Non-Planar Tool-Path Algorithm for Fused Deposition Modelling

Simarpal Kalsi

May 19 2024

Abstract

The following white paper proposes an algorithmic solution to creating non-planar tool paths for FDM (Fused Deposition Modelling) 3D Printers. The computational geometry processes used to create the non-planar tool paths are documented here. By creating an iterative process that uses a series of complex surfaces modeled through surface fitting a NURBS surface patch (Non-Uniform Rational Basis Splines) whilst iteratively using mesh boolean operations to transform the original mesh until the non-planar slice normalizes so that the normal slicing methods can be used after plane normalization. This algorithm will be implemented within `libslic3r` so that `OrcaSlicer` can be used to output non-planar G-Code.

1 Introduction

Why NURBS

Utilizing a NURBS surface was preferred because of the complex deformation that can be achieved by using a NURBS surface. The goal is to utilize a parametrized NURBS surface to indicate the points in space where the toolhead will travel. NURBS deformation algorithms also allow for more delicate control of a transformed surface.

2 Facet Selection

The first step is to iterate over the triangles that represent the polygonal mesh and filter them out by calculating the normal of each facet and checking if it's within the bounds of constraints of the nozzle. This method will filter out all the triangles that cannot be printed by the nozzle. Furthermore, by calculating the Gaussian curvature of the vertices of the mesh, we can check for continuity of the mesh and extract subsets of the mesh that, whilst not connected, can be printed by a non-planar nozzle (30 degrees is the placeholder for nozzle constraints).

$$\text{paintedTris} = \emptyset$$

for each face $\in$ mesh.its.indices :

$$v_1 = \text{mesh.its.vertices[face[0]]}$$
$$v_2 = \text{mesh.its.vertices[face[1]]}$$
$$v_3 = \text{mesh.its.vertices[face[2]]}$$
$$\mathbf{n} = \frac{(\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1)}{\|(\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1)\|}$$
$$\mathbf{u} = (0, 0, 1)$$
$$\cos(\theta) = \mathbf{n} \cdot \mathbf{u}$$
$$\theta = \arccos(\cos(\theta))$$

if $\theta < 30°$:

$$\text{paintedTris} = \text{paintedTris} \cup \{\text{face}\}$$

return paintedTris

After painting the triangles that are within the constraints of the nozzle geometry, we must calculate the Gaussian curvature of the vertices in the painted facets. This is done in order to find subsets within the painted triangles that can be considered discontinuous. Further along the algorithmic pipeline, we will have to create a NURBS surface patch for each subset of the painted triangles in order to compute the non-planar tool path. Another thing to consider is that faces that have the normal pointing upward will also be painted using this method, but they have no need for a non-planar tool path. The triangles need to be painted beforehand because the Gaussian curvature is a localized mesh attribute.

$$\text{discontinuousFacets} = \emptyset$$

for each $v \in$ mesh.its.vertices :

$$\Theta_v = \sum \text{angles around } v$$
$$K(v) = 2\pi - \Theta_v$$

if $K(v) > \tau$ or $\exists$ neighbor with $|K(v) - K(\text{neighbor})| > \epsilon$:

mark v as having discontinuous curvature

for each face $\in$ mesh.its.indices :

if any vertex of face is marked as having discontinuous curvature :

$$\text{discontinuousFacets} = \text{discontinuousFacets} \cup \{\text{face}\}$$

return discontinuousFacets

After the subsets of the mesh that are suitable for computing a non-planar tool path are found, the algorithm extracts the vertices from each subset and uses them to compute a NURBS surface patch that must fit atop the aforementioned vertex subset. There are two methods to compute the NURBS surface patch for either the set of discontinuous painted facets or a single set of facets that

are within the bounds of the nozzle geometry estimated through the Gaussian curvature that can be computed using the algorithmic solution posted above.

There are two algorithms that can be utilized for finding out the control vectors and weights of a NURBS surface patch required for the initial step of the algorithmic process. The first method is using the least squares method to find the control vectors and the weights for the NURBS surface patch. The second method is to use a nonlinear least squares fitting of NURBS surfaces to measured data by the Gauss-Newton method. In the Gauss-Newton method, line search and regularization and trust region methods are used to reach global convergence as well as variable substitution and simple bounds.

3 NURBS surface fitting

NURBS Surface Fitting Using the Least Squares Method:

Given a set of vertices $\{\mathbf{v}_k\}_{k=1}^n$ that the surface must fit through, we aim to fit a NURBS surface $\mathbf{S}(u, v)$ to these vertices using the least squares method.

3.1 Define the NURBS Surface

A NURBS surface is defined as:

$$\mathbf{S}(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)\mathbf{P}_{i,j}w_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{i,j}}$$

where:

- $N_{i,p}(u)$ and $N_{j,q}(v)$ are the B-spline basis functions of degree p and q respectively,
- $\mathbf{P}_{i,j}$ are the control points,
- $w_{i,j}$ are the weights associated with the control points.

3.2 Formulate the Least Squares Problem

We aim to minimize the sum of squared residuals between the given vertices $\{\mathbf{v}_k\}_{k=1}^n$ and the points on the NURBS surface $\mathbf{S}(u_k, v_k)$.

Define the residuals as:

$$\mathbf{r}_k = \mathbf{v}_k - \mathbf{S}(u_k, v_k)$$

The objective is to minimize:

$$E = \sum_{k=1}^{n} \|\mathbf{r}_k\|^2 = \sum_{k=1}^{n} \|\mathbf{v}_k - \mathbf{S}(u_k, v_k)\|^2$$

3.3 Set Up the System of Equations

To minimize E, we need to solve for the control points $\mathbf{P}_{i,j}$ and weights $w_{i,j}$.

Formulate the system as a linear least squares problem:

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

where:

- $\mathbf{A}$ is the design matrix containing the basis functions evaluated at the parameter values (u_k, v_k),
- $\mathbf{x}$ is the vector of unknowns (including flattened control points and weights),
- $\mathbf{b}$ is the vector of the given vertices (flattened).

3.4 Solve the Least Squares Problem

Solve the system using a stable least squares solver:

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$$

Update the control points and weights:

$$\mathbf{P}_{i,j} = \mathbf{x}_{P_{i,j}}, \quad w_{i,j} = \mathbf{x}_{w_{i,j}}$$

3.5 Iterate Until Convergence

Repeat the process of setting up the system and solving until the residuals $\mathbf{r}_k$ are minimized to an acceptable level, or until other convergence criteria are met for each of the subset of discontinuious or single contunious printable geomtery.

4 Gauss-Newton Surface Fitting

The Gauss-Newton method is used for nonlinear least squares fitting of NURBS surfaces to measured data, employing techniques such as line search, regularization, and trust region methods to ensure global Convergence

5 NURBS Surface Interpolation

In this work, we focus on the interpolation of NURBS surfaces within the bounds of a fitted NURBS surface and a plane that spans across the x- and y-plane, with a single moving z-axis variable. The methodology involves iteratively creating NURBS surfaces that interpolate between the fitted surface and the plane. Each iteration introduces a new NURBS surface that lies between the two, and a cost function evaluates the distance between them to ensure compliance with the printability criteria of the 3D printer.

The volume between any two NURBS surfaces represents individual layers that the 3D printer will print. It is crucial that the volume between these NURBS surfaces maintains non-planar continuity within the 3D printer's capability. The final NURBS surface in this iterative process will transition into a flat planar layer, ensuring the topmost layer of the print is level.

This approach guarantees that each layer adheres to the required geometric constraints, thereby producing a print that is both accurate and within the printer's operational limits.

6 Printability Cost Function

To ensure the NURBS surfaces are continuous and the distance between each surface does not exceed the nozzle's maximum and minimum layer height allotment, we define a printability cost function. This function penalizes deviations from the desired layer height and discontinuities between surfaces.

6.1 Parametrization of NURBS Surfaces

NURBS surfaces are defined by control points, knot vectors, and basis functions. We denote a NURBS surface as $S(u, v)$, where u and v are the parameters.

6.2 Continuity Constraints

To ensure continuity between surfaces, we enforce C^0 or C^1 continuity:

- C^0 continuity: Ensures that the surfaces touch at their boundaries.
- C^1 continuity: Ensures that the surfaces touch and have the same tangent direction at their boundaries.

6.3 Distance Between NURBS Surfaces

The distance between two NURBS surfaces $S_i(u, v)$ and $S_{i+1}(u, v)$ should be within the nozzle's maximum (h_{max}) and minimum (h_{min}) layer height allotment.

6.4 Cost Function Definition

We define a cost function that penalizes deviations from the desired layer height and discontinuities between surfaces as follows:

$$\text{Cost} = \sum_i \left(\int_0^1 \int_0^1 |d(S_i(u,v), S_{i+1}(u,v)) - h| \, du\, dv + \lambda \int_0^1 \int_0^1 \left| \frac{\partial S_i}{\partial u} - \frac{\partial S_{i+1}}{\partial u} \right| + \left| \frac{\partial S_i}{\partial v} - \frac{\partial S_{i+1}}{\partial v} \right| du\, dv \right) \tag{1}$$

Where:

- $d(S_i(u, v), S_{i+1}(u, v))$ is the distance between the corresponding points on two consecutive surfaces.
- h is the desired layer height within h_{min} and h_{max}.
- λ is a weighting factor for the continuity term.
- The integrals ensure the cost is calculated over the entire parameter space of the NURBS surfaces.

6.5 Implementation

The cost function can be implemented in a computational framework as follows:

```
 function distance(S1, S2, u, v)
    return ||S1(u, v) - S2(u, v)||

function continuity(S1, S2, u, v)
    dS1_du = gradient(S1(u, v), axis=0)
    dS2_du = gradient(S2(u, v), axis=0)
    dS1_dv = gradient(S1(u, v), axis=1)
    dS2_dv = gradient(S2(u, v), axis=1)
    return ||dS1_du - dS2_du|| + ||dS1_dv - dS2_dv||

function cost_function(Surfaces, h, h_min, h_max, lambda_weight)
    cost = 0
    for i = 1 to length(Surfaces) - 1
        S1 = Surfaces[i]
        S2 = Surfaces[i+1]
        for u = 0 to 1 with step 0.01
            for v = 0 to 1 with step 0.01
                dist = distance(S1, S2, u, v)
                if dist < h_min or dist > h_max
                    cost += |dist - h|
                cost += lambda_weight * continuity(S1, S2, u, v)
    return cost
```

This approach guarantees that each layer adheres to the required geometric constraints, thereby producing a print that is both accurate and within the printer's operational limits.

6.6 Iterative process

New NURBS surfaces will be created iteratively that lie between the fitted surface and the plane. Each iteration introduces a new NURBS surface that lies between the two, and a cost function evaluates the distance between them to ensure compliance with the printability criteria of the 3D printer.

6.7 NURBS surface deformation untill cost function is minimized

The nurbs surface will be deformed using gradient decent untill the cost function is minimized. The gradient decent will be performed on the control points of the nurbs surface untill the printibility cost function is minimized.

$$\mathbf{P}_{ij}^{k+1} = \mathbf{P}_{ij}^{k} - \alpha \frac{\partial f}{\partial \mathbf{P}_{ij}}(\mathbf{P}^{k}) \tag{2}$$

where $\mathbf{P}_{ij}$ are the control points, k is the iteration index, α is the learning rate, and f is the cost function.

7 NURBS Surface Interpolation for 3D Printing

In this section, we discuss the methodology for NURBS surface interpolation with respect to a plane that spans across the x and y axes with a single moving z-axis variable. The primary objective is to iteratively create NURBS surfaces that are interpolated between a fitted NURBS surface and the plane. Each iteration introduces a new NURBS surface between the fitted surface and the plane. The cost function is employed to check the distance between these surfaces, ensuring compliance with the 3D printer's printability criteria.

The process begins with a fitted NURBS surface and a reference plane defined by:

$$z = c$$

where c is a constant. The iterative interpolation process generates intermediate NURBS surfaces, S_i, such that:

$$S_i = (1 - \alpha_i)S_f + \alpha_i P$$

Here, S_f represents the fitted NURBS surface, P is the reference plane, and α_i is a parameter that varies between 0 and 1, controlling the interpolation between the surface and the plane.

The cost function, $C(S_i)$, is defined to ensure that the distance between consecutive NURBS surfaces S_i and S_{i+1} remains within the printer's layer height capabilities. The cost function can be expressed as:

$$C(S_i, S_{i+1}) = |z(S_i) - z(S_{i+1})|$$

This cost function checks the vertical distance between the interpolated NURBS surfaces, ensuring that it remains within the maximum and minimum layer height allotment of the 3D printer.

The volume between any two NURBS surfaces, $V_{i,i+1}$, represents the individual layers that the 3D printer will print. The objective is to maintain non-planar continuity within the bounds of the 3D printer's capability, ensuring smooth transitions between layers. The final NURBS surface should normalize into a flat planar layer as:

$$S_n \to P$$

By iteratively adjusting the parameter α_i and using the cost function to guide the process, we can generate a sequence of NURBS surfaces that meet the 3D printer's printability criteria, ensuring that the resulting printed object maintains structural integrity and smooth surface quality.

This method of NURBS surface interpolation ensures that each layer produced by the 3D printer is within the printer's capability, ultimately leading to high-quality 3D printed objects with non-planar surface continuity.

8 References