Geometria Computacional

Aula 22

Algoritmos para fecho convexo 3D

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

artigo sobre o QuickHull:

Barber, Dobkin, Huhdanpaa,
The Quickhull algorithm for convex hulls

`https://dl.acm.org/doi/pdf/10.1145/235815.235821`

Algoritmo incremental

Também conhecido como Beneath-Beyond algorithm, foi proposto pela primeira vez por Seidel e Kallay.

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

Também conhecido como Beneath-Beyond algorithm, foi proposto pela primeira vez por Seidel e Kallay.

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

Também conhecido como Beneath-Beyond algorithm, foi proposto pela primeira vez por Seidel e Kallay.

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.

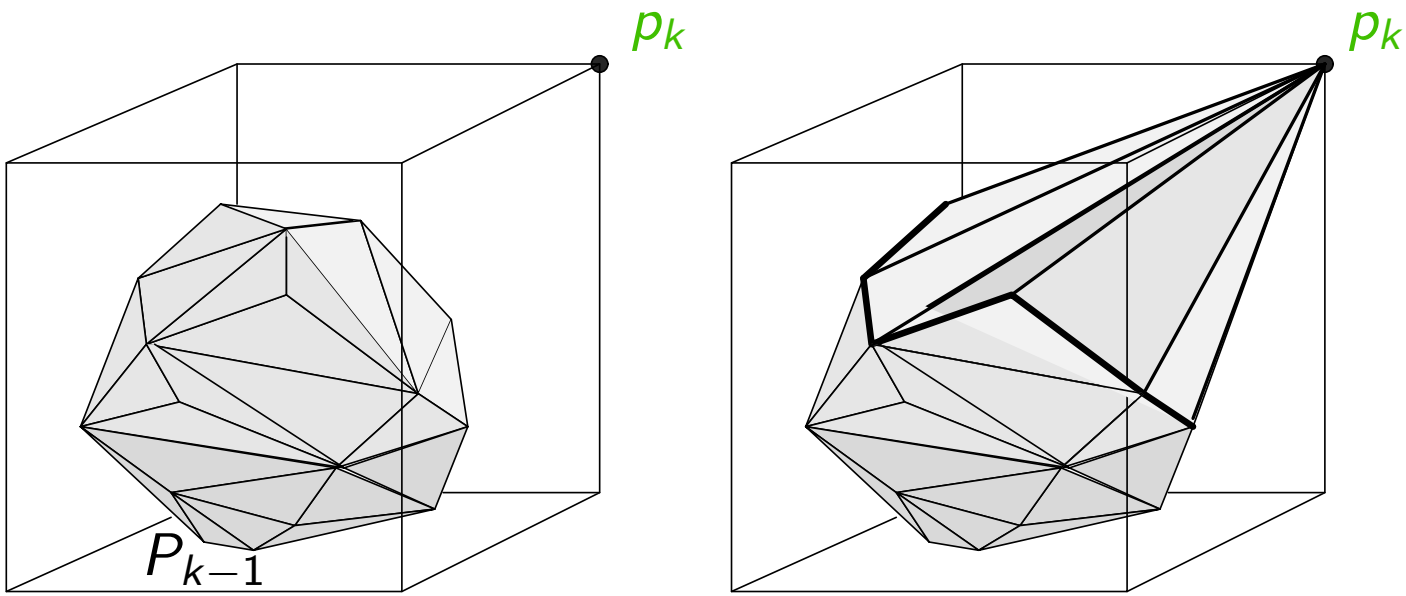
- $p_k \in P_{k-1}$
  Esta decisão pode ser feita em $\mathrm{O}(n)$, usando a rotina Volume6.
- $p_k \notin P_{k-1}$
  Generalizaremos a ideia da versão 2D deste algoritmo.

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}.

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 ← 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 foi proposta por Clarkson e Shor, e escolhe uniformemente uma permutação dos pontos $p_4, \ldots, p_{n-1}$, e os processa nesta ordem.

Incremental(P, n)

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

Algoritmo incremental probabilístico

A versão probabilística do algoritmo foi proposta por Clarkson e Shor, e escolhe uniformemente uma permutação dos pontos $p_4, \ldots, p_{n-1}$, e os processa nesta ordem.

Incremental(P, n)

1 $P_3 \leftarrow \text{conv}(\{p_0, p_1, p_2, p_3\})$
2 permuta aleatoriamente $p_4, \ldots, p_{n-1}$
3 para $k \leftarrow 4$ até $n-1$ faça
4 $\quad P_k \leftarrow \text{conv}(P_{k-1} \cup \{p_k\})$
5 devolva P_{n-1}

Algoritmo incremental probabilístico

A versão probabilística do algoritmo foi proposta por Clarkson e Shor, e escolhe uniformemente uma permutação dos pontos $p_4, \ldots, p_{n-1}$, e os processa nesta ordem.

Incremental(P, n)

1 $P_3 \leftarrow \text{conv}(\{p_0, p_1, p_2, p_3\})$
2 permuta aleatoriamente $p_4, \ldots, p_{n-1}$
3 para $k \leftarrow 4$ até $n-1$ faça
4 $\quad P_k \leftarrow \text{conv}(P_{k-1} \cup \{p_k\})$
5 devolva P_{n-1}

Usando uma ED extra, essa versão consome tempo $\mathrm{O}(n \lg n)$.

Algoritmo incremental probabilístico

A versão probabilística do algoritmo foi proposta por Clarkson e Shor, e escolhe uniformemente uma permutação dos pontos $p_4, \ldots, p_{n-1}$, e os processa nesta ordem.

Incremental(P, n)

1 $P_3 \leftarrow \text{conv}(\{p_0, p_1, p_2, p_3\})$
2 permuta aleatoriamente $p_4, \ldots, p_{n-1}$
3 para $k \leftarrow 4$ até $n-1$ faça
4 $\quad P_k \leftarrow \text{conv}(P_{k-1} \cup \{p_k\})$
5 devolva P_{n-1}

Usando uma ED extra, essa versão consome tempo $\mathrm{O}(n \lg n)$.

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

Análise do incremental probabilístico

Lema: O número esperado de faces criadas pelo Incremental3D probabilístico é no máximo $6n - 20$.

Análise do incremental probabilístico

Lema: O número esperado de faces criadas pelo Incremental3D probabilístico é no máximo $6n - 20$.

Prova: Suponha que p_k é um vértice de P_k.
O 1-esqueleto de P_k é um grafo planar.

Análise do incremental probabilístico

Lema: O número esperado de faces criadas pelo Incremental3D probabilístico é no máximo $6n - 20$.

Prova: Suponha que p_k é um vértice de P_k.
O 1-esqueleto de P_k é um grafo planar.

Excluindo p_0, p_1, p_2, p_3,
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.

Análise do incremental probabilístico

Lema: O número esperado de faces criadas pelo Incremental3D probabilístico é no máximo $6n - 20$.

Prova: Suponha que p_k é um vértice de P_k.
O 1-esqueleto de P_k é um grafo planar.

Excluindo p_0, p_1, p_2, p_3,
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,
excluindo p_0, p_1, p_2, p_3, que é

$$\leq \frac{2(3k-6)-12}{k-4} = \frac{6k-24}{k-4} = 6.$$

Análise do incremental probabilístico

Lema: O número esperado de faces criadas pelo Incremental3D probabilístico é no máximo $6n - 20$.

Prova: Suponha que p_k é um vértice de P_k.
O 1-esqueleto de P_k é um grafo planar.

Excluindo p_0, p_1, p_2, p_3,
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,
excluindo p_0, p_1, p_2, p_3, que é

$$\leq \frac{2(3k-6)-12}{k-4} = \frac{6k-24}{k-4} = 6.$$

Então o número esperado de faces criadas é no máximo $4+6(n-4)$.

Análise do incremental probabilístico

Lema: O número esperado de faces criadas pelo Incremental3D probabilístico é no máximo $6n - 20$.

Então o tempo para remover e acrescentar faces no total é $\mathrm{O}(n)$.

Análise do incremental probabilístico

Lema: O número esperado de faces criadas pelo Incremental3D probabilístico é no máximo $6n - 20$.

Então o tempo para remover e acrescentar faces no total é $\mathrm{O}(n)$.

Ou seja, o custo total para construir os fechos $P_3, \ldots, P_{n-1}$ poderia ser $\mathrm{O}(n)$ em princípio.

Análise do incremental probabilístico

Lema: O número esperado de faces criadas pelo Incremental3D probabilístico é no máximo $6n - 20$.

Então o tempo para remover e acrescentar faces no total é $\mathrm{O}(n)$.

Ou seja, o custo total para construir os fechos $P_3, \ldots, P_{n-1}$ 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.

Análise do incremental probabilístico

Lema: O número esperado de faces criadas pelo Incremental3D probabilístico é no máximo $6n - 20$.

Então o tempo para remover e acrescentar faces no total é $\mathrm{O}(n)$.

Ou seja, o custo total para construir os fechos $P_3, \ldots, P_{n-1}$ 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.

Como fazer isso mais rápido que na versão da aula passada do algoritmo?

Estrutura de dados extra

Para isso, o algoritmo mantém um grafo bipartido G

Estrutura de dados extra

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

Estrutura de dados extra

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

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

Estrutura de dados extra

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

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

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

Estrutura de dados extra

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

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

$F_{conflito}(p)$: conjunto das faces vizinhas em G ao ponto p
$P_{conflito}(f)$: conjunto dos pontos 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.

Atualização do fecho

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

Atualização do fecho

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.

Atualização do fecho

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.

Atualização do fecho

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.

Atualização do grafo bipartido

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.

Atualização do grafo bipartido

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 face 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.

Atualização do grafo bipartido

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

Atualização do grafo bipartido

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 $\mathrm{O}(|P(e)|)$.

Atualização do grafo bipartido

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 $\mathrm{O}(|P(e)|)$.

Na Seção 11.3 do 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...

Atualização do grafo bipartido

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 $\mathrm{O}(|P(e)|)$.

Na Seção 11.3 do 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...

Esta prova usa os mesmos ingredientes que a prova do algoritmo probabilístico para construção da triangulação de Delaunay.

Conclusão

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

Quickhull 3D

Começa com um tetraedro, formado por quatro pontos da coleção, de preferência, pontos extremos já.

Quickhull 3D

Começa com um tetraedro, formado por quatro pontos da coleção, de preferência, pontos extremos já.

Divide os pontos entre as quatro faces do tetraedro, e os do interior.

Pontos que poderiam pertencer a duas faces
podem escolher arbitrariamente uma delas.

Quickhull 3D

Começa com um tetraedro, formado por quatro pontos da coleção, de preferência, pontos extremos já.

Divide os pontos entre as quatro faces do tetraedro, e os do interior.

Pontos que poderiam pertencer a duas faces
podem escolher arbitrariamente uma delas.

Para cada face por processar, ache, dentre seus pontos,
um ponto mais distante da face (usando Volume6) e
proceda com esse ponto como no algoritmo incremental:

Quickhull 3D

Começa com um tetraedro, formado por quatro pontos da coleção, de preferência, pontos extremos já.

Divide os pontos entre as quatro faces do tetraedro, e os do interior.

Pontos que poderiam pertencer a duas faces
podem escolher arbitrariamente uma delas.

Para cada face por processar, ache, dentre seus pontos,
um ponto mais distante da face (usando Volume6) e
proceda com esse ponto como no algoritmo incremental:

remova as faces visíveis do fecho corrente
e forme faces novas com o ponto processado.

Quickhull 3D

Começa com um tetraedro, formado por quatro pontos da coleção, de preferência, pontos extremos já.

Divide os pontos entre as quatro faces do tetraedro, e os do interior.

Pontos que poderiam pertencer a duas faces
podem escolher arbitrariamente uma delas.

Para cada face por processar, ache, dentre seus pontos,
um ponto mais distante da face (usando Volume6) e
proceda com esse ponto como no algoritmo incremental:

remova as faces visíveis do fecho corrente
e forme faces novas com o ponto processado.

Depois redivida os pontos da face processada
entre as novas e a parte interna, que agora aumentou.

Quickhull 3D

Começa com um tetraedro, formado por quatro pontos da coleção, de preferência, pontos extremos já.

Divide os pontos entre as quatro faces do tetraedro, e os do interior.

Pontos que poderiam pertencer a duas faces
podem escolher arbitrariamente uma delas.

Para cada face por processar, ache, dentre seus pontos,
um ponto mais distante da face (usando Volume6) e
proceda com esse ponto como no algoritmo incremental:

remova as faces visíveis do fecho corrente
e forme faces novas com o ponto processado.

Depois redivida os pontos da face processada
entre as novas e a parte interna, que agora aumentou.

Consumo de tempo: $O(n^2)$
Experimentalmente melhor que o incremental probabilístico.

Casos degenerados

Ao buscar um ponto extremo (no Embrulho, ou no Quickhull), cuidado com empates:
garanta que o ponto escolhido é de fato extremo.

Casos degenerados

Ao buscar um ponto extremo (no Embrulho, ou no Quickhull), cuidado com empates:
garanta que o ponto escolhido é de fato extremo.

Construa um poliedro simplifical e depois, se necessário, faça ajustes para "grudar" faces adjacentes coplanares.

Casos degenerados

Ao buscar um ponto extremo (no Embrulho, ou no Quickhull), cuidado com empates:
garanta que o ponto escolhido é de fato extremo.

Construa um poliedro simplifical e depois, se necessário, faça ajustes para "grudar" faces adjacentes coplanares.

No incremental, cuidado com faces coplanares com o ponto p_k.