SEI-SICITE 2021
Pesquisa e Extensão para um mundo em transformação

XI Seminário de Extensão e Inovação
XXVI Seminário de Iniciação Científica e Tecnológica
08 a 12 de Novembro - Guarapuava/PR

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS GUARAPUAVA

Fatiamento de malha triangular para manufatura aditiva por meio da estrutura de dados topológica DCEL

Triangle mesh slicing for additive manufacturing using the DCEL topological data structure

Luis Felipe Moro Coelho (orientado) [*], Ricardo Dutra da Silva (orientador)[†], Alan Jun Kowa Onnoda[‡], Henrique Romaniuk Ramalho[§], Neri Volpato[¶], Rodrigo Minetto[‖]

RESUMO

O processo de manufatura aditiva, também conhecido como impressão 3D, consiste na adição sucessiva de material, em camadas, para produzir um modelo tridimensional. Uma das etapas o processo de manufatura, conhecida como etapa de fatiamento, consiste em computar polígonos para representar o modelo em cada camada. Estes polígonos são computados a partir da intersecção de planos com um malha triangular do modelo, de forma que cada triângulo intersectado determina um segmento que compõe um polígono. No entanto, se os triângulos são processados em ordem arbitrária, é preciso recuperar as adjacências entre segmentos para formar os polígonos. Neste passo podem ocorrer erros de conectividade que levam ao não fechamento de um polígono. O método descrito neste artigo apresenta uma modificação para um algoritmo de fatiamento estado da arte. Uma estrutura de dados topológica é usada para representar a malha triangular de entrada e um algoritmo para computação dos polígonos é apresentado. O algoritmo ajuda a evitar segmentos desconexos utilizando informações de vizinhança entre triângulos da malha. O método proposto foi implementado e testado usando o software RP3, desenvolvido pelo Núcleo de Manufatura Aditiva de Ferramental (NUFER) da Universidade Tecnológica Federal do Paraná (UTFPR).

Palavras-chave: Algoritmo de fatiamento, manufatura aditiva, estrutura de dados DCEL.

ABSTRACT

The additive manufacturing process, also known as 3D printing, consists of successively adding material, in layers, to produce a three-dimensional model. One of the steps in the manufacturing process, known as the slicing step, consists of computing polygons to represent the model in each layer. These polygons are computed from the intersection of planes with a triangular mesh of the model, so that each intersected triangle determines a segment that makes up a polygon. However, if the triangles are processed in arbitrary order, it is necessary to retrieve the adjacencies between segments to form the polygons. In this step, connectivity errors may occur and lead to a polygon not being closed. The method described in this article presents a modification to a state-of-the-art slicing algorithm. A topological data structure is used to represent the input triangle mesh and an algorithm to compute the polygons is presented. The algorithm helps to avoid disconnected segments by using neighborhood information between mesh triangles. The proposed method was implemented and tested using the RP3 software, developed by the Núcleo de Manufatura Aditiva de Ferramental (NUFER) of the Federal University of Technology - Parana (UTFPR).

Keywords: Slicing algorithm, additive manufacturing, DCEL data structure.

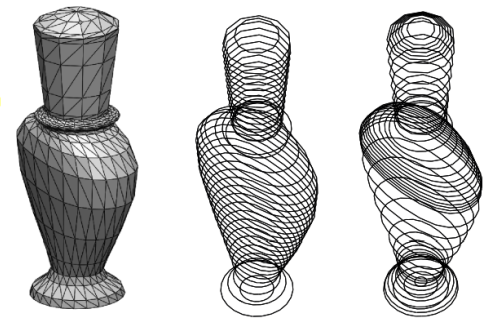
[*] Sistemas de Informação, Universidade Tecnológica Federal do Paraná, Curitiba, Paraná, Brasil; coelho.luisfelipe@gmail.com.
[†] Universidade Tecnológica Federal do Paraná, Campus Curitiba; rdsilva@utfpr.edu.br.
[‡] Universidade Tecnológica Federal do Paraná, Campus Curitiba; alanjun@outlook.com.
[§] Universidade Tecnológica Federal do Paraná, Campus Curitiba; henriqueramalho@alunos.utfpr.edu.br.
[¶] Universidade Tecnológica Federal do Paraná, Campus Curitiba; nvolpato@utfpr.edu.br.
[‖] Universidade Tecnológica Federal do Paraná, Campus Curitiba; rminetto@utfpr.edu.br.

1 INTRODUÇÃO

O processo de fabricação por manufatura aditiva, também chamado de impressão 3D, consiste na adição sucessiva de material na forma de camadas para produzir uma peça tridimensional. O planejamento do processo de manufatura é composto por etapas como orientação e posicionamento de um modelo 3D, fatiamento deste para produzir as camadas a serem impressas, cálculo de estruturas de suporte para regiões suspensas, planejamento da trajetória de impressão e geração de dados a serem enviados à máquina (VOLPATO, 2017).

A etapa de fatiamento é realizada sobre uma malha de triângulos que representa um modelo 3D. A imagem à esquerda da Fig. 1 mostra um exemplo de malha triangular. Um conjunto de planos horizontais é utilizado para seccionar a malha de triângulos e computar polígonos representando os contornos do objeto, como mostram as duas outras imagens da Fig. 1. Esses polígonos formam a base para outras etapas do planejamento de manufatura e representam regiões das camadas usadas para a impressão de um objeto.

Figura 1 – Exemplo de malha triangular à esquerda e dois possíveis fatiamentos da malha.



Fonte: (MINETTO et al., 2017)

Minetto et al. (2017) apresentam um algoritmo assintoticamente ótimo para o fatiamento de malhas triangulares. Dado um plano, o algoritmo identifica os triângulos intersectados por ele e computa segmentos de reta representando a intersecção com cada triângulo. Os segmentos são então conectados para formar os polígonos que representam os contornos de um objeto. A conexão dos segmentos resulta em polígonos orientados: no sentido anti-horário, para representar contornos de faces externas de um objeto, e no sentido horário, para representar contornos de faces internas (buracos ou túneis no objeto).

O algoritmo proposto por Minetto et al. (2017) utiliza uma malha de triângulos não organizada topologicamente, ou seja, a malha é representada por um conjunto de triângulos em que não há informação sobre quais triângulos compartilham uma aresta ou quais compartilham um vértice (falta de informações sobre vizinhanças). Desta forma, a ordem que os triângulos são processados é arbitrária e, consequentemente, os segmentos computados para formar um polígono também formam um conjunto em que não há informação sobre as adjacências dos segmentos. Para resolver este problema, e conectar segmentos adjacentes, Minetto et al. (2017) utilizam uma estrutura de dados *hash* para encontrar pontos compartilhados por segmentos. A orientação dos polígonos é obtida então utilizando um algoritmo de posição de ponto relativa a um polígono (SHIMRAT, 1962).

O método descrito para conexão dos segmentos, no entanto, enfrenta problemas quando a estrutura *hash* utilizada não encontra dois segmentos compartilhando um ponto. Como consequência, um polígono não consegue ser fechado e isso pode gerar erros nas etapas subsequentes do planejamento de manufatura. A falha

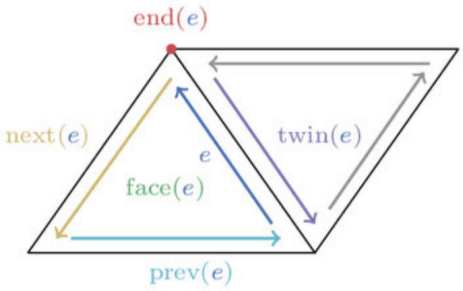
em encontrar segmentos compartilhando um ponto pode ocorrer por defeitos da malha triangular ou por que os pontos finais dos segmentos computados possuem certa imprecisão devido à representação em ponto flutuante. Seria então possível usufruir de informações topológicas para facilitar a conexão de segmentos que formam os polígonos gerados das fatias?.

Este artigo apresenta uma modificação do algoritmo de Minetto et al. (2017) para que a conexão dos segmentos que formam um polígono seja obtida da topologia de uma malha de triângulos. Os segmentos produzidos são computados percorrendo triângulos vizinhos em uma malha de triângulos e, como consequência, as adjacências entre segmentos são obtidas explicitamente. Para obter a topologia da malha triangular foi utilizada uma estrutura de dados *Doubly Connected Edge List* (DCEL) (BERG et al., 2008). A Seção 2 apresenta a estrutura DCEL e os algoritmos desenvolvidos. Na Seção 3 são mostrados resultados da implementação dos algoritmos. As conclusões do artigo são então apresentadas na Seção 4.

2 MÉTODO

A partir de um arquivo STL, um modelo 3D é representado como uma estrutura *Doubly Connected Edge List* (DCEL). A estrutura DCEL é formada por listas de faces, arestas e vértices. Como uma aresta limita duas faces, a estrutura de dados divide arestas em duas meias-arestas (half-edges), orientadas de um ponto de origem para um ponto de destino e tendo à sua esquerda a face à qual pertencem (ver Fig. 2). Dada uma meia-aresta e (Fig. 2), a meia-aresta $twin(e)$ representa a metade compartilhada com a face adjacente à face de e ($face(e)$). O vértice de origem da aresta e é dado por $orig(e)$, enquanto $next(e)$ e $prev(e)$ representam, respectivamente, a próxima meia-aresta e a meia-aresta anterior ao percorrer a face de e em sentido anti-horário. Ainda, considerando uma face f qualquer, é possível recuperar uma das meias-arestas que a formam por $boundary(f)$.

Figura 2 – Exemplo de estrutura de dados DCEL.



Fonte: (ZÖNNCHEN; KÖSTER, 2019)

O Algoritmo 1 descreve a função principal para fatiamento de uma malha triangular dada por uma DCEL M. Além da malha, o algoritmo recebe os planos P de fatiamento, representados pelas coordenas z em que cada um corta a malha triangular. Como resultado, o algoritmo computa os polígonos que formam as camadas fatiadas C. Para cada coordenada z de P (linha 1), o algoritmo encontra os triângulos da DCEL que são cortados pelo plano (linha 2). Esta operação é realizada usando o algoritmo de agrupamento de triângulos descrito em Minetto et al. (2017). A linha 3 do algoritmo inicia uma camada C_z para armazenar os polígonos computados para o plano z. O laço da linha 4 varre então os triângulos do plano, computando os polígonos (linha 6) e os orientando, na linha 7, em sentido anti-horário ou horário (este passo usa o mesmo algoritmo de Minetto et al. (2017)).

Para computar os polígonos, a topologia da DCEL é usada na função ProcessaPoligono, descrita no Algoritmo 2. O algoritmo inicia um polígono vazio (linha 1) e recupera uma meia-aresta do triângulo de entrada

Algoritmo 1: FATIAMENTO

Entrada: Malha DCEL M, Planos de fatiamento $F = z_1, z_2, \ldots, z_n$.
Saída: Camadas de polígonos de contorno $C = C_1, C_2, \ldots, C_n$.

```
1  para cada plano z em F faça
2      T ← TriangulosIntersectados(M,z)
3      IniciaCamada(C_z)
4      para cada triângulo t em T faça
5          se t não processado então
6              P ← ProcessaPoligono(t,z)
7              OrientaPoligono(P)
8              InserePoligono(C_z,P)
```

Algoritmo 2: PROCESSAPOLIGONO

Entrada: Triângulo t da DCEL M, plano de fatiamento z.
Saída: Polígono de contorno P contendo t.

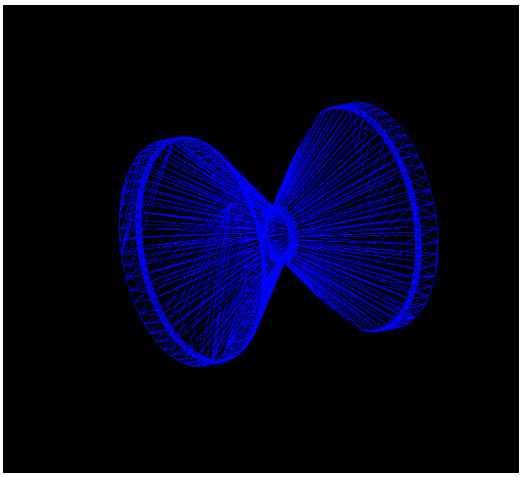
```
1  IniciaPoligono(P)
2  e ← boundary(t)
3  se z não intersecta aresta com pontos finais orig(e) e orig(twin(e)) então
4      e ← next(e)
5  inicial ← e
6  repita
7      InsereInterseccao(P,orig(e), orig(twin(e)))
8      MarcaTrianguloProcessado(face(e))
9      e ← next(twin(e))
10     se z não intersecta aresta com pontos finais orig(e) e orig(twin(e)) então
11         e ← next(e)
12 até e = inicial
```

(linha 2). As linha 3 e 4 garantem que a aresta chamada *inicial* na linha 5 seja intersectada pelo plano z. O laço iniciado na linha 6 insere o ponto de intersecção, entre a meia-aresta e e o plano, no polígono P. A linha 8 marca o triângulo ao qual a meia-aresta e pertence, isso é usado para controlar a linha 5 no Algoritmo 1 e evitar que novos polígonos sejam criados a partir de triângulos já processados (resultaria em um mesmo polígono). Na sequência (linhas 9 a 11), uma meia-aresta do triângulo vizinho, que também intersecta o plano z, é obtida para que seu ponto de intersecção seja incluído no polígono na iteração seguinte. O laço termina quando a meia-aresta atual e é igual à dada por *inicial* (linha 12), ou seja, o polígono foi fechado. O polígono resultante é formado pela sequência circular de m intersecções $P = i_1, i_2, \ldots, i_m$ computadas e seus segmentos a cada duas intersecções subsequentes i_k e i_{k+1}, considerando que se $k = m$, então $k + 1 = 1$.

3 RESULTADOS

O algoritmo desenvolvido foi implementado no software de planejamento de processo RP3 (*Rapid Prototyping Process Planning*), desenvolvido pelo Núcleo de Manufatura Aditiva de Ferramental (NUFER) da Universidade Tecnológica Federal do Paraná (UTFPR). O software é desenvolvido por alunos da UTFPR e sofre constantes aprimoramentos em suas funções. Após a implementação da estrutura de dados e dos algoritmos, foram realizados experimentos com exemplos de peças 3D para analisar qual seriam seus resultados dentro do software utilizando-se da nova estrutura. A Fig. 3 mostra uma malha triangular utilizando a DCEL.

Figura 3 – Exemplo de malha triangula usando a estrutura de dados DCEL no software RP3.



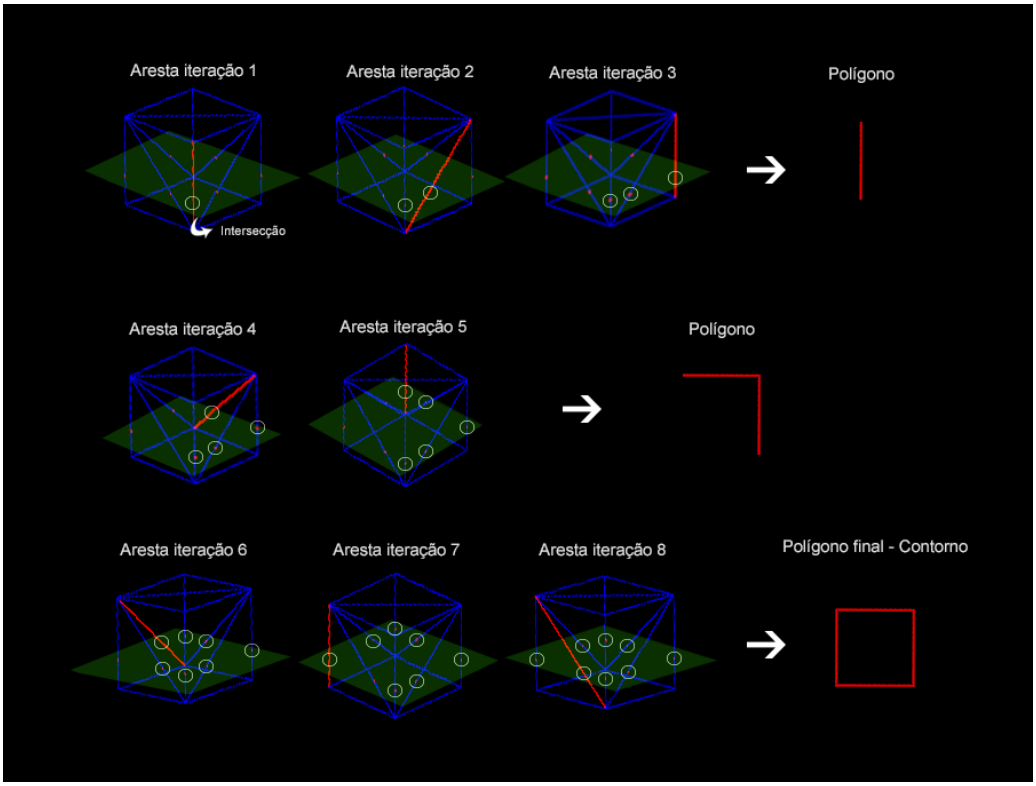
Fonte: Autoria própria (2021)

Outro resultado importante a ser explicitado é a demonstração da maneira como ocorre a identificação das intersecções utilizando-se da DCEL. A Fig. 4 demonstra as iterações da computação de um polígono para o modelo de um cubo. Dado um plano de intersecção, as arestas no percurso são mostradas em vermelho e os pontos de interseção já computados são mostrados com círculos verdes. Alguns polígonos parciais, resultantes destas intersecções, são mostrados ao lado dos cubos, desta maneira durante a navegação por meio da própria peça é realizada a construção passo a passo do objeto e todos os passos de sua impressão, assim seguidamente podendo calcular seus devidos suportes e também realizar o processo de preenchimento com as devidas estratégias,desta maneira tornando possível a impressão da peça com alto grau de fidelidade e precisão.

4 CONCLUSÕES

Este artigo apresentou uma mudança para um algoritmo de fatiamento no processo de planejamento de manufatura aditiva. A implementação da malha DCEL e dos algoritmos propostos, dentro do software RP3, demonstra a possibilidade de facilitar a criação de polígonos na etapa de fatiamento, fazendo com que o método seja mais preciso. As mudanças propostas tornam também possível evitar alguns dos problemas de fechamento de polígonos enfrentados anteriormente, o que impacta outras etapas no processo de planejamento, como o preenchimento dos polígonos e a geração de suportes. Esta pesquisa permite que na realização de trabalhos futuros, explore-se as informações de topologia dadas pela DCEL em outras etapas do planejamento de manufatura aditiva, como, por exemplo, na computação de estruturas suporte, podendo gerar suportes exatos para as necessidades de cada objeto e até mesmo nos processos de preenchimento de peça.

Figura 4 – Iterações do processo de fatiamento do RP3.



Fonte: Autoria própria (2021)

AGRADECIMENTOS

Agradecimentos ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pela bolsa de iniciação científica concedida ao projeto, aos professores Ricardo Dutra da Silva, Rodrigo Minetto e Neri Volpato, pela orientação, e a Alan Jun Kowa Onnoda e Henrique Romaniuk Ramalho, pela colaboração durante todo o desenvolvimento do projeto.

REFERÊNCIAS

BERG, Mark de et al. **Computational Geometry: Algorithms and Applications**. 3rd ed. Santa Clara, CA, USA: Springer-Verlag TELOS, 2008.

MINETTO, Rodrigo et al. An optimal algorithm for 3D triangle mesh slicing. **Computer-Aided Design**, v. 92, p. 1–10, 2017.

SHIMRAT, M. Algorithm 112: Position of Point Relative to Polygon. **Communications of the ACM**, Association for Computing Machinery, New York, NY, USA, v. 5, n. 8, p. 434, ago. 1962.

VOLPATO, Neri. **Manufatura Aditiva**: Tecnologias e Aplicações da Impressão 3D. São Paulo: Blucher, 2017.

ZÖNNCHEN, Benedikt; KÖSTER, Gerta. A parallel generator for sparse unstructured meshes to solve the eikonal equation. **Journal of Computational Science**, v. 32, p. 141–147, 2019.