Geometria Computacional

Aula 21

ED para fecho convexo 3D

Cap 4 do livro do O'Rourke e
parte do Cap 11 do livro de de Berg et al.

Estrutura de dados

Problema: Dado um conjunto finito P de pontos no $\mathbb{R}^3$, encontrar o fecho convexo $\mathrm{conv}(P)$ dos pontos em P.

Estrutura de dados *winged-edge* (arestas aladas)

Estrutura de dados

Problema: Dado um conjunto finito P de pontos no $\mathbb{R}^3$, encontrar o fecho convexo $\mathrm{conv}(P)$ dos pontos em P.

Estrutura de dados *winged-edge* (arestas aladas)

A ED mantém uma lista de vértices, arestas e faces onde

Vértice. cada vértice v mantém as suas coordenadas (x, y, z) e um apontador $av(v)$ para uma aresta arbitrária incidente a v;

Estrutura de dados

Problema: Dado um conjunto finito P de pontos no $\mathbb{R}^3$, encontrar o fecho convexo $\mathrm{conv}(P)$ dos pontos em P.

Estrutura de dados *winged-edge* (arestas aladas)

A ED mantém uma lista de vértices, arestas e faces onde

Vértice. cada vértice v mantém as suas coordenadas (x, y, z) e um apontador $av(v)$ para uma aresta arbitrária incidente a v;

Face. cada face f mantém um apontador para uma aresta arbitrária $af(f)$ da fronteira de f;

Estrutura de dados

Problema: Dado um conjunto finito P de pontos no $\mathbb{R}^3$, encontrar o fecho convexo $\mathrm{conv}(P)$ dos pontos em P.

Estrutura de dados *winged-edge* (arestas aladas)

A ED mantém uma lista de vértices, arestas e faces onde

Vértice. cada vértice v mantém as suas coordenadas (x, y, z) e um apontador $av(v)$ para uma aresta arbitrária incidente a v;

Face. cada face f mantém um apontador para uma aresta arbitrária $af(f)$ da fronteira de f;

Aresta. cada aresta e tem oito apontadores...

Arestas aladas

Cada aresta e tem oito apontadores:

- Dois apontadores para os extremos $v_1(e)$ e $v_2(e)$ de e.
  A ordem destes vértices fornece uma orientação para e.

Arestas aladas

Cada aresta e tem oito apontadores:

- Dois apontadores para os extremos $v_1(e)$ e $v_2(e)$ de e.
  A ordem destes vértices fornece uma orientação para e.
- Apontadores $fccw(e)$ e $fcw(e)$ para as duas faces incidentes a e. A face $fccw(e)$ é a esquerda de $e = v_1(e)v_2(e)$ e a face $fcw(e)$ é a direita.

Arestas aladas

Cada aresta e tem oito apontadores:

- Dois apontadores para os extremos $v_1(e)$ e $v_2(e)$ de e.
  A ordem destes vértices fornece uma orientação para e.
- Apontadores $fccw(e)$ e $fcw(e)$ para as duas faces incidentes a e. A face $fccw(e)$ é a esquerda de $e = v_1(e)v_2(e)$ e a face $fcw(e)$ é a direita.
- Quatro apontadores para as asas (*wings*) de e:
  arestas que precedem e sucedem e em $fccw(e)$ e $fcw(e)$.

Arestas aladas

Cada aresta e tem oito apontadores:

- Dois apontadores para os extremos $v_1(e)$ e $v_2(e)$ de e.
  A ordem destes vértices fornece uma orientação para e.
- Apontadores $fccw(e)$ e $fcw(e)$ para as duas faces incidentes a e. A face $fccw(e)$ é a esquerda de $e = v_1(e)v_2(e)$ e a face $fcw(e)$ é a direita.
- Quatro apontadores para as asas (*wings*) de e:
  arestas que precedem e sucedem e em $fccw(e)$ e $fcw(e)$.

  Especificamente, $pccw(e)$ e $nccw(e)$ representam as arestas que precedem e sucedem e na face $fccw(e)$ (sentido anti-horário).

Arestas aladas

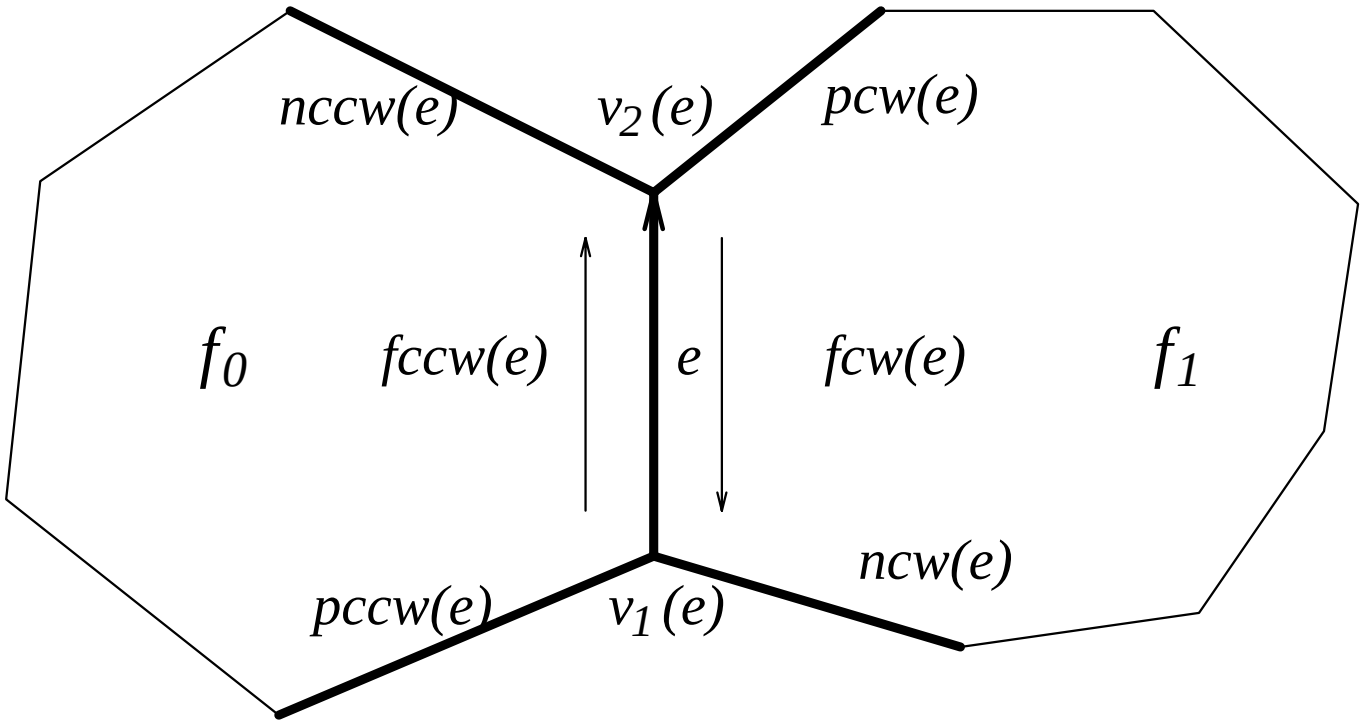
Cada aresta e tem oito apontadores:

- Dois apontadores para os extremos $v_1(e)$ e $v_2(e)$ de e.
  A ordem destes vértices fornece uma orientação para e.
- Apontadores $fccw(e)$ e $fcw(e)$ para as duas faces incidentes a e. A face $fccw(e)$ é a esquerda de $e = v_1(e)v_2(e)$ e a face $fcw(e)$ é a direita.
- Quatro apontadores para as asas (*wings*) de e:
  arestas que precedem e sucedem e em $fccw(e)$ e $fcw(e)$.

  Especificamente, $pccw(e)$ e $nccw(e)$ representam as arestas que precedem e sucedem e na face $fccw(e)$ (sentido anti-horário).

  Analogamente, $pcw(e)$ e $ncw(e)$ representam as arestas que precedem e sucedem e na face $fcw(e)$.

Arestas aladas



Note que cada registro dessa ED ocupa espaço constante.

Embrulho para presente

Embrulho para presente 2D: Jarvis.

Para dimensões arbitrárias: Chand & Kapur.

Embrulho para presente

Embrulho para presente 2D: Jarvis.

Para dimensões arbitrárias: Chand & Kapur.

Consumo de tempo para calcular $\mathrm{conv}(P)$:
$\mathrm{O}(nh)$, onde $n = |P|$ e h é o número de arestas de $\mathrm{conv}(P)$.

Embrulho para presente

Embrulho para presente 2D: Jarvis.

Para dimensões arbitrárias: Chand & Kapur.

Consumo de tempo para calcular $\mathrm{conv}(P)$:
$\mathrm{O}(nh)$, onde $n = |P|$ e h é o número de arestas de $\mathrm{conv}(P)$.

Suponha que $|P| \geq 4$.

Hipótese simplificadora: pontos de P estão em posição geral,
ou seja, não existem quatro pontos em P que sejam coplanares.

Embrulho para presente

Embrulho para presente 2D: Jarvis.

Para dimensões arbitrárias: Chand & Kapur.

Consumo de tempo para calcular $\mathrm{conv}(P)$:
$O(nh)$, onde $n = |P|$ e h é o número de arestas de $\mathrm{conv}(P)$.

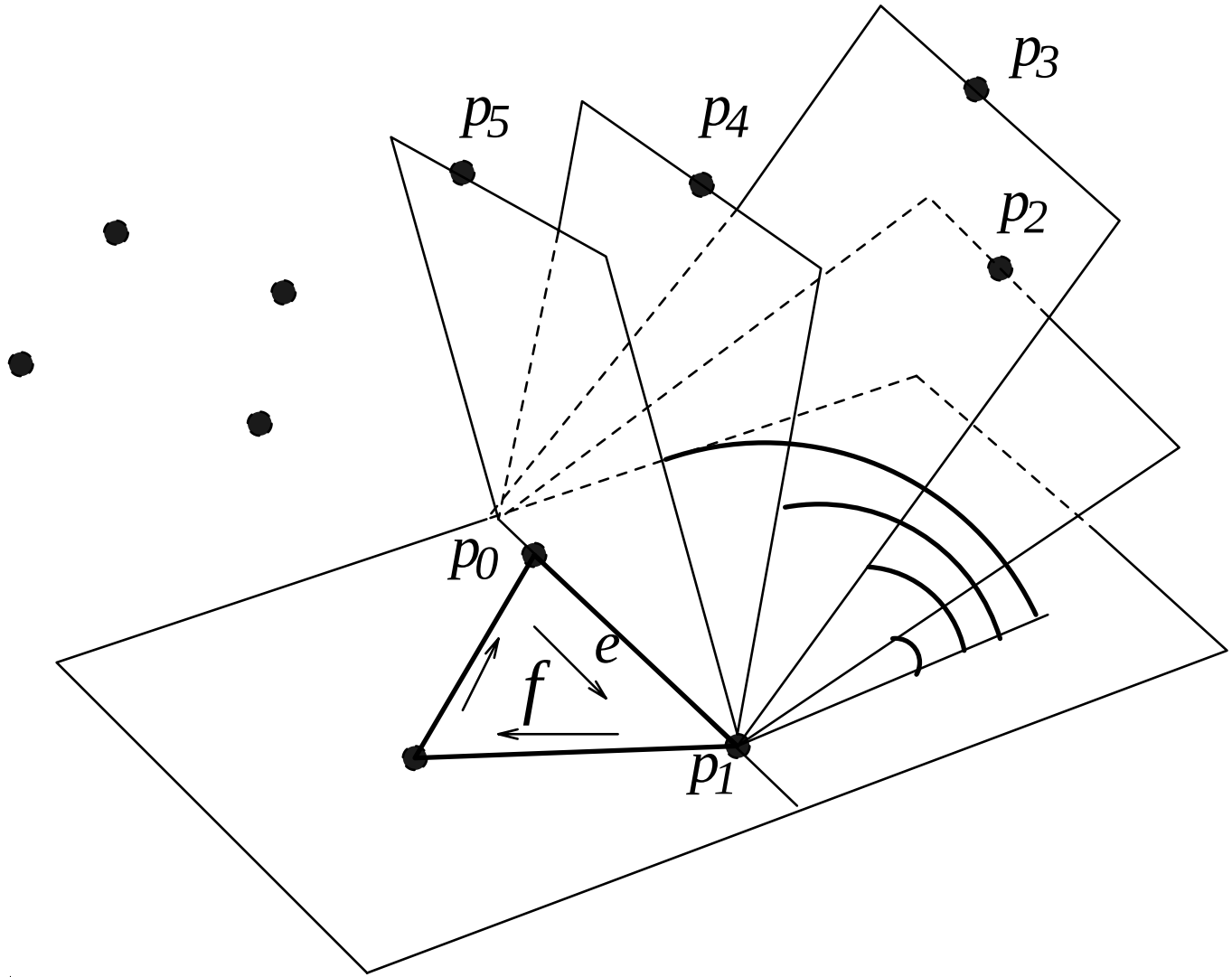
Suponha que $|P| \geq 4$.

Hipótese simplificadora: pontos de P estão em posição geral,
ou seja, não existem quatro pontos em P que sejam coplanares.

Consequência: $\mathrm{conv}(P)$ é simplicial (faces triangulares).

Embrulho para presente

Ilustração de um passo do algoritmo Embrulho3D.



Embrulho para presente

O algoritmo é iterativo.

Embrulho para presente

O algoritmo é iterativo.

Cada iteração começa com um conjunto de faces do fecho já determinadas, e um conjunto de arestas a serem processadas: arestas que estão em exatamente uma das faces já determinadas.

Embrulho para presente

O algoritmo é iterativo.

Cada iteração começa com um conjunto de faces do fecho já determinadas, e um conjunto de arestas a serem processadas: arestas que estão em exatamente uma das faces já determinadas.

A cada iteração, o algoritmo toma uma destas arestas, e determina a segunda face do fecho que a contém, incluindo no conjunto de arestas a serem processadas algumas novas arestas.

Inicialização do algoritmo Embrulho

Encontre um ponto extremo p_0:

Tome um ponto com coordenada Z menor possível.

Inicialização do algoritmo Embrulho

Encontre um ponto extremo p_0:
Tome um ponto com coordenada Z menor possível.

Encontre uma face contendo p_0:
Tome a reta por p_0 paralela ao eixo das abscissas
e o semi-plano horizontal π contendo p_0 e
orientado positivamente na direção do eixo y.

Inicialização do algoritmo Embrulho

Encontre um ponto extremo p_0:
Tome um ponto com coordenada Z menor possível.

Encontre uma face contendo p_0:
Tome a reta por p_0 paralela ao eixo das abscissas
e o semi-plano horizontal π contendo p_0 e
orientado positivamente na direção do eixo y.
Gire π no sentido de Y para Z
até encontrar outro ponto extremo p_1 da coleção.

Inicialização do algoritmo Embrulho

Encontre um ponto extremo p_0:
Tome um ponto com coordenada Z menor possível.

Encontre uma face contendo p_0:
Tome a reta por p_0 paralela ao eixo das abscissas
e o semi-plano horizontal π contendo p_0 e
orientado positivamente na direção do eixo y.
Gire π no sentido de Y para Z
até encontrar outro ponto extremo p_1 da coleção.

Encontre uma face (triangular) contendo p_0p_1:
Gire o novo plano π em torno da aresta p_0p_1
até encontrar um outro ponto extremo p_2 da coleção.

Embrulho para presente

Embrulho3D(P, n)

1 CrieFila(Q) ▷ fila com as faces encontradas
2 CrieWE(T) ▷ ED winged edges para $\mathrm{conv}(S)$
3 $f \leftarrow$ FaceInicial(P, n)
4 InsiraFila(Q, f)
5 InsiraWE(T, f)

Embrulho para presente

```
Embrulho3D(P, n)
 1  CrieFila(Q)      ▷ fila com as faces encontradas
 2  CrieWE(T)        ▷ ED winged edges para conv(S)
 3  f ← FaceInicial(P, n)
 4  InsiraFila(Q, f)
 5  InsiraWE(T, f)
 6  enquanto não FilaVazia(Q) faça
 7      f ← RemovaFila(Q)
 8      para cada aresta livre e de f faça
 9          f' ← FaceAdjacente(f, e)
```

Embrulho para presente

```
Embrulho3D(P, n)
 1  CrieFila(Q)      ▷ fila com as faces encontradas
 2  CrieWE(T)        ▷ ED winged edges para conv(S)
 3  f ← FaceInicial(P, n)
 4  InsiraFila(Q, f)
 5  InsiraWE(T, f)
 6  enquanto não FilaVazia(Q) faça
 7      f ← RemovaFila(Q)
 8      para cada aresta livre e de f faça
 9          f' ← FaceAdjacente(f, e)
10          InsiraFila(Q, f')
11          InsiraWE(T, f)
12  devolva T
```

Embrulho para presente

```
Embrulho3D(P, n)
 1  CrieFila(Q)      ▷ fila com as faces encontradas
 2  CrieWE(T)        ▷ ED winged edges para conv(S)
 3  f ← FaceInicial(P, n)
 4  InsiraFila(Q, f)
 5  InsiraWE(T, f)
 6  enquanto não FilaVazia(Q) faça
 7      f ← RemovaFila(Q)
 8      para cada aresta livre e de f faça
 9          f' ← FaceAdjacente(f, e)
10          InsiraFila(Q, f')
11          InsiraWE(T, f)
12  devolva T
```

Consumo de tempo:

$O(hn)$, onde h é o número de arestas do fecho convexo.

Algoritmo incremental

Incremental(P, n)

1 $P_3 \leftarrow \text{conv}(\{p_0, p_1, p_2, p_3\})$

Algoritmo incremental

Incremental(P, n)

1 $P_3 \leftarrow \text{conv}(\{p_0, p_1, p_2, p_3\})$
2 para $k \leftarrow 4$ até $n - 1$ faça
3 $\quad P_k \leftarrow \text{conv}(P_{k-1} \cup \{p_k\})$

Algoritmo incremental

Incremental(P, n)

1 $P_3 \leftarrow \text{conv}(\{p_0, p_1, p_2, p_3\})$
2 para $k \leftarrow 4$ até $n - 1$ faça
3 $\quad P_k \leftarrow \text{conv}(P_{k-1} \cup \{p_k\})$
4 devolva P_{n-1}

Algoritmo incremental

Incremental(P, n)

1 $P_3 \leftarrow \mathrm{conv}(\{p_0, p_1, p_2, p_3\})$
2 para $k \leftarrow 4$ até $n-1$ faça
3 $\quad P_k \leftarrow \mathrm{conv}(P_{k-1} \cup \{p_k\})$
4 devolva P_{n-1}

Linha 3: dois casos a serem tratados.

- $p_k \in P_{k-1}$
  Esta decisão pode ser feita em $\mathrm{O}(n)$, usando a rotina Volume6.

Algoritmo incremental

Incremental(P, n)

1 $P_3 \leftarrow \mathrm{conv}(\{p_0, p_1, p_2, p_3\})$
2 para $k \leftarrow 4$ até $n-1$ faça
3 $\quad P_k \leftarrow \mathrm{conv}(P_{k-1} \cup \{p_k\})$
4 devolva P_{n-1}

Linha 3: dois casos a serem tratados.

- $p_k \in P_{k-1}$
  Esta decisão pode ser feita em $\mathrm{O}(n)$,
  usando a rotina Volume6.
- $p_k \notin P_{k-1}$
  Generalizaremos a ideia da versão 2D deste algoritmo.

Se $p_k \notin P_{k-1}$...

No caso 2D, encontrávamos as duas retas que passavam pelo ponto p_k e que são tangentes ao polígono P_{k-1}.

Se $p_k \notin P_{k-1}$...

No caso 2D, encontrávamos as duas retas que passavam pelo ponto p_k e que são tangentes ao polígono P_{k-1}.

No caso 3D, encontramos planos tangentes em vez de retas tangentes.

Se $p_k \notin P_{k-1}$...

No caso 2D, encontrávamos as duas retas que passavam pelo ponto p_k e que são tangentes ao polígono P_{k-1}.

No caso 3D, encontramos planos tangentes em vez de retas tangentes.

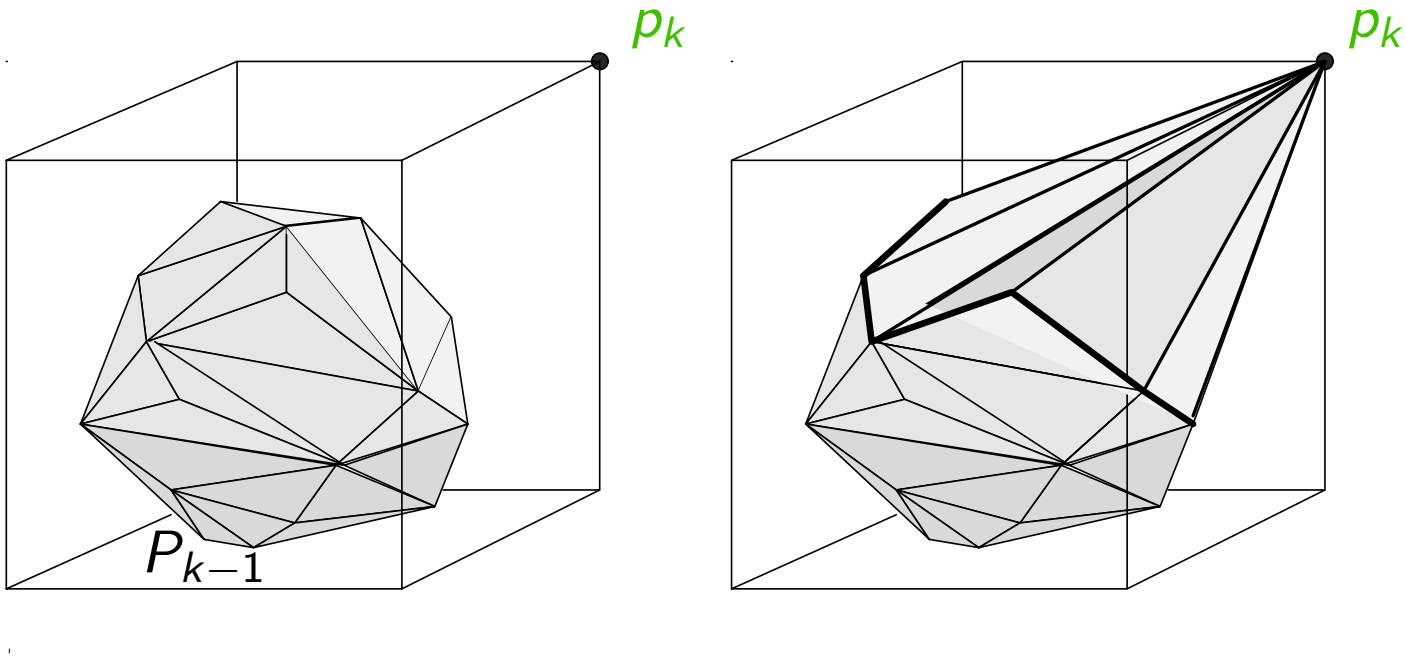
Estes planos tangentes determinam um cone que tem como faces triângulos e tem como bico o ponto p_k.

Se $p_k \notin P_{k-1}$...

No caso 2D, encontrávamos as duas retas que passavam pelo ponto p_k e que são tangentes ao polígono P_{k-1}.

No caso 3D, encontramos planos tangentes em vez de retas tangentes.

Estes planos tangentes determinam um cone que tem como faces triângulos e tem como bico o ponto p_k.



Como encontrar tais faces?

Faces de P_{k-1} a serem descartadas:
aquelas que são visíveis da posição que está o ponto p_k.

Como encontrar tais faces?

Faces de P_{k-1} a serem descartadas:
aquelas que são visíveis da posição que está o ponto p_k.

Faces visíveis: têm orientação positiva olhando de p_k.

Face $f = \triangle(a, b, c)$ é visível de p se
o sinal do volume do tetraedro formado por a, b, c e p é positivo.

Como encontrar tais faces?

Faces de P_{k-1} a serem descartadas:
aquelas que são visíveis da posição que está o ponto p_k.

Faces visíveis: têm orientação positiva olhando de p_k.

Face $f = \triangle(a, b, c)$ é visível de p se
o sinal do volume do tetraedro formado por a, b, c e p é positivo.

Arestas na fronteira das faces visíveis:
formarão as faces triangulares do cone com o ponto p_k.

Como encontrar tais faces?

Faces de P_{k-1} a serem descartadas:
aquelas que são visíveis da posição que está o ponto p_k.

Faces visíveis: têm orientação positiva olhando de p_k.

Face $f = \triangle(a, b, c)$ é visível de p se
o sinal do volume do tetraedro formado por a, b, c e p é positivo.

Arestas na fronteira das faces visíveis:
formarão as faces triangulares do cone com o ponto p_k.

Suponha que e é uma aresta de P_{k-1} tal que
o plano contendo e e o ponto p_k é tangente a P_{k-1}.

Como encontrar tais faces?

Faces de P_{k-1} a serem descartadas:
aquelas que são visíveis da posição que está o ponto p_k.

Faces visíveis: têm orientação positiva olhando de p_k.

Face $f = \triangle(a, b, c)$ é visível de p se
o sinal do volume do tetraedro formado por a, b, c e p é positivo.

Arestas na fronteira das faces visíveis:
formarão as faces triangulares do cone com o ponto p_k.

Suponha que e é uma aresta de P_{k-1} tal que
o plano contendo e e o ponto p_k é tangente a P_{k-1}.

Cada aresta é compartilhada por exatamente duas faces.
Uma das faces incidentes a e é visível a partir de p_k e a outra não.
Logo, e está na fronteira da região visível de p_k.

Algoritmo incremental

Incremental3D(P, n)

1 $P_3 \leftarrow$ Tetraedro(p_0, p_1, p_2, p_3)

Algoritmo incremental

Incremental3D(P, n)

1 $P_3 \leftarrow$ Tetraedro(p_0, p_1, p_2, p_3)
2 para $k \leftarrow 4$ até $n - 1$ faça
3 para cada face f de P_{k-1} faça
4 $v \leftarrow$ Volume6(f, p_k)
5 se $v > 0$ então marque f como visível de p_k

Algoritmo incremental

Incremental3D(P, n)

1 $P_3 \leftarrow$ Tetraedro(p_0, p_1, p_2, p_3)
2 para $k \leftarrow 4$ até $n - 1$ faça
3 para cada face f de P_{k-1} faça
4 $v \leftarrow$ Volume6(f, p_k)
5 se $v > 0$ então marque f como visível de p_k
6 se nenhuma face é visível de p_k
7 então $P_k \leftarrow P_{k-1}$

Algoritmo incremental

Incremental3D(P, n)

1 $P_3 \leftarrow$ Tetraedro(p_0, p_1, p_2, p_3)
2 para $k \leftarrow 4$ até $n-1$ faça
3 para cada face f de P_{k-1} faça
4 $v \leftarrow$ Volume6(f, p_k)
5 se $v > 0$ então marque f como visível de p_k
6 se nenhuma face é visível de p_k
7 então $P_k \leftarrow P_{k-1}$
8 senão para cada e na fronteira das faces visíveis
9 construa a face determinada por e e p_k

Algoritmo incremental

```
Incremental3D(P, n)
 1  P_3 ← Tetraedro(p_0, p_1, p_2, p_3)
 2  para k ← 4 até n − 1 faça
 3     para cada face f de P_{k−1} faça
 4        v ← Volume6(f, p_k)
 5        se v > 0 então marque f como visível de p_k
 6     se nenhuma face é visível de p_k
 7        então P_k ← P_{k−1}
 8        senão para cada e na fronteira das faces visíveis
 9                 construa a face determinada por e e p_k
10              para cada face visível f
11                 remova f de P_{k−1}
12              faça os acertos finais obtendo P_k
```

Algoritmo incremental

```
Incremental3D(P, n)
 1  P_3 ← Tetraedro(p_0, p_1, p_2, p_3)
 2  para k ← 4 até n − 1 faça
 3     para cada face f de P_{k−1} faça
 4        v ← Volume6(f, p_k)
 5        se v > 0 então marque f como visível de p_k
 6     se nenhuma face é visível de p_k
 7        então P_k ← P_{k−1}
 8        senão para cada e na fronteira das faces visíveis
 9                 construa a face determinada por e e p_k
10              para cada face visível f
11                 remova f de P_{k−1}
12              faça os acertos finais obtendo P_k
13  devolva P_{n−1}
```

Algoritmo incremental

```
Incremental3D(P, n)
 1  P_3 ← Tetraedro(p_0, p_1, p_2, p_3)
 2  para k ← 4 até n − 1 faça
 3     para cada face f de P_{k−1} faça
 4        v ← Volume6(f, p_k)
 5        se v > 0 então marque f como visível de p_k
 6     se nenhuma face é visível de p_k
 7        então P_k ← P_{k−1}
 8        senão para cada e na fronteira das faces visíveis
 9                 construa a face determinada por e e p_k
10              para cada face visível f
11                 remova f de P_{k−1}
12              faça os acertos finais obtendo P_k
13  devolva P_{n−1}
```

Consumo de tempo: Pela fórmula de Euler, é $\mathrm{O}(n^2)$.

Algoritmo incremental probabilístico

A versão probabilística do algoritmo escolhe uma permutação uniformemente dos n pontos dados, e os processa nesta ordem.

Algoritmo incremental probabilístico

A versão probabilística do algoritmo escolhe uma permutação uniformemente dos n pontos dados, e os processa nesta ordem.

Essa versão consome tempo $\mathrm{O}(n \lg n)$.

Isso está provado na seção 11.3 do livro de de Berg et al.

Algoritmo incremental probabilístico

A versão probabilística do algoritmo escolhe uma permutação uniformemente dos n pontos dados, e os processa nesta ordem.

Essa versão consome tempo $\mathrm{O}(n \lg n)$.

Isso está provado na seção 11.3 do livro de de Berg et al.

Lema: O número esperado de faces criadas por esta variante do Incremental3D é no máximo $6n - 20$.

Algoritmo incremental probabilístico

A versão probabilística do algoritmo escolhe uma permutação uniformemente dos n pontos dados, e os processa nesta ordem.

Essa versão consome tempo $\mathrm{O}(n \lg n)$.

Isso está provado na seção 11.3 do livro de de Berg et al.

Lema: O número esperado de faces criadas por esta variante do Incremental3D é no máximo $6n - 20$.

Esboço da prova: O 1-esqueleto é um grafo planar.

Algoritmo incremental probabilístico

A versão probabilística do algoritmo escolhe uma permutação uniformemente dos n pontos dados, e os processa nesta ordem.

Essa versão consome tempo $\mathrm{O}(n \lg n)$.

Isso está provado na seção 11.3 do livro de de Berg et al.

Lema: O número esperado de faces criadas por esta variante do Incremental3D é no máximo $6n - 20$.

Esboço da prova: O 1-esqueleto é um grafo planar.
O vértice p_k é um dos vértices de P_k com probabilidade uniforme.
O número de faces criadas para inserir p_k é o grau de p_k em P_k.

Algoritmo incremental probabilístico

A versão probabilística do algoritmo escolhe uma permutação uniformemente dos n pontos dados, e os processa nesta ordem.

Essa versão consome tempo $\mathrm{O}(n \lg n)$.

Isso está provado na seção 11.3 do livro de de Berg et al.

Lema: O número esperado de faces criadas por esta variante do Incremental3D é no máximo $6n - 20$.

Esboço da prova: O 1-esqueleto é um grafo planar.
O vértice p_k é um dos vértices de P_k com probabilidade uniforme.
O número de faces criadas para inserir p_k é o grau de p_k em P_k.
O grau esperado de p_k é o grau médio do 1-esqueleto de P_k,

Algoritmo incremental probabilístico

A versão probabilística do algoritmo escolhe uma permutação uniformemente dos n pontos dados, e os processa nesta ordem.

Essa versão consome tempo $\mathrm{O}(n \lg n)$.

Isso está provado na seção 11.3 do livro de de Berg et al.

Lema: O número esperado de faces criadas por esta variante do Incremental3D é no máximo $6n - 20$.

Esboço da prova: O 1-esqueleto é um grafo planar.
O vértice p_k é um dos vértices de P_k com probabilidade uniforme.
O número de faces criadas para inserir p_k é o grau de p_k em P_k.
O grau esperado de p_k é o grau médio do 1-esqueleto de P_k, que é $\leq 2(3k - 6)/k < 6$.

Algoritmo incremental probabilístico

A versão probabilística do algoritmo escolhe uma permutação uniformemente dos n pontos dados, e os processa nesta ordem.

Essa versão consome tempo $\mathrm{O}(n \lg n)$.

Isso está provado na seção 11.3 do livro de de Berg et al.

Lema: O número esperado de faces criadas por esta variante do Incremental3D é no máximo $6n - 20$.

Observe que o lema acima garante que o custo total para construir os fechos $P_1, \ldots, P_n$ poderia ser $\mathrm{O}(n)$ em princípio.

Algoritmo incremental probabilístico

A versão probabilística do algoritmo escolhe uma permutação uniformemente dos n pontos dados, e os processa nesta ordem.

Essa versão consome tempo $\mathrm{O}(n \lg n)$.

Isso está provado na seção 11.3 do livro de de Berg et al.

Lema: O número esperado de faces criadas por esta variante do Incremental3D é no máximo $6n - 20$.

Observe que o lema acima garante que o custo total para construir os fechos $P_1, \ldots, P_n$ poderia ser $\mathrm{O}(n)$ em princípio.

No entanto, para isso, temos que ser capazes de encontrar as faces visíveis sem testar a visibilidade face a face.

Algoritmo incremental probabilístico

Para isso, o algoritmo carrega um grafo bipartido G

Algoritmo incremental probabilístico

Para isso, o algoritmo carrega um grafo bipartido G
onde um lado da bipartição são os vértices $p_k, \ldots, p_n$
e do outro lado da bipartição as faces de P_{k-1}.

Algoritmo incremental probabilístico

Para isso, o algoritmo carrega um grafo bipartido G onde um lado da bipartição são os vértices $p_k, \ldots, p_n$ e do outro lado da bipartição as faces de P_{k-1}.

Um vértice p_t é adjacente a uma face f se enxerga f.

Algoritmo incremental probabilístico

Para isso, o algoritmo carrega um grafo bipartido G
onde um lado da bipartição são os vértices $p_k, \ldots, p_n$
e do outro lado da bipartição as faces de P_{k-1}.

Um vértice p_t é adjacente a uma face f se enxerga f.

$F_{conflito}(p)$: conjunto das faces vizinhas em G ao vértice p
$P_{conflito}(f)$: conjunto dos vértices vizinhos em G à face f

Algoritmo incremental probabilístico

Para isso, o algoritmo carrega um grafo bipartido G
onde um lado da bipartição são os vértices $p_k, \ldots, p_n$
e do outro lado da bipartição as faces de P_{k-1}.

Um vértice p_t é adjacente a uma face f se enxerga f.

$F_{conflito}(p)$: conjunto das faces vizinhas em G ao vértice p
$P_{conflito}(f)$: conjunto dos vértices vizinhos em G à face f

Em cada iteração,
$F_{conflito}(p_k)$ são as faces a serem removidas de P_{k-1}.

Remova as faces de $F_{conflito}(p_k)$ da ED das arestas aladas.

Algoritmo incremental probabilístico

Enquanto remove as faces de $F_{conflito}(p_k)$,
determine o conjunto $\mathcal{L}$ das arestas na borda da região visível.

Algoritmo incremental probabilístico

Enquanto remove as faces de $F_{conflito}(p_k)$,
determine o conjunto $\mathcal{L}$ das arestas na borda da região visível.

Para cada aresta e do conjunto $\mathcal{L}$,
acrescente uma nova face com e e p_k na ED das arestas aladas,
para obter a representação de P_k.

Algoritmo incremental probabilístico

Enquanto remove as faces de $F_{conflito}(p_k)$,
determine o conjunto $\mathcal{L}$ das arestas na borda da região visível.

Para cada aresta e do conjunto $\mathcal{L}$,
acrescente uma nova face com e e p_k na ED das arestas aladas,
para obter a representação de P_k.

O tempo para isso é proporcional à soma do tamanho
dos dois conjuntos, e isso, pelo lema anterior, é $\mathrm{O}(n)$ no total.

Algoritmo incremental probabilístico

Enquanto remove as faces de $F_{conflito}(p_k)$,
determine o conjunto $\mathcal{L}$ das arestas na borda da região visível.

Para cada aresta e do conjunto $\mathcal{L}$,
acrescente uma nova face com e e p_k na ED das arestas aladas,
para obter a representação de P_k.

O tempo para isso é proporcional à soma do tamanho
dos dois conjuntos, e isso, pelo lema anterior, é $\mathrm{O}(n)$ no total.

Mas... adicionalmente precisamos atualizar G.

Isso envolve:

- remover p_k;
- remover as faces de $F_{conflito}(p_k)$;
- inserir as faces novas, uma para cada $e \in \mathcal{L}$;
- determinar o conjunto $P_{conflito}(f)$ para cada face nova f.

Algoritmo incremental probabilístico

Como determinar o conjunto $P_{conflito}(f)$ para cada f?

Esta é a parte do algoritmo que consome $\mathrm{O}(n \lg n)$ no total.

Algoritmo incremental probabilístico

Como determinar o conjunto $P_{conflito}(f)$ para cada f?

Esta é a parte do algoritmo que consome $\mathrm{O}(n \lg n)$ no total.

Para mostrar isso, primeiro observe que, se $e \in \mathcal{L}$
e f é a nova aresta formada por e e p_k, então

$$P_{conflito}(f) \subseteq P_{conflito}(f_1) \cup P_{conflito}(f_2)$$

onde f_1 e f_2 são as faces de P_{k-1} que compartilham e.

Algoritmo incremental probabilístico

Como determinar o conjunto $P_{conflito}(f)$ para cada f?

Esta é a parte do algoritmo que consome $\mathrm{O}(n \lg n)$ no total.

Para mostrar isso, primeiro observe que, se $e \in \mathcal{L}$
e f é a nova aresta formada por e e p_k, então

$$P_{conflito}(f) \subseteq P_{conflito}(f_1) \cup P_{conflito}(f_2)$$

onde f_1 e f_2 são as faces de P_{k-1} que compartilham e.

Seja $P(e) = P_{conflito}(f_1) \cup P_{conflito}(f_2)$.

Algoritmo incremental probabilístico

Seja $P(e) = P_{conflito}(f_1) \cup P_{conflito}(f_2)$.

Usando a rotina Volume6, podemos calcular
o conjunto $P_{conflito}(f)$ para cada face nova f em tempo $\mathrm{O}(|P(e)|)$.

Algoritmo incremental probabilístico

Seja $P(e) = P_{conflito}(f_1) \cup P_{conflito}(f_2)$.

Usando a rotina Volume6, podemos calcular
o conjunto $P_{conflito}(f)$ para cada face nova f em tempo $\mathrm{O}(|P(e)|)$.

No livro de de Berg et al., se encontra a prova de que

$$\mathrm{E}[\textstyle\sum_e |P(e)|] = \mathrm{O}(n \lg n)$$

onde a soma é sobre todas as arestas que
fizeram parte de $\mathcal{L}$ em alguma iteração do algoritmo. Então...

Algoritmo incremental probabilístico

Seja $P(e) = P_{conflito}(f_1) \cup P_{conflito}(f_2)$.

Usando a rotina Volume6, podemos calcular
o conjunto $P_{conflito}(f)$ para cada face nova f em tempo $\mathrm{O}(|P(e)|)$.

No livro de de Berg et al., se encontra a prova de que

$$\mathrm{E}[\textstyle\sum_e |P(e)|] = \mathrm{O}(n \lg n)$$

onde a soma é sobre todas as arestas que
fizeram parte de $\mathcal{L}$ em alguma iteração do algoritmo. Então...

Lema:
O consumo de tempo esperado desta variante do Incremental3D
é $\mathrm{O}(n \lg n)$ para calcular o fecho convexo de n pontos do $\mathbb{R}^3$.