Geometria Computacional

Aula 21

ED e algoritmos para fecho convexo 3D

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

Arestas aladas

Proposta por Baumgart em 1975,
é a mais popular para representar a superfície de um poliedro.

Seu foco são as arestas.

Lembra a DCEL, ED que vimos para mapas planares.

Arestas aladas

Proposta por Baumgart em 1975,
é a mais popular para representar a superfície de um poliedro.

Seu foco são as arestas.

Lembra a DCEL, ED que vimos para mapas planares.

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 (olhando de fora do poliedro) de $e = v_1(e)v_2(e)$ e a face $fcw(e)$ é a direita.

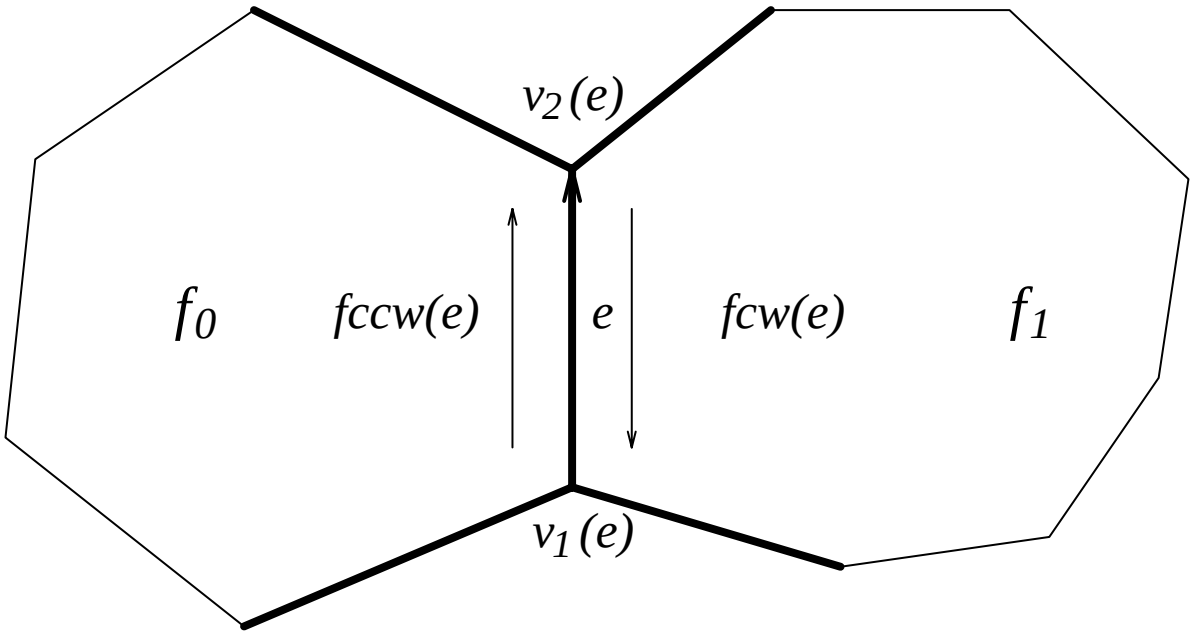
  A aresta e induz orientação anti-horária (*counterclockwise*) em $fccw(e)$ e orientação horária (*clockwise*) em $fcw(e)$.

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 (olhando de fora do poliedro) de $e = v_1(e)v_2(e)$ e a face $fcw(e)$ é a direita.

  A aresta e induz orientação anti-horária (*counterclockwise*) em $fccw(e)$ e orientação horária (*clockwise*) em $fcw(e)$.



Arestas aladas

Cada aresta e tem oito apontadores:

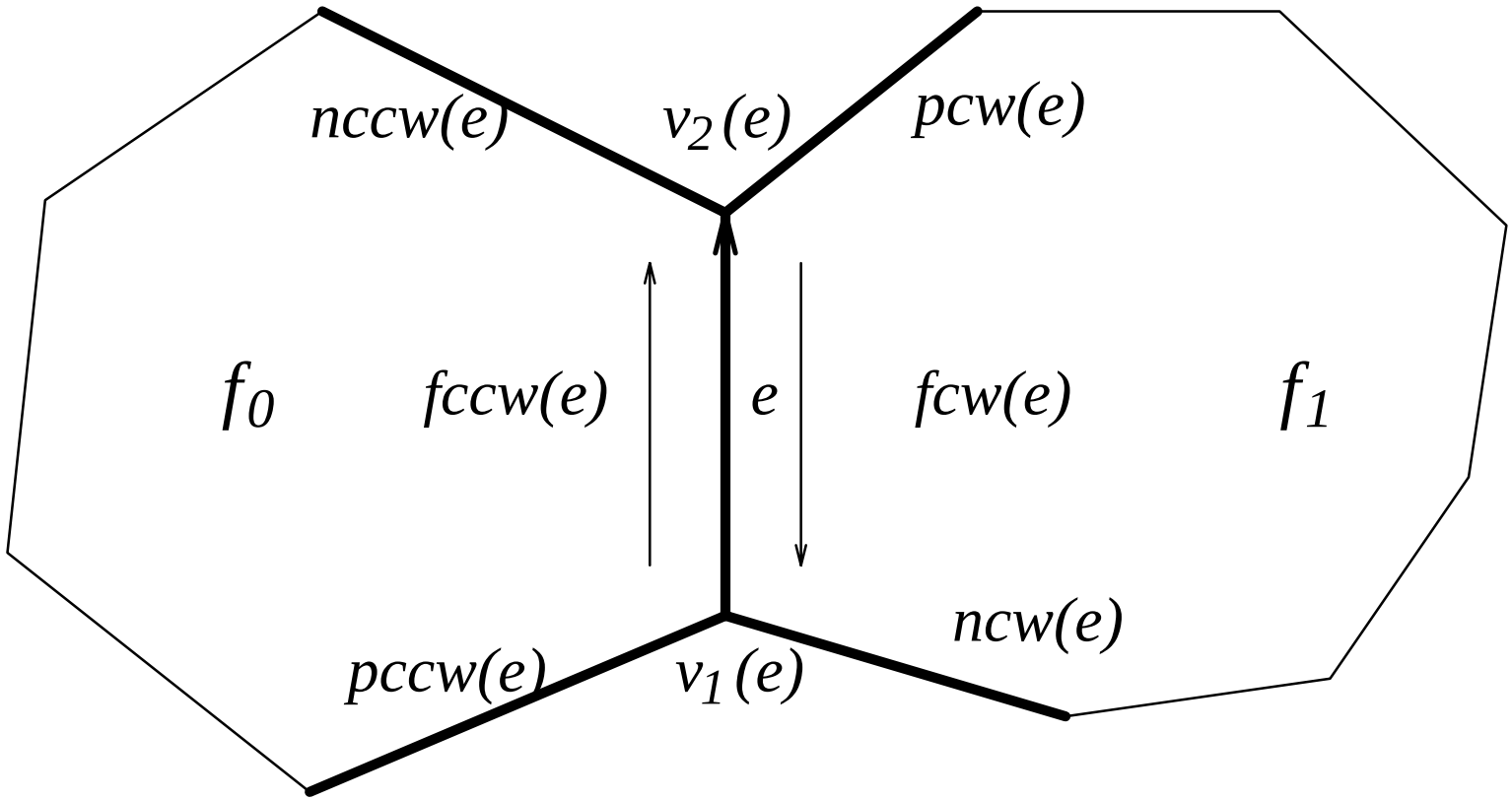
- Dois apontadores para os extremos $v_1(e)$ e $v_2(e)$ de e.
  A ordem destes vértices fornece uma orientação para e.
- Apontadores $fccw(e)$ e $fcw(e)$ para as duas faces incidentes a e. A face $fccw(e)$ é a esquerda de $e = v_1(e)v_2(e)$ e a face $fcw(e)$ é a direita.
- Quatro apontadores para as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

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

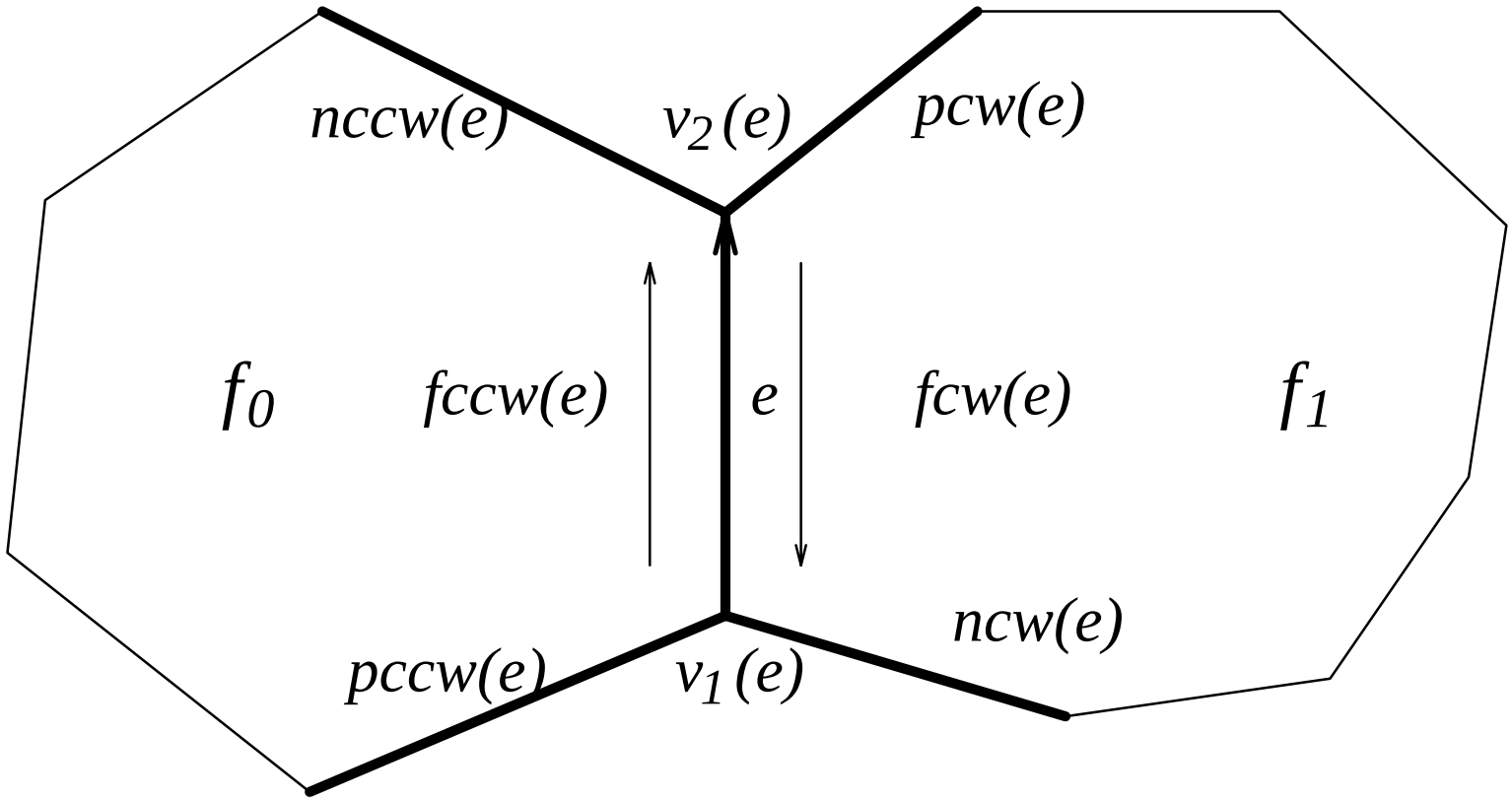
Arestas aladas

Vista de fora do poliedro.



Arestas aladas

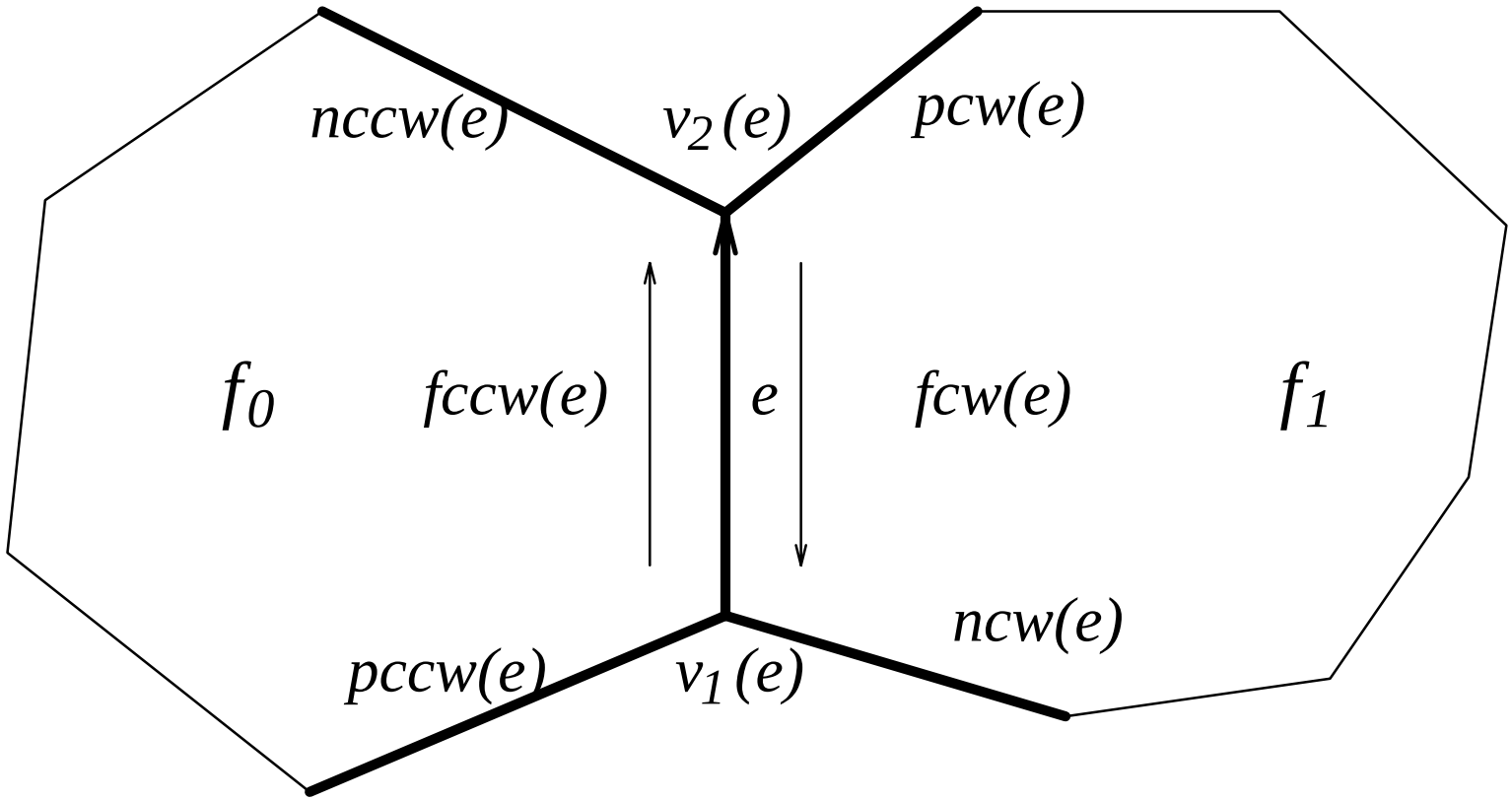
Vista de fora do poliedro.



Note que cada registro dessa ED ocupa espaço constante.

Arestas aladas

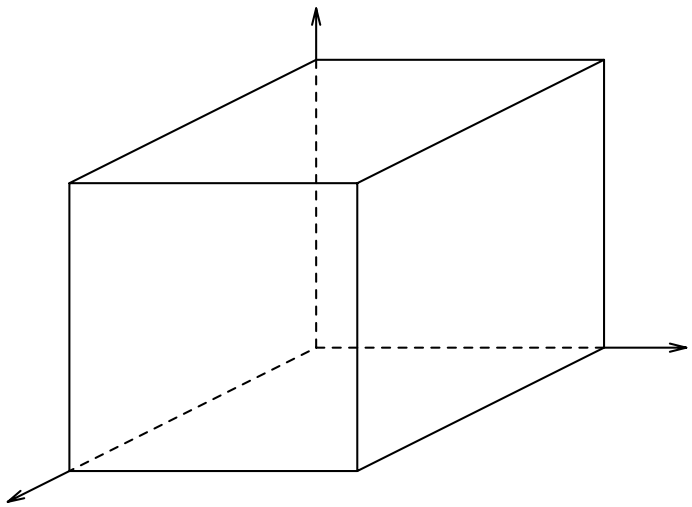
Vista de fora do poliedro.

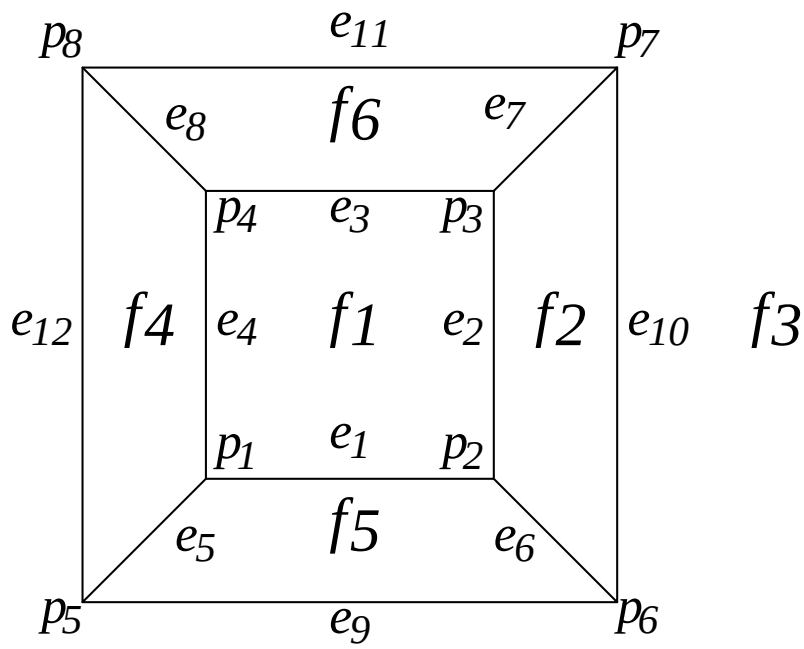


Note que cada registro dessa ED ocupa espaço constante.

Vejamos um exemplo completo... o cubo!

O cubo





Um cubo e seu 1-esqueleto (grafo planar).

Arestas aladas para o cubo

Lista de vértices:

vértice	(x, y, z)	av
p_1	$(1, 0, 0)$	e_1
p_2	$(1, 1, 0)$	e_2
p_3	$(1, 1, 1)$	e_3
p_4	$(1, 0, 1)$	e_4
p_5	$(0, 0, 0)$	e_9
p_6	$(0, 1, 0)$	e_{10}
p_7	$(0, 1, 1)$	e_{11}
p_8	$(0, 0, 1)$	e_{12}

Arestas aladas para o cubo

Lista de faces:

face	af
f_1	e_1
f_2	e_2
f_3	e_{10}
f_4	e_{12}
f_5	e_1
f_6	e_8

Arestas aladas para o cubo

aresta	v_1	v_2	fcw	$fccw$	pcw	ncw	$pccw$	$nccw$
e_1	p_1	p_2	f_1	f_5	e_2	e_4	e_5	e_6
e_2	p_2	p_3	f_1	f_2	e_3	e_1	e_6	e_7
e_3	p_3	p_4	f_1	f_6	e_4	e_2	e_7	e_8
e_4	p_4	p_1	f_1	f_4	e_1	e_3	e_8	e_5
e_5	p_1	p_5	f_5	f_4	e_9	e_1	e_4	e_{12}
e_6	p_2	p_6	f_2	f_5	e_{10}	e_2	e_1	e_9
e_7	p_3	p_7	f_6	f_2	e_{11}	e_3	e_2	e_{10}
e_8	p_4	p_8	f_4	f_6	e_{12}	e_4	e_3	e_{11}
e_9	p_5	p_6	f_5	f_3	e_6	e_5	e_{12}	e_{10}
e_{10}	p_6	p_7	f_2	f_3	e_7	e_6	e_9	e_{11}
e_{11}	p_7	p_8	f_6	f_3	e_8	e_7	e_{10}	e_{12}
e_{12}	p_8	p_5	f_4	f_3	e_5	e_8	e_{11}	e_9

Fecho convexo 3D

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

Fecho convexo 3D

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

Descreveremos quatro algoritmos:

- Embrulho para presente
- Incremental
- Incremental probabilístico
- Quickhull (brevemente)

Fecho convexo 3D

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

Descreveremos quatro algoritmos:

- Embrulho para presente
- Incremental
- Incremental probabilístico
- Quickhull (brevemente)

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

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 $\mathrm{conv}(P)$.

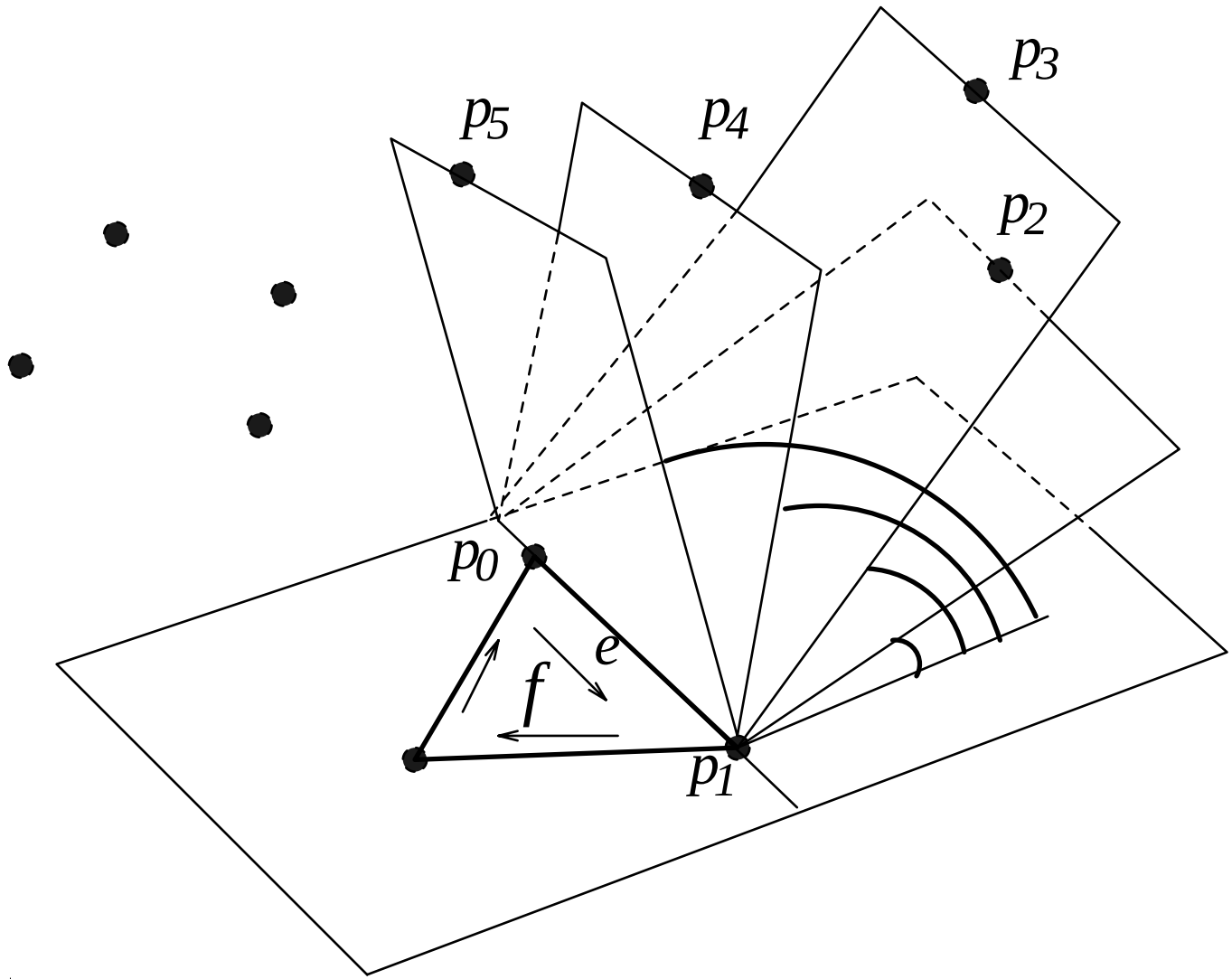
Suponha que $|P| \geq 4$.

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 X 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 X 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 X 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
```

FaceInicial(P, n) e FaceAdjacente(f, e) consomem $\mathrm{O}(n)$.
As demais rotinas consomem $\mathrm{O}(1)$.

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
```

InsiraWE(T, f):
algumas arestas de f já tinham sido descobertas, outras não.

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.

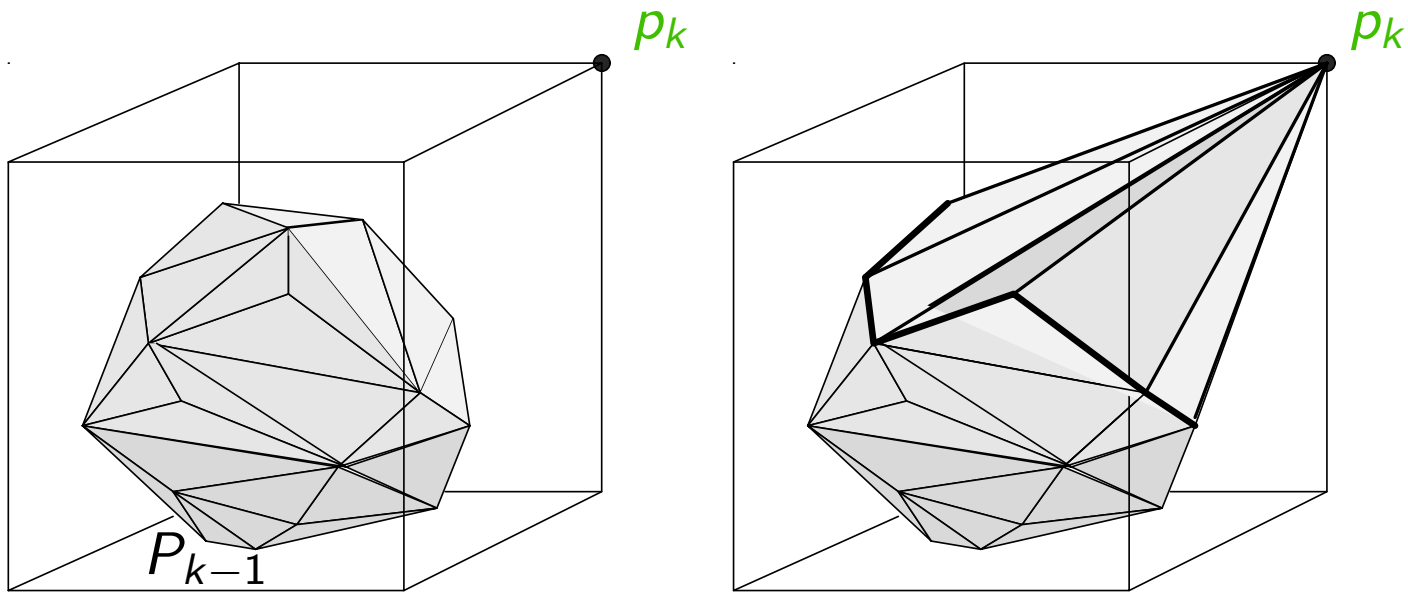
Estes planos tangentes determinam um cone que tem como faces triângulos e tem como bico o ponto p_k.

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 $\mathrm{O}(n^2)$.

Aula que vem

Versão aleatorizada do algoritmo incremental.

Breve descrição do QuickHull para 3D.