Программное обеспечение 3D-принтера по металлу

Eugeny L. Romanov, Vadim G. Tokarev, Yuliya V. Novitskaya
Novosibirsk State Technical University, NSTU
Novosibirsk, Russian Federation,
kafedra_vt@vt.cs.nstu.ru

Ivan A.Bezrukov,
Vladimir N. Filimonenko
line 1 (of *Affiliation*): dept. name of organization
Electroplasma Equipment & Systems EPOS
Novosibirsk, Russian Federation
office@epos-nsk.ru

***Abstract* — Рассматривается аппаратно-программная архитектура 3D-принтера по металлу с технологией Selective Laser Melting (SLM), состоящая из контроллера и хост-компьютера. Программное обеспечение 3D-печати для хост-компьютера разработано на Java и выполняет функции: просмотр 3D-моделей, слайсинг модели в формате STL в файл собственного формата, 2D-редактирование и коррекция данных в слоях, управление контроллером в процессе печати, генерация тестовых моделей. Встраиваемое программное обеспечение контроллера выполняет прием команд и данных от хоста, управление зеркалами системы позиционирования лазерного луча в двух координатах, а также управление перемещением столика с образцом и механизмов засыпки слоя. Протокол связи, построенный поверх USB или UDP(TCP/IP), обеспечивает выдачу настроек и параметров печати, мониторинг состояния, управление потоком команд печати в контроллер.**

Рассматриваются алгоритмы слайсинга для различных вариантов штриховки и перемещения луча. Показаны примеры моделей, в которых требуется ручная коррекция генерируемых контуров. Приведены показатели текущей производительности слайсинга и печати, Рассматриваются вопросы технологии программирования (проектирования) и управления производительностью ПО - использование паттернов, графическое индексирование, профилирование производительности, параллельная обработка.

Keywords— selective laser melting, 3D-slicing, additive technologies, software performance, software profiling, parallel processing, programming patterns

Селективное лазерное плавление, 3D-слайсинг, аддитивные технологии, производительность ПО, профилирование ПО, параллельная обработка, паттерны программирования.

I. Введение

Процесс 3D-печати по металлу по технологии селективного лазерного плавления (SLM) [1] представляет собой послойное наращивание детали путем сплавления слоя порошка под воздействием луча лазера в атмосфере инертного газа. Рабочий стол с подложкой, на которой наплавляется деталь, и емкость для порошка имеют подвижное дно, управляемое шагово-винтовыми приводами. Механизм разравнивания также управляется шаговым приводом: он производит нанесение очередного слоя порошка при поднятии дна емкости при опускании подложки с деталью. Модулируемый луч лазера проходит через сканатор, который задает нужное направления луча, и блок фокусировки. Сканатор состоит из двух зеркал с управляемыми углами поворота. Сканатор и лазер имеют собственные системы управления (контроллеры). Всеми перечисленными устройствами, датчиками и исполнительными механизмами управляет контроллер принтера (далее, контроллер). В хост-компьютере под Windows 7 развернут пакет ПО 3D-печати. Взаимодействие ПО и контроллера осуществляется по протоколу, построенному поверх USB или UDP (TCP/IP).

В основные функции ПО 3D-печати входят: просмотр 3D-моделей, слайсинг модели в формате STL в файл собственного формата, 2D-редактирование и коррекция данных в слоях, управление контроллером в процессе печати, генерация тестовых моделей, хранение настроек и т.д..

Встраиваемое ПО контроллера выполняет тестирование исполнительных механизмов, прием команд, прием и буферизацию данных от хоста, управляет в режиме реального времени процессом печати (модуляция лазерного луча, поворот зеркал сканатора, перемещение столика с образцом и механизмов засыпки слоя).

Протокол обеспечивает выдачу настроек и параметров печати, мониторинг состояния, управление потоком данных печати в контроллер.

A. Терминология

Слайсинг – разбиение исходной 3D-модели на слои, с последующим выделением контуров и их заливкой. Как правило, слайсинг слоев является независимой процедурой для каждого слоя. При наличии средств измерения и контроля процесса печати возможна корректировка данных слайсинга последующих слоев.

Оконтуривание – предварительная печать контура в текущем слое модели, в SLM не является обязательной операцией.

Линия прожига – отрезок прямой, проходимой лучом лазера по поверхности порошка. Состоит из последовательности точек сплавления, расположенных вдоль линии перемещения.

Заливка – множество линий прожига слоя, поверхность сплавления.

Штриховка (стратегия сканирования) – способ заполнения поверхности сплавления линиями прожига: линейная (рис.1), шахматная (клетка, рис.2), случайная, максимальной длины и др.

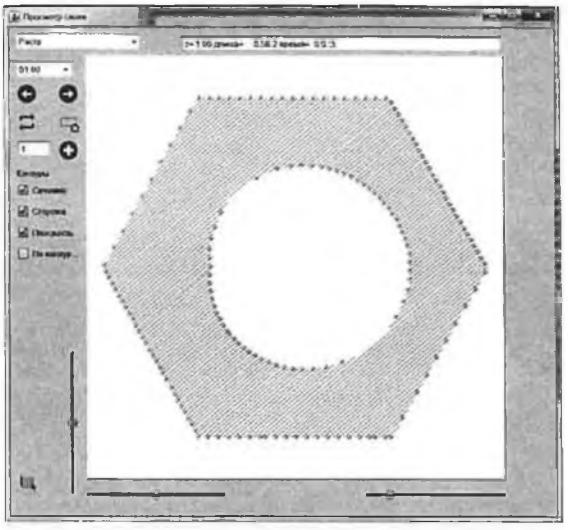

Рис.1. Линейная штриховка

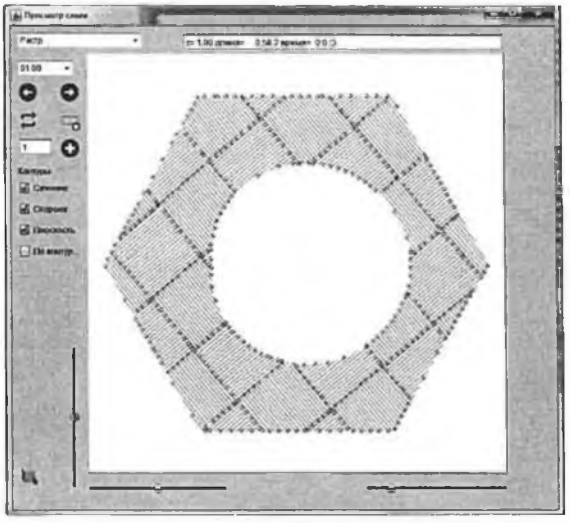

Рис.2. Шахматная штриховка

Вид перемещения – способ перемещения от конца линии прожига к началу другой. На способ перемещения влияют два фактора: уменьшение холостого хода луча и влияние способа перемещения на процессы разогрева поверхности слоя. Возможно перемещение луча в одном направлении (вид перемещения - растр, рис.3), во встречных направлениях (вид перемещения - зигзаг, рис.4), а также непрерывное перемещение - зигзаг с соединением линий прожига одной группы.

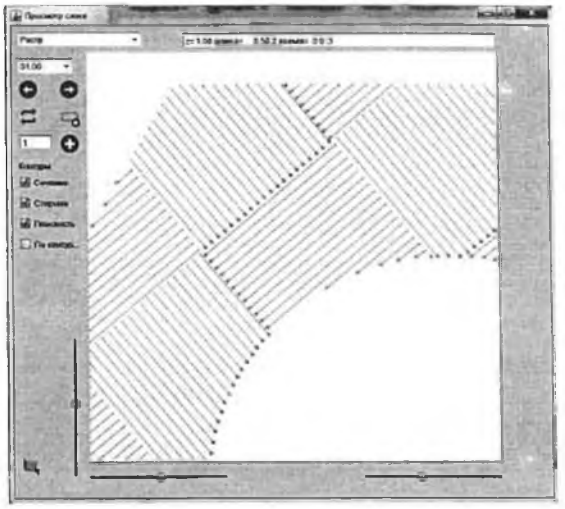

Рис.3. Вид перемещения – растр

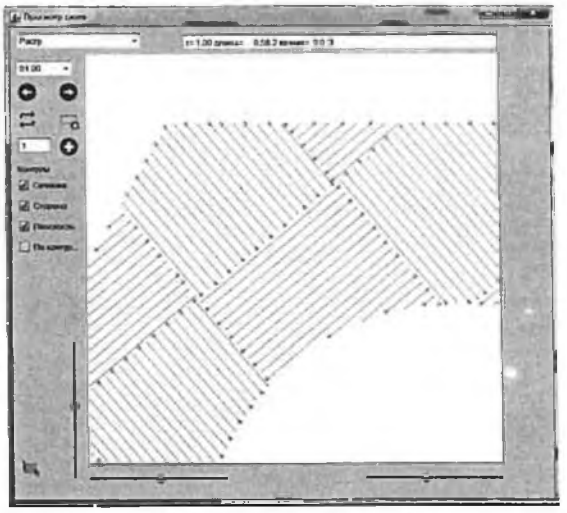

Рис.4. Вид перемещения – зигзаг

II. Программная архитектура 3D-принтера

A. ПО 3D-печати

Основные функциональные возможности ПО 3D-печати:

- загрузка исходной 3D-модели в формате STL и ее просмотр в 3D (3D-визуализация);
- простые манипуляции с моделью - нормирование размеров, центровка, поворот вокруг осей;
- хранение глобальных и локальных настроек принтера и параметры слайсинга модели;
- слайсинг модели, сохранение модели, данных слайсинга и локальных настроек в двоичный файл в собственном формате (slm3d-файл);

- подсчет и прогнозирование параметров печати модели: общая длина линий прожига, оценка времени печати, процент холостого хода;
- просмотр данных слайсинга по компонентам слоя (сечение, контуры, растр), редактирование контуров и штриховки слоя и слайсинг отдельного слоя при редактировании контуров, слайсинг при добавлении слоя;
- визуализация процесса печати slm3d-файла;
- генерация тестовых моделей в формате STL;
- прямое управление контроллером, выдача всех команд с необходимыми параметрами, мониторинг состояния контроллера и процесса печати;
- передача slm3d-файла в принтер, управление буферизацией данных печати в контроллере;
- исполнение настроечных тестов для проверки технологических режимов печати.

B. ПО контроллера

Контроллер принтера выполнен на базе микроконтроллера ARM архитектуры STM32F769BI [2,3]. Шаблон встраиваемое ПО подготовлен с использованием генератора исходных кодов STM32Cube-MX [4] с использованием библиотеки HAL фирмы STMicroelectronics. ПО контроллера разработано на Си и представляет собой внесистемную (standalone) программу, работающую в режиме реального времени. В ее основе лежит событийная модель и диспетчер, выполняющий цикл опроса источников событий. ПО состоит из следующих структурно-функциональных компонент:

- программные модули USB интерфейса;
- программные модули UDP (TCP/IP) интерфейса;
- диспетчер команд протокола. Выполняет декодирование принятых по интерфейсу USB/UDP команд, инициализацию или исполнение соответствующих им операций и формирование ответов;
- модуль команд длительного исполнения и отложенных команд, а также начальных настроек;
- модуль функций преобразования координат и прожига треков. Производит преобразование линейных координат точек линии в координаты позиционирования лазерного луча с учетом геометрии установки. Формирует временную развертку движения лазерного луча: линия прожига состоит из последовательности точек плавления, в которых луч приостанавливает свое движение, одновременно производится модуляция луча. Сами точки могут иметь несколько видов специфического размещения относительно линии;
- модуль выдачи координат позиционирования лазерного луча по последовательному аудио интерфейсу (Serial Audio Interface -SAI) в формате XY2-100;
- модуль взаимодействие с исполнительными механизмами: включение и отключение лазера, контроль датчиков позиционирования моторов.

C. Протокол обмена контроллер-хост

Протокол взаимодействия контроллера и ПО 3D-печати построен по принципу синхронного запроса программы и ответа контроллера. Асинхронные сообщения от контроллера исключены, поскольку недопустимы при реализации в USB (хотя и возможны в UDP).

Поведение контроллера описывается диаграммой состояний. Состояния контроллера связаны с исполнением тестов оборудования, продолжительных по времени команд (засыпка слоя порошка), аварийными ситуациями, ожиданием данных, процессом печати слоя и т.п. В связи с этим драйвер протокола содержит методы ожидания перехода из состояния в состояние с устанавливаемыми программными тайм-аутами со значительными интервалами (порядка десятков секунд).

Драйвер протокола управляет потоком данных печати (команды прожига линий) и записью их в буфер контроллера, дожидается окончания печати содержимого буфера перед сменой слоя, производит накопление команд прожига линий в режиме блочной передачи.

Система команд протокола включает в себя:

- жесткий и мягкий сброс контролера и интерфейса;
- опрос состояния;
- передача команд печати (прожиг линии, смена слоя), контроль свободной памяти контроллера;
- блочная передача команд прожига линий;
- управление печатью: старт, отмена (останов), останов по завершению печати слоя, пауза и возобновление;
- запуск тестов оборудования;
- передача параметров и уставок контроллера;
- управление лазером и моторами.

III. Алгоритмы слайсинга

A. Слайсинг 3D-модели

Исходная 3D-модель загружается в STL-формате, который имеет достаточно простую структуру. Каждая поверхность образована множеством элементарных треугольников. Сам формат не выделяет поверхности, а хранит все треугольники единым массивом. Таким образом, топологическая структура модели не передается от CAD и не может быть использована при слайсинге. Сам слайсинг состоит из следующих этапов:

- определение отрезков – пересечений элементарных треугольников с горизонтальной плоскостью слоя;

- составление и сглаживание контуров из полученных отрезков;
- заливка контуров линейной штриховкой;
- получение шахматной и др. видов штриховки путём нарезки из линейной;
- оптимизация перемещений.

Получение сечений, составление и сглаживание контуров. Сечение модели плоскостью создается как множество отрезков - пересечений элементарных треугольников модели. Возможны 3 варианта пересечения. Треугольник:

- лежит полностью в плоскости сечения;
- соприкасается одной стороной с плоскостью сечения;
- пересекается плоскостью.

Первый вариант соответствует горизонтальной плоскости модели, эти треугольники создают отдельные контуры. Остальные варианты пересечения учитывается опционально: контуры могут состоять из пересечений одного либо различных видов.

Контуры горизонтальных плоскостей составляются путем присоединения очередного треугольника к существующему контуру при совпадении одной из его сторон с отрезком контура.

Составление контуров производится в два этапа. Сначала из каждого отрезка сечения составляется отдельный незамкнутый контур, а затем производится их слияние по совпадению концов. Образующиеся замкнутые контуры удаляются. Процесс продолжается, пока имеет место слияние.

Линейная штриховка получается путем генерации множества линий с заданным углом наклона и шагом по всему рабочему полю. Определяются точки пересечения линий с отрезками контуров (рис.6), каждая пара точек пересечения образует линию прожига.

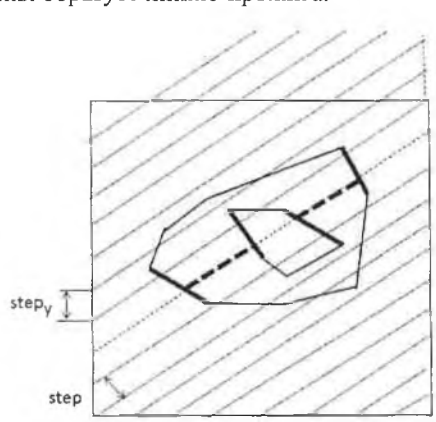

Рис.6. Алгоритм линейной штриховки

Алгоритм корректно работает для непересекающихся и вложенных контуров, учитывает взаимное расположение точек пересечения контуров на линии, фиксирует проблемные ситуации, корректирует их для нечетного количества точек.

Шахматная штриховка генерируется при помощи двух взаимно перпендикулярных линейных штриховок. Линии штриховки поворачиваются до горизонтального (вертикального) положения, затем на них накладывается «шахматная доска». Линии делятся на части, соответствующие «белым» («черным») клеткам, полученные отрезки поворачиваются на исходный угол.

Оптимизация перемещений производится путем поиска конца отрезка линии прожига, ближайшего к концу предыдущей. В результате при регулярной структуре штриховки получается вид перемещения – загзаг.

B. Проблемы слайсинга

Исходный 3D-формат STL не описывает напрямую топологическую структуру модели. Восстановление топологии модели из множества треугольников, составляющих ее поверхности, является достаточно сложной и трудоемкой задачей. В то же время простое построение множества контуров и их заливка могут происходить некорректно, если в слое оказываются контуры от пересечения несколько видов поверхностей: горизонтальных, сквозных и примыкающих. Рассмотрим конкретный пример (рис.7).

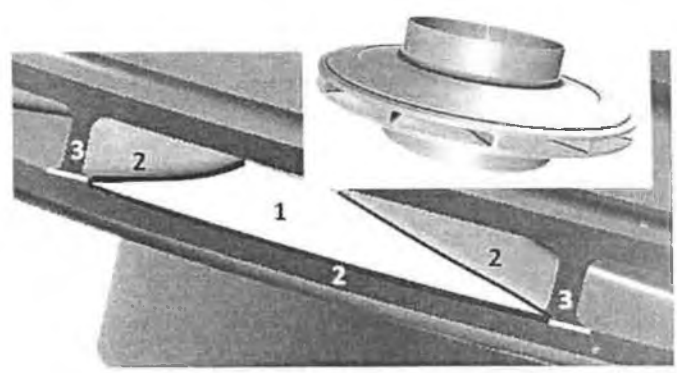

Рис.7. Слайсинг горизонтальной поверхности и ребер крыльчатки

Алгоритм слайсинга слоя на уровне нижней поверхности крыльчатки в месте ее сопряжения с ребрами обнаруживает контуры трех видов:

- контур горизонтальной поверхности (1) (рис.8);
- контур, образованный примыкающими вертикальными поверхностями, идентичный предыдущему (2) (рис.8);
- незамкнутые контуры пересечения вертикальных поверхностей (3) (рис.9).

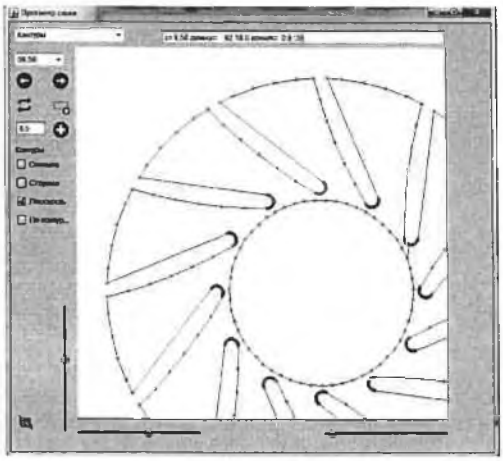

Рис.8. Контуры горизонтальной и примыкающих вертикальных поверхностей

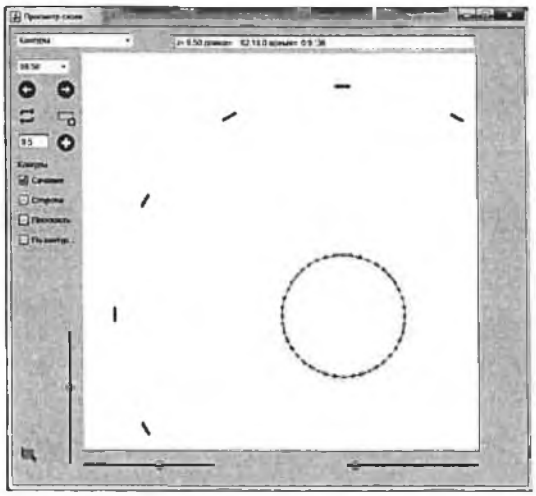

Рис.9. Незамкнутые контуры пересечения вертикальных поверхностей

В результате возникает проблемный слой (рис.10) с заливкой, «инвертированной» по отношению к соседним слоям.

Рис.10. Слайсинг предыдущего, проблемного и последующего слоя

В данном конкретном случае ошибка возникает при точном попадании в горизонтальный слой. Уже в процессе слайсинга группы слоев накапливаемая погрешность представления вещественного числа приводит к тому, что ошибка исчезает и генерируется заливка, соответствующая соседним слоям. Отсюда следует, что проблемный слой иногда возможно скорректировать, изменив высоту его сечения в пределах допустимой погрешности шага слайсинга и выполнив повторный слайсинг (например, с 9.5 до 9.5001 мм при шаге 0.05 мм).

Для коррекции заливки в подобных ситуациях в ПО 3D-печати включены средства 2D-редактирования данных слайсинга (рис. 1-4,8,9), включающие в себя:

- послойный просмотр заливки, контуров, отрезков сечения слоя, обнаруженных ошибок и коррекций, выполняемых алгоритмом слайсинга;
- редактирование контуров и составляющих их отрезков;
- повторный слайсинг отредактированных контуров;
- удаление слоев, добавление слоев на заданном уровне и их слайсинг.

IV. Технологические аспекты разработки ПО

ПО 3D-печати написано на Java, содержит около 9500 строк кода, 125 классов, около 1000 методов. Поддержка 3D-графики осуществляется с помощью коннектора к OpenGL - Java 3D [5]. Для работы с USB используется драйвер/библиотека программного доступа с C# и Java - LibUsbK [6] совместно с Java-библиотекой Usb4Java [7].

A. Структура проекта, паттерны проектирования

Классы проекта сгруппированы в функционально-ориентированные пакеты: экранные формы (представления, View), настройки, USB-интерфейс, протокол обмена, команды принтера и их фабрики, STL-формат и 2D-сущности слайсинга, алгоритмы и структуры данных слайсинга, средства 3D просмотра.

Использование паттернов ООП-проектирования [8] обеспечивает универсальность и развиваемость кода. Прежде всего, это ряд абстракций (интерфейсов, базовых классов), позволяющих создавать многообразия сущностей, таких как:

- конвертеры данных слайсинга в форматы принтера, GCODE, визуализатора, внутреннего представления в памяти;
- команды основных форматов, фабрики систем команд принтеров;
- драйверы интерфейсов USB,UDP(TCP/IP);
- 2D- и 3D-точки, ошибки слайсирования, генераторы моделей.

Аналогичным образом на основе интерфейсов обратного вызова (CallBack) создаются свободные коммуникации между функциональными компонентами, том числе:

- конвейер обработчиков событий слайсинга – позволяет подключать различные опции слайсинга (вид штриховки, оптимизация) путем связывания объектов классов-преобразователей в цепочку;

- уведомление представлений (View) о событиях, управление потоками из представлений (View) – приостановка, возобновление, отмена.

B. Графическое индексирование

В процессе слайсинга (построение контуров) и оптимизации перемещений (вид перемещения - зигзаг) производится поиск концов отрезка, точно совпадающих с заданной точкой, либо наиболее близко расположенной. Для исключения полного перебора отрезков в слое используются 2 варианта графического индексирования точек (концов отрезка) на плоскости. В обоих случаях производится ограничение пространства поиска и селекция подмножества отрезков, внутри которого делается полный перебор.

Вариант 1. Графический индекс содержит вектор ссылок на исходные объекты-отрезки, упорядоченные по одной из координат (x) одного из концов отрезка. При поиске конца отрезка, ближайшего к заданной точке, берется координата x этой точки и методом двоичного поиска ищется позиция отрезка в индексе с ближайшей координатой конца - i. Относительно i-ой позиции выбирается интервал i-k…i+k, в котором производится полный перебор отрезков уже со сравнением расстояний от конца каждого до заданной точки. Для поиска строятся два индекса – по одному для каждого из концов отрезка.

Вариант 2 – рабочая область разбивается на клетки (NxN), графический индекс содержит двумерный массив. Каждая ячейка массива соответствует отдельной клетке и содержит массив ссылок на отрезки, одни из концов которых попадает в соответствующую клетку. При оптимизации перемещений в выбирается ячейка графического индекса, соответствующая клетке, в которой находится текущая точка. Далее производится перебор всех ссылок в массиве выбранной ячейки.

C. Конвейер обработчиков событий слайсинга

Паттерн «обратный вызов» (CallBack) обеспечивает более свободную форму взаимодействия клиента с сервисом нежели обычный вызов. Обратный вызов определяется интерфейсом, который получает сервис: при возникновении событий или получении данных сервис вызывает соответствующие методы в интерфейсе. Клиентом передается объект-адаптер, в котором прописаны действия, производимые в контексте клиента при обратных вызовах.

В процессе слайсинга задействован интерфейс обратного вызова, в котором объявлены методы, связанные с событиями слайсинга: начало, окончание, генерация линии прожига, смена слоя, ошибка слайсинга, передача списка контуров и т.п.. Основной модуль, выполняющий алгоритм линейной штриховки, получает интерфейс, через который связывается с клиентом по соответствующим событиям.

Для выполнения других операций слайсинга (шахматная штриховка, оптимизация перемещений - зигзаг) создается конвейер адаптеров обратного вызова. Каждый объект-адаптер выступает в качестве сервиса для предыдущего и в качестве клиента для последующего. Адаптер оптимизации может производить частичную или полную буферизацию получаемых линий прожига и выдачу их в оптимальной последовательности. Адаптер шахматной штриховки разрезает отрезок линии прожига на несколько частей и генерирует соответствующее количество событий в предыдущем клиенте.

D. Производительность и оптимизация слайсинга

Предварительно оценить реальную производительность программы достаточно сложно. Можно приблизительно определить трудоемкость алгоритмов, т.е. зависимость количества базовых операций от размерности входных данных. Трудоемкость оказывает влияние на масштабирование данных (кратное увеличение их размерности) и позволяет прогнозировать узкие места на этапе проектирования. Однако она не дает ответа на вопрос, какова будет реальная производительность на реальных данных в выбранной архитектуре.

Профилирование альфа-версии. Если разработанное ПО не удовлетворяет по производительности, а предварительное прототипирование не проводилось, то прибегают к профилированию альфа-версии. Как правило, дефекты производительности обнаруживаются в самых неожиданных местах программного кода, в том числе в используемом стороннем исходном коде и библиотеках. В нашем случае имела место типичная ситуация: при ликвидации ограниченного числа узких мест удалось качественно повысить производительнсть. Ниже перечислены обнаруженные дефекты и способы их устранения.

1. *Ограниченная буферизация.* Первоначально оптимизация перемещений производилась путем накопления классом-адаптером линий прожига для всего слоя. При получении события окончания слоя они передавались предыдущему клиенту в последовательности с минимизированной суммой холостого хода. Ограничение буферизации в пределах 1000..2000 линий с циклом «накопление-оптимизация» дало возможность увеличить производительность за счет контролируемого снижения качества оптимизации;

2. *Графическое индексирование.* При оптимизации перемещений использовано графическое индексирование (см. выше, вариант 2) для поиска ближайшего конца линии

3. *«Логическое» удаление.* Обычное удаление на объекта-линии из стандартного класса - вектора ArrayList не по индексу, а по ссылке реализовано, по всей видимости, линейным алгоритмом с простым перебором и сравнением ссылок. Последовательное удаление всех линий из вектора имеет квадратичную трудоемкость, что на текущих размерностях данных проявляется как дефект производительности. Для устранения дефекта в каждый объект был введен логический признак удаления, позволивший отказаться от физического удаления ссылки.

4. *Промежуточная буферизация ввода/вывода.* Обычная сериализация объектов в двоичных потоках

DataOutputStream / DataInputStream создавала неприемлемые задержки при сохранении/загрузке slm3d-файлов. Использование промежуточной буферизации через потоки данных в памяти ByteArrayOutputStream / ByteArrayInputStream и последущие операции ввода/вывода накопленных байтных массивов позволили увеличить производительность операции на порядок.

5. *Эвристическая оптимизация слияния контуров.* В процессе составления контуров при слиянии разомкнутых пар использовался двойной цикл сравнения концевых точек незамкнутых контуров. При обнаружении совпадения и слияния пары процесс повторялся от начала (завершение метода и повторный его вызов). Дефект производительности состоял в том, что последующий вызов повторял заведомо нерезультативные сравнения. Продолжение внутреннего цикла после удаления второго контура и слияния пары (с коррекцией текущего шага) позволило проводить множество слияний за один вызов метода. Также весьма вероятно, что в исходном 3D-формате составляющие треугольники хранятся более-менее регулярно. Если это так, то при сечении плоскостью отрезки одного контура будут расположены в нужной последовательности и слияние будет производиться в последовательных шагах. Приведенный далее фрагмент трассировки с подсчетом шагов цикла между слияниями подтверждает это предположение и доказывает факт существования дефекта:

1 2 245 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2137 1 2 2378 2376 2373 1 2 2369 16 2352 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2351 2348 1 2 2345 2343 2340 1 2 2336 16 2319 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2318 2315 1 2 2312 2310 2307 1 2 2303 16 2286 1 1 1 1 1 1 1 1 1 1 1 1 2 2285 2282 1 2 2279 2277 2274 1 2 2270 16 2253 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2252 2249 1 2 2246 2244 2241 1 2 2237 16 2220….

Производительность до и после профилирования и оптимизации, а также основные параметры слайсинга приведены в табл.1. Параметры слайсинга:

- шаг растра - 0.020 мм;
- шаг слоя - 0.050 мм;
- тип штриховки - линейная, клетка 2 мм;
- память при запуске Java-программы - 1300 Мб;
- скорость прожига – 160 мм/с;
- процессоры - Intel Core i2-3220 3.3Ггц (1), Intel Core i3-6300 3.8 Ггц (2);
- количество Java-потоков при слайсинге – 10.

ТАБЛИЦА 1. ПРОИЗВОДИТЕЛЬНОСТЬ СЛАЙСИНГА, ОПТИМИЗАЦИЯ

Модель	L мм	W мм	H мм	Слоев	Общая длина линий, м	Время печати (оценка)	Размер файла, Мб	Штриховка	Время слайсинга (1)	Время слайсинга, альфа-версия	Эффект оптими-зации	Время слайсинга (2)
Корпус	55	34	22	440	13016	22:35:54	354	клетка	0:04:03	2:02:06 (0:11:36[a])	30,1 (2,9[a])	0:01:24
Корпус	55	34	22	440	13016	22:35:54	209	линейная	0:01:56	0:07:50	4,1	0:00:43
Крыльчатка (M 1:2)	58	58	21,8	435	12904	22:24:10	213	клетка	0:04:54	0:11:31 [a]	2,4 [a]	0:01:15
Крыльчатка (M 1:2)	58	58	21,8	435	12904	22:24:10	72	линейная	0:01:58	0:07:44	3,9	0:00:46
Крыльчатка (M=0.86)	100	100	37,6	752	66731	115:51:11	1612	клетка	0:24:52	--- [a,b]	---	0:13:39
Зуб	15	15	19,8	396	1732	3:00:29	99	клетка	0:03:19	0:09:10	2,8	0:01:05
Блок 2x2x1	20	20	10	200	4000	6:56:40	48	клетка	0:00:26	0:13:59	32,3	0:00:20
Гайка	5,77	5	6	120	341	0:35:33	8	клетка	0:00:18	0:00:46	2,6	0:00:12
Блок 9x9x4.5	90	90	45	900	364500	632:48:45	108	линейная	0:01:30	0:16:31	11,0	0:01:28

a. без оптимизации перемещений

b. переполнение памяти

Размерность динамической памяти Java также оказывает косвенное влияние на производительность. Даже при значительном превышении объема динамической памяти (кучи) над объемом данных слайсинга процесс утилизации промежуточных неиспользуемых данных (сбор мусора) оказывает влияние на производительность. А поскольку сборщик мусора запускается в Java при отсутствии свободной динамической памяти, подобный дефект связан с объемом памяти, выделяемой приложению при запуске.

Параллельная обработка является стандартным средством повышения производительности. В нашем случае слайсинг слоев является независимым по данным, поэтому распараллелирование алгоритма производится простым рефакторингом соответствующего метода класса. Реализован вариант параллельной обработки с

использованием потоков в Java со стандартными средствами параллельного исполнения на ядрах CPU, предоставляемыми ОС. Он дает приемлемые и предсказуемые результаты (табл.2). Производительность повышается пропорционально количеству потоков в Java, пока оно не превышает количество логических ядер процессора (в данном случае – 4). В дальнейшем влияние оказывает только эффект снижения времени простоя из-за блокировок потоков. В программе реализован пул потоков с буферизацией текущих результатов и отдельным потоком для их записи в последовательный файл в требуемом порядке.

ТАБЛИЦА 2. ПРОИЗВОДИТЕЛЬНОСТЬ СЛАЙСИНГА, ПАРАЛЛЕЛИЗМ

Threads	Intel Core i2-3220	Intel Core i3-6300
1	0:10:13	0:06:36
2	0:04:58	0:03:19
3	0:03:38	0:02:12
4	**0:03:19**	**0:01:45**
5	0:03:03	0:01:29
7	0:02:53	0:01:14
10	0:02:53	0:01:05
20	0:03:00	0:00:59

V. ПЕРСПЕКТИВЫ И АЛЬТЕРНАТИВЫ

Слайсинг в 3D-печати перестал быть экзотической задачей, требующей разработки уникального ПО. Есть возможность выбора альтернативных решений:

- универсальные коммерческие продукты под широкий набор технологий печати (Triangulatica [9], позиционируется как ПО с адаптацией под любое оборудование стороннего производителя);
- свободно распространяемые слайсеры, в том числе с открытым кодом под принтеры определенного производителя (Cura под принтеры Ultimaker [10]);
- openSource проекты различного масштаба и качества. Так, на github в категории «3D slice» найдено 434 проекта, из них 170 на Python (в т.ч Cura), 70 на C++ и 12 на Java: 2 экспериментальных слайсера, визуализатор 3D-печати из GCODE, визуализатор STL-файла [11].

В цели проекта входит сопровождение экспериментальных исследований в области SLM-технологий: интеграция со средствами измерений на базе интерференционной микроскопии для оперативной коррекции процесса спекания, апробация сторонних алгоритмов слайсинга [12], вариантов технологического процесса (вид штриховки, перемещения луча – в ПО 3D-печати, развертки движения лазерного луча – в контроллере). Это требует

Поэтому возможность выбора в рамках имеющейся программной архитектуры ограничена вспомогательными открытыми решениями в области 3D-графики [11], вычислительной геометрии и т.п..

REFERENCES

[1] Gibson I., Rosen D., Stucker B. Additive Manufacturing Technologies: 3D Printing, Rapid Prototyping and Direct Digital Manufacturing. Second Edition. — New York: Springer-Verlag, 2015. — XXI, 498 p. — ISBN: 978-1-4939-2112-6; ISBN: 978-1-4939-2113-3 (eBook).

[2] DS11532: Arm® Cortex®-M7 32b MCU+FPU, 462DMIPS, up to 2MB Flash/512+16+4KB RAM, USB OTG HS/FS, 28 com IF, LCD, DSI./Official site of STMicroelectronics © 2018. [Электронный ресурс]: режим доступа - http://www.st.com/resource/en/datasheet/stm32f765bg.pdf (дата обращения 10.8.2018);

[3] RM0410: STM32F76xxx and STM32F77xxx advanced Arm®-based 32-bit MCUs/Official site of STMicroelectronics © 2018. [Электронный ресурс]: режим доступа - http://www.st.com/resource/en/reference_manual/dm00224583.pdf (Дата обращения 10.8.2018);

[4] STM32Cube MCU Package for STM32F7 series (HAL, Low-Layer APIs and CMSIS (CORE, DSP, RTOS), USB, TCP/IP, File system, RTOS, Graphic - coming with examples running on ST boards: STM32 Nucleo, Discovery kits and Evaluation boards) /Official site of STMicroelectronics © 2018. [Электронный ресурс]: режим доступа - http://www.st.com/content/st_com/en/products/embedded-software/mcus-embedded-software/stm32-embedded-software/stm32cube-mcu-packages/stm32cubef7.html (дата обращения: 10.8.2018).

[5] Java 3D 1.5.1 [Электронный ресурс]: режим доступа - http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-java-client-419417.html#java3d-1.5.1-oth-JPR (дата обращения: 10.8.2018).

[6] A complete software solution for windows usb devices [Электронный ресурс]: режим доступа - https://sourceforge.net/projects/libusbk/ (дата обращения: 10.8.2018).

[7] Библиотека Java для работы с USB [Электронный ресурс]: режим доступа - http://usb4java.org/ (дата обращения: 10.8.2018).

[8] Романов Е. Л. Программная инженерия : учеб. пособие / Е. Л. Романов. - Новосибирск : Изд-во НГТУ, 2017. - 407 с. - (Учебники НГТУ). - ISBN 978-5-7782-3455-0.

[9] Triangulatica – новый двигатель 3D-технологий / Аддитивные технологии [Электронный ресурс]: режим доступа - https://additiv-tech.ru/publications/triangulatica-novyy-dvigatel-3d-tehnologiy.html (дата обращения: 10.8.2018).

[10] Ultimaker. World class open source 3D printers [Электронный ресурс]: режим доступа - https://github.com/Ultimaker (дата обращения: 10.8.2018).

[11] 3D-Viewer STL-файла на Java на основе Java 3D [Электронный ресурс]: режим доступа - https://github.com/ag88/stl-viewer (дата обращения: 10.8.2018).

[12] An Optimal Algorithm for 3D Triangle Mesh Slicing and Loop-Closure Rodrigo Minetto, Neri Volpato, Jorge Stolfi, Rodrigo M. M. H. Gregori1 and Murilo V. G. da Silva1 [Электронный ресурс]: режим доступа - http://www.dainf.ct.utfpr.edu.br/~murilo/public/CAD-slicing.pdf (дата обращения: 10.8.2018).