Fecho Convexo 3D

Felipe A. Ferreira[1]

[1]PUC-Rio, Departamento de Informática, Rio de Janeiro, Brasil

Resumo

Este trabalho tem por objetivo apresentar a técnica de construção do fecho convexo 3D de forma incremental a partir de uma nuvem de pontos no espaço R^3. Para tal, primeiramente foi implementado uma versão em Python deste algoritmo e em seguida foram realizados experimentos com núvens aleatória de pontos com a finalidade de ilustrar a construção de tais fechos em espaços 3D e por fim foram realizados experimentos para avaliar o desempenho na construção de fechos convexos para um cubo e uma esfera com uma nuvem aleatória de pontos internos.

Palavras-chave: fecho, convexo, pontos, faces, polígono, cubo, esfera

Introdução

O fecho convexo $S \subseteq \mathbb{R}^3$ é o menor conjunto convexo de $\mathbb{R}^3$ que contém S. O fecho convexo de um conjunto de pontos é uma aproximação simples onde necessáriamente ele não ocupa mais espaço do que o próprio conjunto de pontos. No pior caso, o polígono tem o mesmo número de vértices do próprio conjunto. Computar o fecho convexo muitas vezes é um passo que precede outros algoritmos sobre conjunto de pontos. Por exemplo, uma das consequências do fecho convexo é a redução do número de pontos para representar objetos em uma imagem.

Na literatura[1] existem várias abordagens de algoritmos como o algorítmo incremental, Gift-wrapping, Divide-and-conquer e Graham scan. Este trabalho concentra-se na experimentação do algoritmo incremental de acordo com a sua descrição a seguir. Para alcançar esta finalidade, neste trabalho foi implementada uma versão deste algoritmo em Python e então a partir desta implementação, foram realizados experimentos para ilustrar e avaliar o desempenho da criação de fechos convexos 3D.

Descrição do Algoritmo

Segundo as definições da literatura[1] onde basicamente, a construção incremental do fecho convexo 3D pode ser resolvido de forma análoga ao fecho convexo 2D. O fecho convexo de pontos no $\mathbb{R}^3$ é fundamentalmente um objeto mais complexo que um polígono convexo, composto por vértices, arestas e faces, onde cada face é por si só um polígono convexo conforme pode ser visto na figura 1.

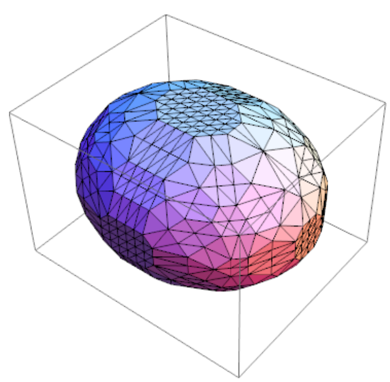


Figure 1. Exemplo do fecho convexo 3d

Dessa forma, o algorítmo incremental 3D funciona de forma análoga ao algorítmo incremental 2D. Intuitivamente, seguindo a definição de[1], o algoritmo funciona da seguinte maneira:

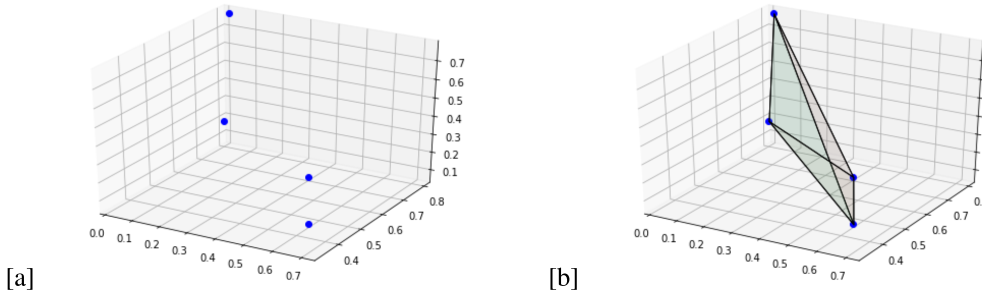


Figure 2. Construção do fecho convexo para uma nuvem de 4 pontos. a) nuvem de 4 pontos. b) fecho convexo encontrado

Seja um ponto $p = p_i$ e o fecho $Q = H_{i-1}$. Decidir se $p \in Q$. Se sim, descarte p, senão, computar o cone tangente a Q cujo o ápice é o ponto p, e então construir o novo fecho.

O teste para definir $p \in Q$, ou seja p está dentro de Q se e somente se p é o lado positivo de cada plano determinado por uma face de Q. O teste do triângulo esquerdo é baseado no volume do tetraedro determinado. Se todas as faces estão orientadas consistentemente, todos os volumes devem ter o mesmo sinal (positivo por convenção dos autores).

Quando p está fora de Q, o problema se torna mais difícil, onde o objetivo é então determinar quais faces de Q são visíveis a partir de p e quais não são visíveis. Então, a partir disso, encontrar as arestas de bordas para a construir o cone e descartar as faces internas desnecessárias.

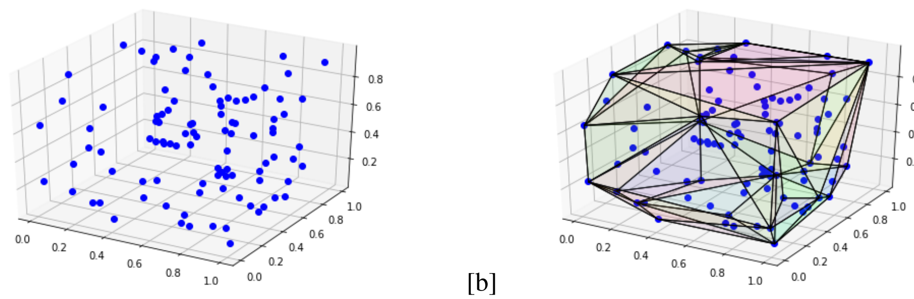
Formalmente, uma face é visível de p se e somente se algum ponto x interior a f é visível a partir de p, ou seja, px não cruza Q exceto $x : px \cap Q = \{x\}$. Dado esta definição, visualizar apenas a borda de uma face não torna a face visível e as faces vistas de frente também são consideradas invisíveis. Se a face de um triânculo (a, b, c) é visível a partir de p pode ser determinado de um volume de tetraedro assinado (a, b, c, p): ele é visível se e somente se o volume é estritamente negativo. Logo, o pseudo-código do algoritmo incremental pode ser ilustrado conforme o algoritmo 1

Algorithm 1 Algoritmo Incremental 3D

procedure INCREMENTAL3D(
)Inicializar H_3 = Tetraedro(p_0, p_1, p_2, p_3)
    for $i = 4, \ldots, n-1$ **do**
        for each face f of H_{i-1} **do**
            Computar o volume v do tetraedro determinado pela face f e p_i
            if $v > 0$ **then**
                Marque f como visível
        if nenhuma face é visível **then**
            Então descarte p_i (ele está dentro de H_{i-1})
        else
            for each border edge e of H_{i-1} **do** ▷ para cada aresta e na fronteira das faces visíveis
                Construa a face cone determinada por e e p_i
            for each visible face f **do**
                Remova f
            Atualize H_i

Resultados

Nesta seção é apresentao os resultados obtidos com a implementação do algoritmo do fecho convexo 3d. Esta implementação tem como base o código do algorítmo incremental em C descrito no livro **Computational Geometry in C (Joseph O'Rourke)**[2]. Primeiramente, para analisar o algoritmo de forma mais ilustrativa, foram gerados aleatóriamente duas nuvens de pontos que foram processadas pelo algoritmo a fim de obter o fecho convexo em 3D. A quantidade de pontos de cada uma foi 4 e 100 pontos respectivamente. A seguir são mostrados os gráficos das nuvens de pontos e os seus respectivos fechos convexos nas figuras 2 e 3.



[a] [b]

Figure 3. Construção do fecho convexo para uma nuvem de 100 pontos. a) nuvem de 100 pontos. b) fecho convexo encontrado

Sobre a perspectiva de desempenho, o algoritmo incremental é quadrático, ou seja, ele tem um limite superior de $O(n^2)$. Neste trabalho foram realizados dois experimentos, sendo o primeiro para a construção do fecho convexo de um cubo com uma nuvem de pontos aleatórios internos e o segundo experimento para a construção do fecho convexo de uma esfera com uma nuvem aleatória de pontos internos. A metodologia de experimentação consistiu em rodar uma sequência de execuções do algoritmo incremental para a criação dos fechos, onde cada execução usou um número específico de pontos. Desta forma, o algoritmo foi submetido a execuções com o número de pontos variando de 1.000 à 30.000 e a cada rodada foram obtidos seus tempos totais em segundos. A figura 4 mostra o comparativo dos tempos de execução em segundos para cada número de pontos do algoritmo incremental de construção dos fechos convexos 3D.

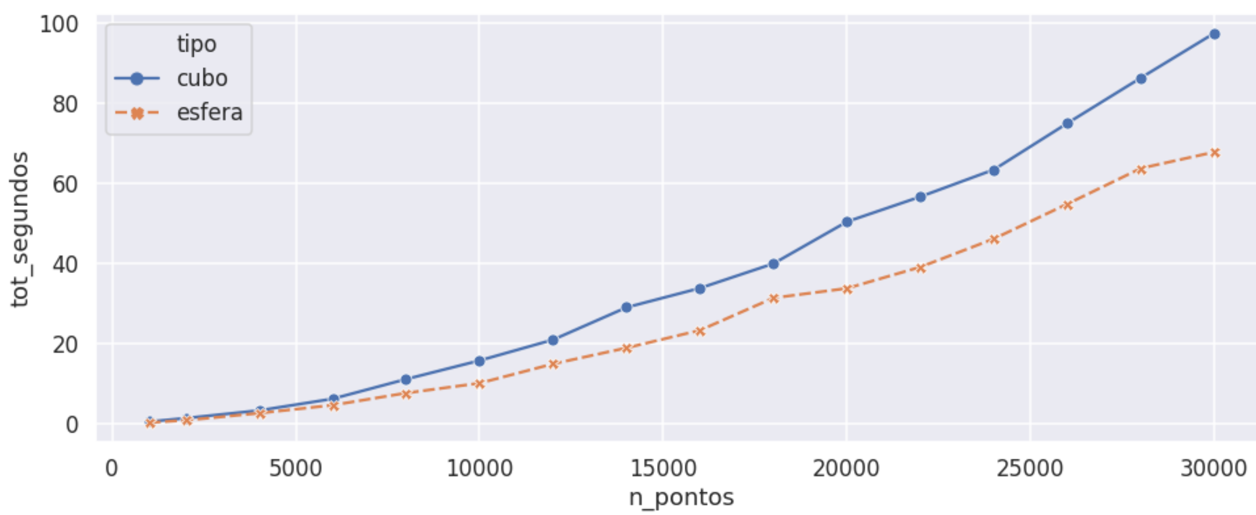


Figure 4. Comparativo de desempenho de execução do algoritmo incremental para a construção do fecho convexo para a esfera e o cubo com diferentes números de pontos internos. Estes pontos foram gerados de forma aleatória

Conclusão

Neste trabalho foi implementado a versão incremental do algoritmo de construção do fecho convexo 3D. Dentre as suas vantagens, ele diminui a quantidade de pontos necessários para representar um objeto e dessa forma é possível melhorar o desempenho de etapas posteriores de processamento de imagens. Uma possível aplicação desta técnica é a visibilidade de pontos onde dado uma nuvem densa de pontos P, o objetivo é determinar o conjunto de pontos visíveis V a partir de um ponto de vista c.

Foram realizados duas experimentações. A primeira com o objetivo de ilustrar a criação de fecho com uma nuvem de 4 pontos e a outra com 100 pontos respectivamente. O segundo experimento foi realizado para avaliar o desempenho do algoritmo na criação de fechos para o cubo e a esfera com diferentes números de pontos internos. Conforme pode ser visto na figura 4, é possível observar que em geral o algoritmo demonstrou seu comportamento quadrático. Um outro ponto interessante a observar é que, em geral os tempos para a construção do fecho do cubo foram superiores aos tempos da esfera. Uma primeira hipótese que surge é que uma grande proporção de pontos no cubo fazem parte do fecho convexo enquanto que esse número é menor na esfera. Contudo, é necessário uma análise mais detalhada como por exemplo realizar a contagem de pontos que fazem parte do fecho para uma melhor conclusão.

References

1. Devadoss, S. L. & O'Rourke, J. *Discrete and Computational Geometry.* (Princeton University Press, 2011).

2. O'Rourke, J. *Computational Geometry in C* (Cambridge University Press, New York, NY, USA, 1998), 2nd edn.