance between the extruders, and the degrees of freedom among extruders. Each of these factors adds constraints that the printer must always satisfy during printing. These constraints also affect the parallel G-code generation — the relative arrangement of the extruders impacts the available parallelism because some movements may be prohibited by construction, and some for safety and correct printing. Below we classify FDM printers in terms of the number of extruders and their allowed movements:

  - $N$ extruders in a fixed arrangement. All extruders move simultaneously "in lockstep" and extrusion is possible at any time by one or more. The most common variant of this is the *fixed-width dual extrusion printer*, which has several commercial variants available.
  - $N$ extruders with a single independent axis which can be the x, y, or z axis. The most common variants of this is a dual extrusion printer with independent x.
  - $N$ extruders with two independent axes, which can be with (x, y), (y, z), or (x, z).
  - $N$ extruders with three independent axes. Such printers could be implemented using multi-axis robotic arms.

  Several optimizations that are valid in the single extruder scenario (e.g., moving at a 45 degree angle with respect to both $X$ and $Y$ axes to maximize velocity) must be re-evaluated and potentially modified to be applicable for multiple extruder machines. The machines may be designed to be modular and reconfigurable depending on the part being printed — for example changing the angle or distance between the extruders if it provides more parallelism. Additionally, currently available multi-head machines [7, 9, 3] are much more expensive compared to their single-head counterparts [2], making prototyping and experimentation difficult. More affordable machines [35] can be used as alternatives as we show in Section 5.1.

- **Firmware support.** Currently, most 3D printer firmware packages are designed to support printing by a single extruder. For printers with multiple extruders [34, 7], the firmware can support multi-material or multi-color printing but it only allows extrusion by a single extruder *at a time*. The closest to parallel printing that common multi-head printers can currently accomplish is "ditto printing" [21], where two identical objects are simultaneously printed.

  Without adequate firmware extensions, parallel printing cannot be accomplished, even if appropriate hardware is available. Designing firmware packages for parallel printing is challenging — it involves solving both geometric and physics constraints that become more complex as the number of extruders increases. A firmware package for parallel printing must additionally also check for safety constraints such as collision avoidance; these are typically

not checked by current firmware packages since a single extruder cannot collide with itself (printers do have limit switches to prevent the extruder from "going off the rails"). Further, for re-configurable machines, the firmware will also require reconfigurability so that it is still compatible with the hardware after adjustment. With recent advances in programming language design for machines [33, 32], optimizing compilers for fabrication [36], and program synthesis directed towards fabrication [17, 16, 6], we hope that in the future, critical components of firmware code can be automatically synthesized given a specification of a machine and its associated constraints. In the meantime, we have found that preliminary forms of parallelism can already be accomplished by simple modifications to open-source firmware [13] (Section 5.2).

- **Slicer extensions.** Currently, slicers can already be configured for the type of printer being used (e.g., delta vs. Cartesian configurations) [1]. To support parallelism, slicers must extend parameterizability to account for number of extruders, their degrees of freedom, and how they move with respect to each other. To achieve parallelism, slicers must solve complex constraint satisfaction problems similar to path planning in robotics. Unlike sequential printing, the slicer must ensure that the generated G-code does not lead to collision between extruders, or overlapping printing, i.e., one extruder extruding over material extruded by another extruder on the same layer. A single sweep from one corner of a slice to the other, which tends to be correct (even if not necessarily the most efficient) for sequential printing, is no longer guaranteed to work for parallel printing — the slicer may be required to move the extruders around, turn on/off one or more extruders, and even adopt new path planning algorithms to achieve parallelism.

  Achieving maximum parallelization also constrains layer and infill paths. Where creating a solid layer with a single toolpath may be most efficient with a space-filling path such as a Hilbert curve or double spiral (`https://www.scientific.net/AMM.190-191.790`), these paths are not necessarily the most efficient use of multiple extruders, because achieving maximum "extruder uptime," with all extruders actively printing, places requirements on the spatial repetition of toolpath shapes.

  The slicing process can be broken down into two phases: (1) generating 2D slices from the 3D mesh of a model, (2) generating parallel G-code for each 2D slice. Within each layer, there are again roughly two different types of extrusion paths (1) perimeters, and (2) infill. Since infill takes the majority of time in most models, we propose that infill G-code should be the first target for parallelism. Perimeter G-code can also be parallelized but will likely require more complex algorithms than infill. Slicers for parallel G-code must also support re-orienting the part on the print bed to maximize the simultaneously extrusion time.

- **CAD analysis.** While early prototypes for parallel printing should focus on firmware and slicers, CAD tools may also be able to assist a designer in making their design more suitable for parallel fabrication. Similar to parallel computing where not all programs are suitable for parallelization, not all designs may be printable in parallel. A CAD level analysis can identify parts of a design that are parallelizable or recommend modifications that may make the design easier to print in parallel without sacrificing user intent or the stability of the printed model. Such a design analysis tool will ultimately solve a multi-objective optimization problem where the objectives include fabrication time, stability of the design, proximity of the

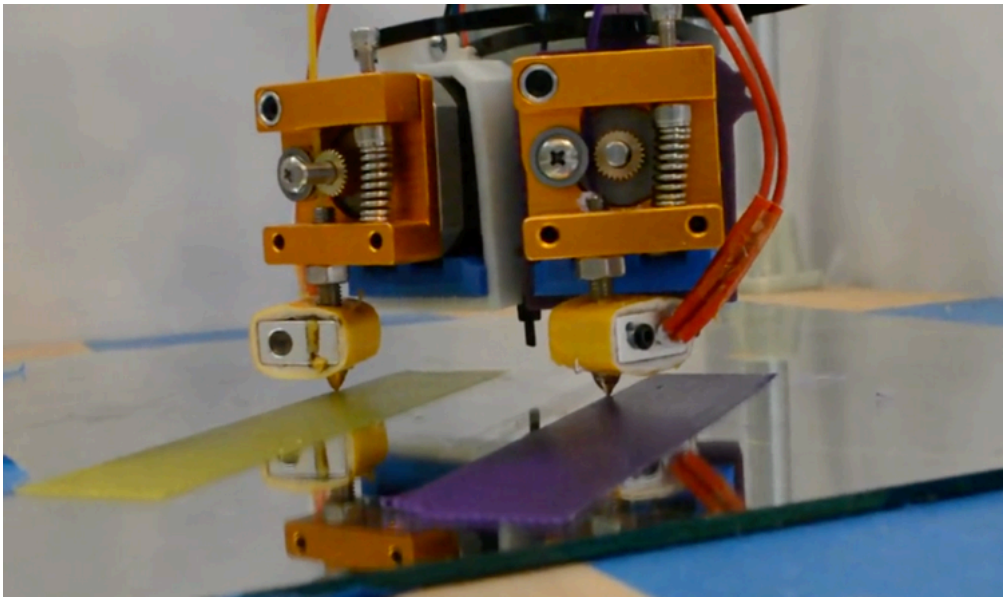

Figure 2: *Extruding from both nozzles to create a maximally parallel diamond shape. The model is oriented in a way that allows both extruders to simultaneously extrude half of the design.*

final result to original designer intent, etc.

As an simple example of parallel printing, consider the large diamond-shaped model in Figure 2 and Figure 3 — this shape is oriented so that maximum parallelism can be obtained by allowing both extruders to extrude simultaneously and each print half of the shape. Indeed, a preliminary estimation of the print time using an online G-code simulator [20] indicates that it would take approx. 17 minutes to print this model with a single extruder whereas with two extruders it reduces to approx. 8.5 minutes at the same print speed.

## 4 A Research Plan

For an end-to-end parallel fabrication system to be successful, the strategies in Section 3 cannot be developed in isolation because the phases in the fabrication pipeline are interdependent and integrating the individual components to build the entire pipeline comes with its own challenges such as compatibility, reproducibility, and inter-operability. These challenges also lead to a bootstrapping problem for facilitating parallel 3D printing research — without adequate hardware parallel printing cannot be accomplished, and without software and firmware support the necessary code for running on the hardware cannot be obtained. We propose the following three criteria that future work in parallel printing should consider.

- Reproducibility. Readily available (e.g., open source) and customizable solutions help with reproducing results. Open-source firmware [13] and slicers [27], and customizable and affordable hardware [35] are examples of tools that may be suitable for early investigations.
- Simulators. Even before hardware is available, building accurate simulators can guide the design of each phase of the pipeline thereby resolving the "chicken and egg" bootstrapping problem. Simulators also make debugging and prototyping easier.

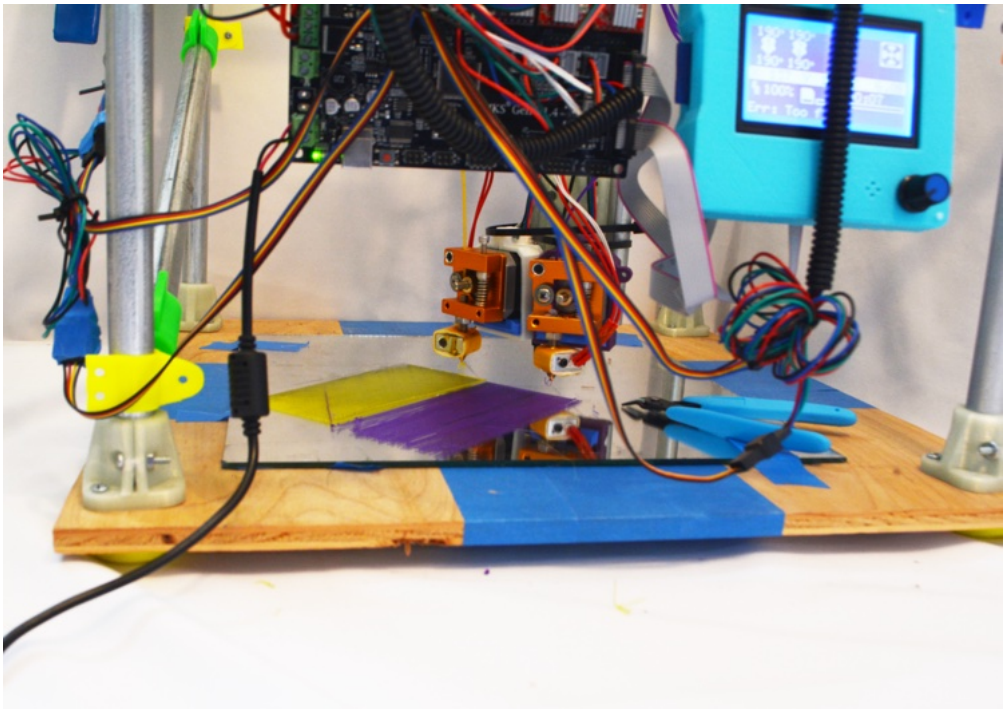

Figure 3: *Full layer of the maximally parallel-printed simple diamond shape. With both nozzles printing for the entire shape, we can achieve maximum speedup.*

- Inter-operability. Common interfaces between different phases of the fabrication pipeline should be established to enable inter-operability. For example, both the firmware packages and slicers should use the same language for geometric (and physical) constraint formalization, and G-code should have unified semantics [23] that corresponds to what firmware packages support.

## 5 Preliminary Experiments

Guided by the strategies from the previous sections, this section describes simple modifications to the fabrication pipeline (Figure 1) that we implemented in order to fabricate example test parts in parallel. While we did not explore how changes to the rest of the pipeline (e.g., CAD analysis) can facilitate parallel 3D printing, we hope these early experiments will provide a starting point for future research to build on.

### 5.1 Hardware

As discussed in Section 3, easily customizable / affordable devices like MPCNC [35] help facilitate research and early prototyping because these open source systems are relatively easy to re-design and update as required. Each of the printer configurations in Section 3, for example, can be designed using MPCNC or other open hardware. To demonstrate, we built an early prototype of such an MPCNC-based 3D printer and modified it to support parallel 3D printing. Figure 4 shows the setup — most of the hardware modification is in the extruder mount, whose original design holds

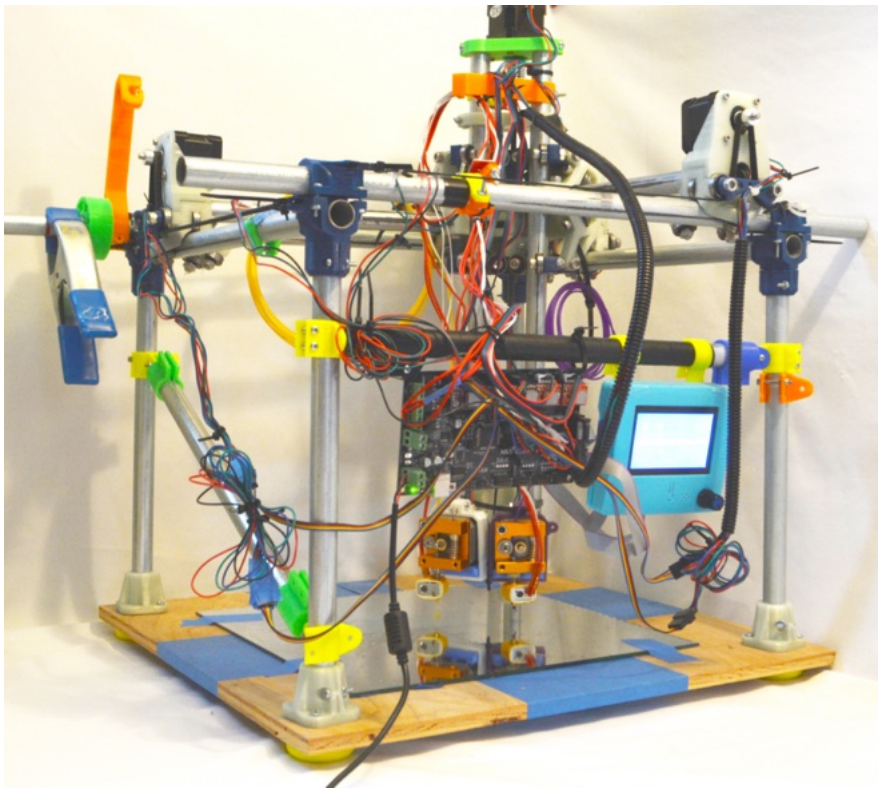

Figure 4: *MPCNC printer with modifications (dual extruder mount) to support basic dual extrusion.*

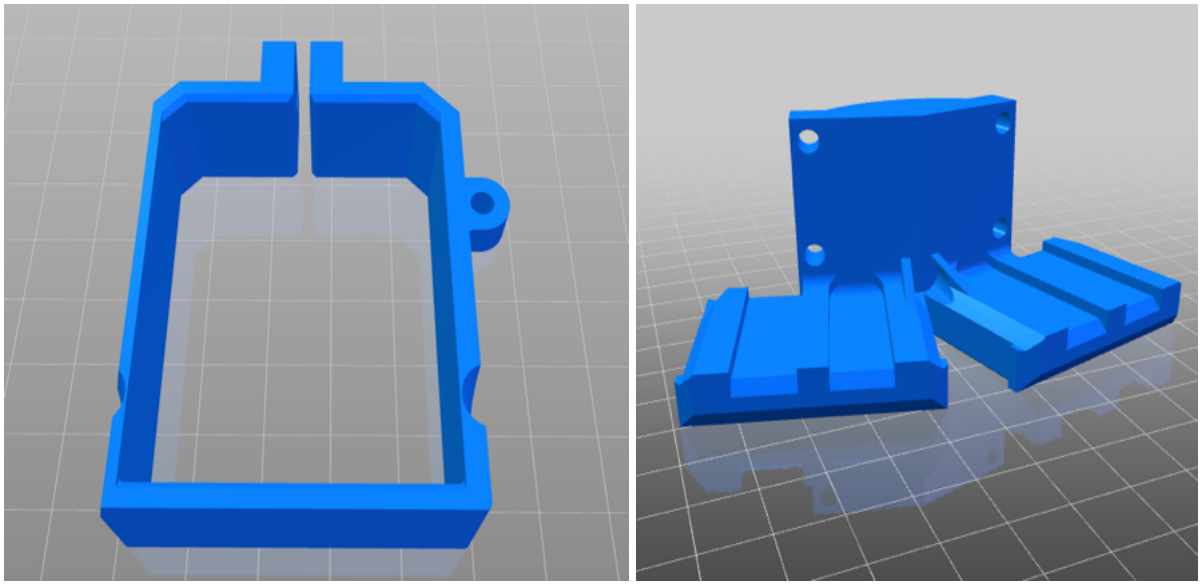

Figure 5: *Our design for a dual MK extruder mount for 1/2 EMT-based scaled MPCNC. The designs for both these parts are publicly available on Thingiverse [4].*

```
; Part 1
T1 ; switch to extruder T1
G1 X40.00 Y0.80 Z0.00 E2.40 ; extrude with T1 up to (40.00, 0.80, 0.00)
; Part 2
M605 S2 ; turn on dual extrusion mode
G1 X80.000 Y0.80 Z0.00 E2.40 ; extrude with both up to (80.00, 0.80, 0.00)
M605 S2 ; turn off dual extrusion mode
; Part 3
T0 ; switch to extruder T0
G1 X120.00 Y0.80 Z0.00 E2.40 ; extrude with T0 up to (120.00, 0.80, 0.00)
```

Figure 6: *Snippet of parallel G-code that the firmware changes in Section 5.2 can run on the printer described in Section 5.1 (retraction lines removed for clarity).* `T0` *and* `T1` *are the two extruders. The semantics of* `G1` *is the same as described in the Marlin documentation [11]. A* `T0` *indicates that the following* `G1` *command will be executed only by extruder* `T0`*, and same holds for* `T1`*. Each G-code command wrapped by an* `M605 S2` *is executed by both extruders simultaneously. To disable extrusion from both extruders, we simply set no extrusion value for the active E axis.*

a single extruder.

We started our setup with a scaled version of the original MPCNC designed to use one inch electrical conduit, producing a maximum build space of about 200 mm in each axis. We designed and 3D printed a holder that can carry a second extruder and attached it to the gantry. These models are shown in Figure 5 and our designs, as well as the scaled MPCNC parts, are available on Thingiverse [4]. This printer setup is equivalent to a fixed-width dual extrusion model (Section 3). The initial positions of the two extruders are $T_0$: (0, 0, 0) and $T_1$: (49.5, -49.5, 0), i.e., the second extruder is at a 135 degree angle with the x-axis, with a diagonal spacing of about 70 mm between the extruder heads. The amount of parallelism offered is limited — both extruders must move together as they are connected. Future research should explore recent 3D printer models [28] that are modular, increasing the freedom for experimentation with not only two fixed width extruders, but additional extruders and more degrees of freedom.

### 5.2 Firmware

While some firmware packages provide basic support for ditto printing, most do not implicitly support the simultaneous extrusion required for parallelizing printing. Our early experiments have shown that adaptable, open-source firmware packages like Marlin [13] can be modified to switch between "single" and "simultaneous" extrusion modes. This section describes how we added support for simultaneous extrusion in Marlin for the dual extruder MPCNC 3D printer (Section 5.1) we built. Marlin has a special instruction (disabled by default) `M605` [12] which can be used to (1) move two x-carriages either completely independently (using the `S0` argument), (2) move a single carriage while the other is parked (using the `S1` argument), or (3) move two carriages in unison (using the `S2` argument). We found that using the command `M605 S2` allows us to extrude through both extruders (`T0`, `T1`) in the following four ways:

1. only `T0`

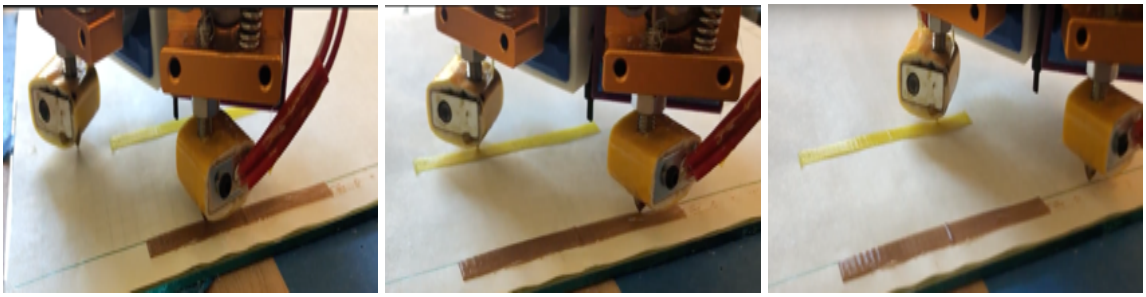

Figure 7: *Printing in three modes: `T1`, `M605 S2`, and `T0`, corresponding to Part1, Part 2, and Part 3 in Figure 6 respectively. `T0` is the extruder above and `T1` is the extruder below.*

2. only `T1`
3. both `T0` and `T1`
4. neither `T0` or `T1`

This mode is called the "dual nozzle duplication mode" [12]. All four above settings are essential — the printer should be able to use both extruders during simultaneous printing (i.e., both are inside the shape's perimeter), or turn one or both off if they are outside the shape. We activated this feature of Marlin with less than 20 lines of change [1]. We made two significant changes:

- By default, Marlin disables the stepper motor of the inactive extruder [10]. We set the field `DISABLE_INACTIVE_EXTRUDER` to `false`, so that none of the extruders are disabled, even when inactive.
- We modified the implementation of `M605` to make it a toggle, i.e., an `M605` followed by another `M605` disables extruder duplication whereas a single `M605` enables it.

We then flashed the printer with our updated Marlin firmware. While this simple modification allows us to use our fixed-width dual extruder MPCNC for simultaneous extrusion, we emphasize that for more complex models / printer configurations (e.g., with independent axes of motion), additional changes will be required. Figure 6 shows a small example of parallel G-code that demonstrates how simultaneous extrusion works and Figure 7 shows our MPCNC-based parallel 3D printer extruding those G-code commands.

### 5.3 G-code Generator

We built an early prototype of a code generator for parallel G-code that can be run on our MPCNC (Section 5.1) using the modified firmware (Section 5.2). As described in Section 3, we broke down the code-generation into two steps, (1) slicing, and (2) generating G-code for each slice. For (1), we used an off-the-shelf geometry processing library [19], and implemented a parallel G-code generator for (2). The latter works by computing the intersection of (1) rays starting at each extruder and (2) the perimeter of the layer. The angle of extrusion, $\theta$ the rays make with the extruder axis can be set by the user. The extruder axis in this case is the line joining the two fixed-width extruders

[1] Our changes are publicly available at `https://github.com/chandrakananandi/gayatri-marlin`.

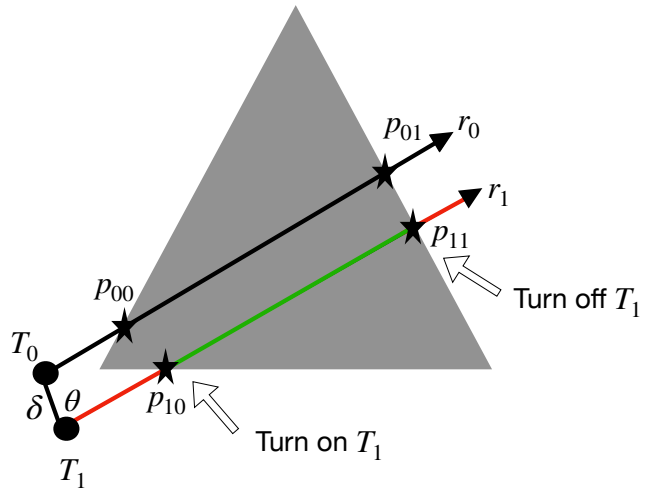


Figure 8: *`T0` and `T1` are two extruders and $\delta$ is the distance between them (i.e., length of the extruder axis). $\theta$ is the angle between the infill and the extruder axis. $r_0$ and $r_1$ are two rays starting from the their initial positions respectively. The points of intersection of the rays and the perimeter of the shape (a triangle) indicate the positions where the extruders should be started/stopped. For instance, the red parts of $r_1$ show no extrusion by `T1` and the green part shows that `T1` is extruding inside the shape.*

(`T0`, `T1`). Let the distance between them be $\delta$. Below we explain how the code generator works for simple shapes and Figure 8 shows an illustration.

The $(x, y)$ coordinates where each extruder will be turned on/off is determined by observing the parity of the points of intersection of the rays and the perimeter. This assumes that initially both extruders are turned off. We consider `T0` to be the primary extruder and `T1` to be the secondary. First, the algorithm casts two parallel rays, $r_0$ and $r_1$ from the origin points of `T0` and `T1` respectively, and computes all the points of intersections with the perimeter of the slice. Let $\{p_{00}, p_{01}, p_{02}, ..., p_{0n}\}$ be the points of intersection of $r_0$ with the perimeter of the slice. Similarly, let $\{p_{10}, p_{11}, ..., p_{1n}\}$ be the points of intersection of $r_1$ with the perimeter of the slice. We compute the starting/stopping points of extrusion for both extruders in terms of the primary extruder by projecting the points of intersection of $r_1$ on $r_0$. The points of intersection for both rays will be interleaved in most cases. Let us consider an interleaving like so: $\{p_{00}, p_{10}, p_{11}, p_{01}, ...\}$. `T0` starts extruding at $p_{00}$ which will be indicated by a G-code like:

```
T0
G1 Xxpos1 Ypos1 Zzpos1 Eepos1 Frate1
```

where `xpos1` and `ypos1` are the $x$ and $y$ components of the projection of the next point of intersection in the list ($p_{10}$) on $r_0$, and `zpos1` is the $z$ height of the current slice. At $p_{10}$, `T1` will also start extruding, which will be indicated by an instruction wrapped by `M605 S2`:

```
M605 S2
G1 Xxpos2 Ypos2 Zzpos2 Eepos2 Frate2
M605 S2
```

where `xpos2` and `ypos2` are the $x$ and $y$ components of the of the projection of the next point of intersection in the list, $p_{11}$ on $r_0$. This instruction indicates *simultaneous* extrusion by both `T0` and `T1`. Once both extruders reach the projection of $p_{11}$ on $r_0$, `T1` stops extruding while `T0` continues until $p_{01}$. Once the entire list of points is visited, the extruders repeat this process in the opposite direction (shifted by some amount determined by the infill percentage). Throughout this infill process, the extruded lines of `T0` and `T1` are separated by $\delta sin(\theta)$.

While our prototype is only a proof of concept, it demonstrates the potential of parallel printing on desktop 3D printers. We focused only on infill with linear moves for simple shapes; perimeter parallelization is much more challenging (but also usually corresponds to a smaller percentage of the total print time, particularly for large prints, because area scales faster than perimeter). Furthermore, for complex shapes, especially with multiple components or holes in the design, this simple approach does not generate parallel G-code; there is much room for improvement and exploration! We hope future research will help build a more general code generator that works on different machines and complex designs, potentially taking into account the shape of the model and how it might best be parallelized.

## 6 Related Work

Nandi et al. [15] presented early ideas on parallel 3D printing but did not propose concrete strategies for implementing such a system. Mhatre et al. [14]'s work on concurrent multi-head extrusion explores various extruder arrangements and the constraints they introduce for the printing direction. They target a 4-axis 3D printer where rotational motion is required, whereas our current prototype is a 3D printer with only x, y, and z movements. More importantly, our approach does not require a rotational axis to achieve parallelism, although we discuss other potential machine configurations in Section 3. The authors also support parallel printing the perimeter — they assume that each layer will have two perimeters (which is a common scenario) and use two extruders to print them simultaneously. Our system does not currently support parallel perimeters; we leave that for future work. Mhatre et al. [14] define a new tool in the firmware that indicates all nozzles should be turned on simultaneously which is different from our approach of using `M605 S2`. One of the biggest differences between our approach and Mhatre et al. [14] is that their toolpath generator performs a post-pass on the G-code for a 3-axis printer whereas we directly generate the parallel toolpath for each slice by computing the start and end positions of the extruders. This entails a very different approach to toolpath computation.

## 7 Conclusions

This paper sets out a research vision for investigating parallel fabrication techniques for desktop 3D printers. We proposed a concrete plan including challenges and mitigation strategies for each stage of the desktop fabrication pipeline. We presented our findings from an early investigation of this research direction that we hope will inspire further research in design and implementation of hardware, firmware, slicer, and CAD tools for allowing parallel 3D printing, thereby making desktop fabrication faster and more scalable.

## Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1813166. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

## References

[1] SIMPLIFY3D, 2021. https://www.simplify3d.com/.

[2] All3DP. Dual Extruder (3D Printing): All You Need to Know, 2021. https://all3dp.com/2/dual-extruder-extrusion-3d-printer-simply-explained/.

[3] BNC3D. BNC3D SIGMAX R19, 2020. https://www.bcn3d.com/bcn3d-sigmax-r19/.

[4] M. Carton. Dual extruder mount for mini-mpcnc (1/2 emt variant), 2021. https://www.thingiverse.com/thing:4883743.

[5] DesignNews. Project Escher Applies Parallel Processing to 3D Printing, Augut, 2016. https://www.designnews.com/design-hardware-software/project-escher-applies-parallel-processing-3d-printing.

[6] T. Du, J. Priya Inala, Y. Pu, A. Spielberg, A. Schulz, D. Rus, A. Solar-Lezama, and W. Matusik. Inversecsg: automatic conversion of 3d models to csg trees. pages 1–16, 12 2018.

[7] Flashforge. Flashforge creator dual extruder 3d printer, 2020. https://flashforge-usa.com/products/creator-dual-extrusion-3d-printer-certified-refurbished.

[8] F. H. Fossdal, J. Dyvik, J. A. Nilsson, J. Nordby, T. N. Helgesen, R. Heldal, and N. Peek. Fabricatable machines: A toolkit for building digital fabrication machines. In *Proceedings of the Fourteenth International Conference on Tangible, Embedded, and Embodied Interaction*, TEI '20, page 411–422, New York, NY, USA, 2020. Association for Computing Machinery.

[9] LeapFrog. Bolt pro, 2020. https://www.lpfrg.com/products/leapfrog-bolt-pro/.

[10] Marlin. Configuring Marlin, 2021. https://marlinfw.org/docs/configuration/configuration.html.

[11] Marlin. G0-G1 - Linear Move, 2021. https://marlinfw.org/docs/gcode/M605.html.

[12] Marlin. M605 - Dual Nozzle Mode, 2021. https://marlinfw.org/docs/gcode/G000-G001.html.

[13] Marlin. Marlin Firmware, 2021. https://marlinfw.org/.

[14] P. S. Mhatre. Process planning for concurrent multi-nozzle 3d printing, 2019. https://scholarworks.rit.edu/theses/10075/.

[15] C. Nandi, A. Caspi, D. Grossman, and Z. Tatlock. Programming language tools and techniques for 3d printing. In B. S. Lerner, R. Bodík, and S. Krishnamurthi, editors, *2nd Summit on Advances in Programming Languages, SNAPL 2017, May 7-10, 2017, Asilomar, CA, USA*, volume 71 of *LIPIcs*, pages 10:1–10:12. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[16] C. Nandi, J. R. Wilcox, P. Panchekha, T. Blau, D. Grossman, and Z. Tatlock. Functional programming for compiling and decompiling computer-aided design. *Proc. ACM Program. Lang.*, 2(ICFP):99:1–99:31, July 2018.

[17] C. Nandi, M. Willsey, A. Anderson, J. R. Wilcox, E. Darulova, D. Grossman, and Z. Tatlock. Synthesizing structured cad models with equality saturation and inverse transformations. In *PLDI '20*, PLDI '20, 2020.

[18] OpenScad. OpenScad. The Programmers Solid 3D CAD Modeller, 2021. http://www.openscad.org/.

[19] PyMesh. PyMesh — Geometry Processing Library for Python, 2021. https://pymesh.readthedocs.io/en/latest/index.html.

[20] N. Raynaud. G-code q'n'dirty toolpath simulator, 2021. https://nraynaud.github.io/webgcode/.

[21] Repetier. Ditto printing, 2020. https://forum.repetier.com/discussion/4140/ditto-printing-mixing-extruder.

[22] Repetier. Repetier-Firmware, 2021. https://www.repetier.com/documentation/repetier-firmware/.

[23] RepRap. G-code, 2021. https://reprap.org/wiki/G-code.

[24] Rhinoceros. Rhinoceros, 2021. https://www.rhino3d.com/.

[25] SketchUp. SketchUp, 2021. http://www.sketchup.com/.

[26] M. A. Skylar-Scott, J. Mueller, C. W. Visser, and J. A. Lewis. Voxelated soft matter via multimaterial multinozzle 3d printing. *Nature*, 575(7782):330–335, 2019.

[27] Slic3r. Slic3r: Open source 3d printing toolbox, 2021. https://slic3r.org/.

[28] Snapmaker. Modular 3-in-1 3d printers, 2020. https://snapmaker.com/platform/?gclid=EAIaIQobChMIp7-KlZPy5wIVbRitBh3H-QvvEAAYASABEgJk_fD_BwE.

[29] C. Software. Cura Software, 2021. https://ultimaker.com/en/products/cura-software.

[30] Solidworks. Solidworks, 2021. http://www.solidworks.com/.

[31] Thingiverse. Thingiverse, 2021. https://www.thingiverse.com/.

[32] J. Tran O'Leary, K. Lee, and N. Peek. A grammar of digital fabrication machines. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI EA '21, New York, NY, USA, 2021. Association for Computing Machinery.

[33] J. Tran O'Leary and N. Peek. Machine-o-matic: A programming environment for prototyping digital fabrication workflows. In *The Adjunct Publication of the 32nd Annual ACM Symposium on User Interface Software and Technology*, UIST '19, page 134–136, New York, NY, USA, 2019. Association for Computing Machinery.

[34] Ultimaker. Reliable 3d printers that simply work for you, 2020. https://ultimaker.com/en/resources/52867-dual-extrusion.

[35] V1Engineering. The mostly printed CNC, 2021. https://docs.v1engineering.com/mpcnc/intro/.

[36] C. Wu, H. Zhao, C. Nandi, J. I. Lipton, Z. Tatlock, and A. Schulz. Carpentry compiler. volume 38, pages 195:1–195:14, 2019.