SÉRGIO LIFSCHITZ

UM ALGORITMO LINEAR PARA O CÁLCULO DE ALGUMAS
FUNÇÕES DISTANCIA ENTRE POLÍGONOS CONVEXOS

Dissertação de mestrado apresentada ao
Departamento de Engenharia Elétrica da
PUC/RJ como parte dos requisitos para
obtenção do título de Mestre em
Ciências em Engenharia Elétrica.

ORIENTADOR : CELSO CARNEIRO RIBEIRO

DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RIO DE JANEIRO, 9 DE AGOSTO DE 1990

Para a Betina

- ao Professor Celso Carneiro Ribeiro pela orientação, confiança e incentivo.
- aos professores Antonio Oliveira, Marcelo Gattass, Nelson Maculan e Ronaldo Marinho, pela participação em minha banca examinadora e pelas sugestões apresentadas.
- aos meus colegas de mestrado e da "favelinha", pela troca de idéias, cafezinho e torcida fundamentais.
- a todo o pessoal da secretaria do DEE, pela ajuda e apoio em todos os momentos.
- a CAPES, pela ajuda financeira recebida durante o curso.
- a meus pais Thea e Jacob, por absolutamente tudo.
- a minha irmãzinha Danielle, pelas dicas e conselhos úteis.
- aos maridos Bia e Sidnei, por estarem sempre presentes.
- ao camarada Guijermo, pela amizade e companhia.
- ao Celso, pela amizade e pela força de sempre, dentro e fora da PUC.

RESUMO

Apresenta-se nesta dissertação um novo algoritmo para o cálculo de algumas funções distância entre polígonos convexos, no caso geral em que os polígonos podem se interseptar, cuja complexidade linear de pior caso é melhor do que a dos algoritmos até então conhecidos na literatura. O algoritmo é baseado em um algoritmo de complexidade linear originalmente proposto para a determinação da distância de Hausdorff entre polígonos convexos disjuntos e utiliza como sua principal componente um algoritmo linear para o cálculo da interseção entre polígonos convexos. A motivação para o estudo de algoritmos eficientes para este problema de cálculo de distâncias decorre de aplicações em reconhecimento de formas e superposição ótima de contornos. Resultados computacionais também são apresentados.

ABSTRACT

We present in this dissertation a new algorithm for the computation of some distance functions between convex polygons, in the general case where they can intersect, whose worst case time complexity is better than that of the previously known algorithms. The algorithm is based on an algorithm originally proposed for the computation of the Hausdorff distance between disjoint polygons and uses as its main component a linear time algorithm for finding the intersection of convex polygons. The motivation for the study of efficient algorithms for this distance computation problem comes from applications in pattern recognition and contour fitting. Computational results are also presented.

SUMÁRIO

LISTA DE FIGURAS.....	V
CAPÍTULO I. INTRODUÇÃO	
1. MOTIVAÇÃO.....	1
2. DEFINIÇÃO DO PROBLEMA.....	2
3. REVISÃO BIBLIOGRÁFICA.....	6
4. ORGANIZAÇÃO DO TEXTO.....	10
CAPÍTULO II. CÁLCULO DA DISTÂNCIA ENTRE DOIS POLÍGONOS CONVEXOS QUE NÃO SE INTERSEPTAM	
1. INTRODUÇÃO.....	11
2. UM ALGORITMO TRIVIAL.....	12
3. ALGORITMO DE COMPLEXIDADE LINEAR.....	13
3.1 Considerações Preliminares.....	14
3.2 Descrição do Algoritmo Linear.....	18
4. RESULTADOS COMPUTACIONAIS.....	21
CAPÍTULO III. UM ALGORITMO LINEAR PARA O CASO GERAL DE POLÍGONOS CONVEXOS	
1. INTRODUÇÃO.....	24
2. UM ALGORITMO TRIVIAL.....	25
3. ALGORITMO DE COMPLEXIDADE LINEAR.....	27
3.1 Considerações Preliminares.....	27
3.2 Descrição do Algoritmo Linear.....	32
4. RESULTADOS COMPUTACIONAIS.....	37
CAPÍTULO IV. CONCLUSÕES.....	39
REFERÊNCIAS BIBLIOGRÁFICAS.....	40

APÊNDICE 1 : GERAÇÃO ALEATÓRIA DE POLÍGONOS CONVEXOS.....	42
APÊNDICE 2 : DETERMINAÇÃO DOS PONTOS INTERIORES A UM POLÍGONO CONVEXO.....	46
APÊNDICE 3 : ALGORITMO LINEAR PARA A DETERMINAÇÃO DA INTERSEÇÃO ENTRE DOIS POLÍGONOS CONVEXOS.....	51

LISTA DE FIGURAS

I-1	Polígono P_1 com $n_1 = 7$ vértices.....	3
I-2	Distâncias $\delta_{P_1}(u, v)$	4
I-3	Distância de Hausdorff $D_H(P_1, P_2)$	5
I-4	Distância mínima $d(P_1, P_2)$	7
I-5	Propriedade unimodal.....	9
II-1	Algoritmo trivial: distâncias d_a e d_v	13
II-2	Retas suporte a um polígono convexo.....	14
II-3	Lema 1.....	16
II-4	Movimentos da sequência σ e Lema 2.....	17
II-5	Interpretação do teste da reta suporte pode acarretar a determinação de uma projeção incorreta.....	20
II-6	Tabela com os resultados computacionais obtidos pelos algoritmos $O(n_1 n_2)$ e $O(n_1 + n_2)$ para polígonos disjuntos.....	22
II-7	Gráfico comparativo do desempenho dos algoritmos $O(n_1 n_2)$ e $O(n_1 + n_2)$ para polígonos disjuntos.....	23
III-1	Algoritmo de Atallah pode gerar projeções incorretas no caso geral em que os polígonos podem se interceptar	25
III-2	Vértices interiores e distâncias relevantes para o cálculo de $D_{CMMR}(P_1, P_2)$	26
III-3	Região P e sua decomposição em pedaços.....	28
III-4	Pedaços e respectivas cadeias interna e externa.....	29
III-5	Um pedaço e respectivos pedaços predecessor e sucessor	30

III-6	Polígonos construídos a partir das cadeias externas e de arestas artificiais.....	31
III-7	Merge entre as cadeias externa e interna de um pedaço.	33
III-8	Estrutura de Dados utilizada para armazenar os pedaços	35
III-9	Tabela com os resultados computacionais obtidos pelos algoritmos $O(n_1 n_2)$ e $O(n_1 + n_2)$ para o caso geral onde os polígonos podem se interceptar.....	37
III-10	Gráfico comparativo do desempenho dos algoritmos $O(n_1 n_2)$ e $O(n_1 + n_2)$ para o caso geral onde os polígonos podem se interceptar.....	38
A1-1	Exemplo de geração aleatória de um polígono.....	43
A1-2	Situação onde a geração aleatória de um próximo vértice pode ser muito difícil.....	45
A2-1	Centróide, cunhas e ângulos horário e anti-horário....	47
A3-1	Definições e notações necessárias para a descrição do algoritmo linear para o cálculo da interseção entre polígonos convexos.....	52
A3-2	Exemplo de execução do algoritmo linear para o cálculo da interseção entre polígonos convexos.....	53
A3-3	Casos degenerados.....	56

CAPITULO I

INTRODUÇÃO

1. MOTIVAÇÃO

O cálculo da projeção de um ponto no $\mathbb{R}^2$ em um polígono convexo é um problema básico de geometria computacional. Uma extensão deste problema consiste na determinação da distância entre dois polígonos convexos, disjuntos ou não, tomando-se como função distância a soma das distâncias entre cada vértice de cada polígono e sua projeção no outro polígono.

Uma aplicação deste problema é apresentada no artigo de Cox, Maitre, Minoux e Ribeiro [COX89], onde é descrito um algoritmo para o problema de ajuste ótimo de dois polígonos convexos, representando, no caso, dois conjuntos de observações diferentes retratando contornos de uma mesma secção plana de um dado objeto. Em cada passo do algoritmo, o principal sub-problema a ser resolvido consiste na determinação da projeção de um vértice qualquer de um dos polígonos no outro polígono. O ajuste ótimo é atingido quando se determina a posição relativa entre os dois polígonos que minimiza uma certa distância entre os mesmos.

Naquele artigo, utiliza-se um algoritmo trivial de complexidade computacional $O(n_1 n_2)$ para o cálculo da distância entre dois polígonos convexos P_1 e P_2 , onde n_1 e n_2 são os números de vértices de cada polígono respectivamente. Utilizando-se algoritmos de complexidade computacional logarítmica disponíveis na literatura (tais como os descritos em [CHIN83] ou em [EDEL85]), para o cálculo da projeção de cada vértice, pode-se construir um algoritmo de complexidade $O(n_1 \log n_2 + n_2 \log n_1)$ para o cálculo da mesma distância entre P_1 e P_2 .

Um algoritmo de complexidade linear $O(n_1 + n_2)$ para o cálculo de tal distância é apresentado nesta dissertação. Este algoritmo é baseado em um artigo de Atallah [ATAL83] para o cálculo da distância de Hausdorff no caso em que não há interseção entre os dois polígonos.

Como o cálculo da distância representa mais de 95% do tempo de execução do algoritmo descrito em [COX89], este novo algoritmo de complexidade linear poderá ser utilizado para melhorar sensivelmente seu desempenho.

2. DEFINIÇÃO DO PROBLEMA

Sejam P_1 e P_2 dois polígonos convexos tais que:

- (a) P_1 (resp. P_2) tem n_1 (resp. n_2) vértices;
- (b) O vértice v_j^1 (resp. v_j^2) de P_1 (resp. P_2) tem coordenadas dadas por (a_j^1, b_j^1) , $j \in \{1, 2, \dots, n_1\}$ (resp. (a_j^2, b_j^2) , $j \in \{1, 2, \dots, n_2\}$) e
- (c) A numeração dos vértices de P_1 (resp. P_2) é tal que, para qualquer $j \in \{1, 2, \dots, n_1\}$ (resp. $j \in \{1, 2, \dots, n_2\}$), se definir-se

$$ind(j) = \begin{cases} j, & \text{se } j \leq n_1 \text{ (resp. } j \leq n_2), \\ ind(j - n_1) \text{ (resp. } ind(j - n_2)), & \text{se } j > n_1 \text{ (resp. } j > n_2), \end{cases}$$

então o segmento de reta unindo os vértices v_j^1 e $v_{ind(j+1)}^1$ (resp. v_j^2 e $v_{ind(j+1)}^2$) é a j -ésima aresta de P_1 (resp. P_2).

Supõe-se ainda que os vértices de P_1 e P_2 sejam numerados no sentido anti-horário [Fig. I-1].

FIGURA [I-11]: Polígono P_1 com $n_1 = 7$ vértices numerados em sentido anti-horário.

A distância euclidiana entre dois pontos do plano p_1 e p_2 , definidos por suas respectivas coordenadas (u_1, v_1) e (u_2, v_2) , é dada por $||p_2 - p_1|| = ((u_2 - u_1)^2 + (v_2 - v_1)^2)^{1/2}$. Sendo (u, v) um ponto qualquer do plano, a distância $\delta_{P_\ell}(u, v)$, de (u, v) até o polígono P_ℓ ($\ell = 1, 2$), é definida por

$$\delta_{P_\ell}(u, v) = \begin{cases} \|(u, v) - Proj_{P_\ell}(u, v)\|, & \text{se } (u, v) \notin Interior(P_\ell), \\ 0, & \text{caso contrário,} \end{cases}$$

onde $Proj_{P_\ell}(u, v)$ é a projeção de (u, v) em P_ℓ (isto é, o ponto de

P_ℓ com menor distância euclidiana até o ponto (u,v) e $\text{Interior}(P_\ell)$ representa o interior do polígono P_ℓ (isto é, P_ℓ menos sua fronteira) [Fig. I-2].

FIGURA [I-2]: Distâncias $\delta_{P_1}(u_1, v_1)$, $\delta_{P_1}(u_2, v_2)$ e $\delta_{P_1}(u_3, v_3)$, respectivamente dos pontos q_1 , q_2 e q_3 até o polígono P_1 .

Considerando tais notações, a distância definida em [COX89] é dada por

$$D_{\text{CMR}}^{(P_1, P_2)} = D_{\text{CMR}}^{(P_2, P_1)} = \sum_{j=1}^{n_1} \left[\delta_{P_2}(a_j^1, b_j^1) \right]^2 + \sum_{j=1}^{n_2} \left[\delta_{P_1}(a_j^2, b_j^2) \right]^2,$$

isto é, a soma dos quadrados das distâncias euclidianas de cada vértice de cada polígono até sua projeção no outro polígono. Já a distância de Hausdorff [GRUN67], considerada em [ATAL83], é dada por

$$D_H(P_1, P_2) = D_H(P_2, P_1) = \max_{\ell=1,2} \left\{ \max_{j=1,2,\dots,n_\ell} \delta_{P_{3-\ell}}(a_j^\ell, b_j^\ell) \right\} =$$

$$= \max \left\{ \max_{j=1,2,\dots,n_1} \delta_{P_2}(a_j^1, b_j^1), \max_{j=1,2,\dots,n_2} \delta_{P_1}(a_j^2, b_j^2) \right\},$$

isto é, a máxima distância euclidiana de qualquer vértice de qualquer polígono até sua projeção no outro polígono [Fig. I-3].

FIGURA [I-3]: Distância de Hausdorff $D_H(P_1, P_2)$ entre os polígonos P_1 e P_2 .

Deseja-se então desenvolver um algoritmo eficiente para o cálculo de D_{CMMR} . O cálculo de D_H poderá ser feito por um algoritmo similar, pois ambas as distâncias são calculadas determinando-se todas as projeções de cada vértice de cada polígono no outro, sendo que, ao invés de somar-se os quadrados de todas as distâncias euclidianas calculadas, como no cálculo de D_{CMMR} , identifica-se a maior destas distâncias.

3. REVISÃO BIBLIOGRÁFICA

O problema do cálculo da distância entre dois polígonos convexos é mencionado em vários artigos e livros como um problema básico de geometria computacional. Para a distância D_{CMMR} também são muito importantes os algoritmos básicos existentes na literatura referentes ao cálculo da projeção de um ponto em um polígono convexo.

Sejam x e y dois pontos quaisquer de $\mathbb{R}^2$ e $d(x,y)$ a distância euclidiana entre os mesmos. Sendo P_1 e P_2 dois polígonos convexos e $F(P_1)$ e $F(P_2)$ as suas respectivas fronteiras, a distância mínima entre P_1 e P_2 é normalmente definida por

$$d(P_1, P_2) = \min \{ d(x, y) \mid x \in F(P_1) \text{ e } y \in F(P_2) \},$$

isto é, $d(P_1, P_2)$ corresponde à distância euclidiana entre dois pontos, um em cada polígono, que são os mais próximos dentre todos os pontos das fronteiras dos dois polígonos [Fig. I-4].

Se a interseção entre os dois polígonos é não vazia e nenhum dos polígonos contém o outro, existe pelo menos um ponto da interseção que pertence à fronteira de ambos polígonos e, portanto, $d(P_1, P_2) = 0$. Caso contrário, $d(P_1, P_2) > 0$ e ou x ou y (ou ambos) é necessariamente um vértice. Em 1981, Schwartz

apresentou um algoritmo de complexidade computacional $O(\log n_1 \cdot \log n_2)$ para a determinação de $d(P_1, P_2)$, baseado em uma dupla pesquisa binária nos vértices dos polígonos [SCHW81].

FIGURA [I-4]: Distância mínima $d(P_1, P_2)$ entre os polígonos P_1 e P_2 .

Chin e Wang propuseram em 1983 um algoritmo que resolve o mesmo problema com complexidade $O(\log n_1 + \log n_2)$ [CHIN83]. Para isto, consideraram as seguintes definições e os seguintes fatos relativos à geometria de polígonos convexos:

Definição I3.1: A cadeia de vértices $\langle v_j^1, v_k^1 \rangle$ de P_1 é definida pelo conjunto de pontos $\langle v_j^1, v_{ind(j+1)}^1, \dots,$

$v_{ind(k-1)}^1, v_k^1$), que é um subconjunto ordenado do conjunto de vértices $\langle v_1^1, v_2^1, \dots, v_{n_1}^1 \rangle$ de P_1 .

Definição I3.2: A cadeia de arestas $\langle v_j^1, v_k^1 \rangle$ de P_1 é o conjunto de arestas associado à cadeia de vértices $\langle v_j^1, v_k^1 \rangle$, isto é, o conjunto $\langle (v_j^1, v_{ind(j+1)}^1), (v_{ind(j+1)}^1, v_{ind(j+2)}^1), \dots, (v_{ind(k-1)}^1, v_k^1) \rangle$, onde cada elemento é uma aresta definida por um par de vértices consecutivos de P_1 .

Definição I3.3: A cadeia de arestas $\langle v_j^1, v_k^1 \rangle$ de P_1 visível a partir de um ponto q externo a P_1 é definida pela parte da fronteira de P_1 visível a partir de q , isto é, o conjunto de pontos x pertencentes à $F(P_1)$ tais que o segmento de reta definido por q e x não contém outros pontos de $F(P_1)$.

Considerando-se as definições acima, tem-se que o ponto $p \in P_1$ mais próximo de um ponto q da fronteira de P_2 pertencerá à cadeia de arestas $\langle v_j^1, v_k^1 \rangle$ de P_1 visível a partir de q . Além disso, percorrendo-se esta cadeia monotonicamente a partir de v_j^1 , as distâncias de q até os pontos da cadeia serão decrescentes até se atingir o ponto p que lhe é mais próximo, tornando-se crescentes deste ponto até v_k^1 [Fig. I-5]. Isto é, a distância de q até os pontos da cadeia de arestas $\langle v_j^1, v_k^1 \rangle$ de P_1 visível a partir de q é dita ser unimodal com respeito ao ponto p pertencente à cadeia.

O algoritmo proposto em [CHIN83] utiliza na determinação de $d(P_1, P_2)$ o fato de que a cadeia de arestas de P_1 visível a partir dos pontos de P_2 e a cadeia de arestas de P_2 visível a partir dos pontos de P_1 têm a propriedade de serem unimodais. A

FIGURA [I-5]: Distância de q até a cadeia de arestas $\langle v_2^1, v_5^1 \rangle$ de P_1 visível a partir de q é unimodal com respeito a p .

idéia básica do algoritmo consiste em determinar-se as duas cadeias visíveis de cada polígono a partir dos pontos do outro polígono em $O(\log n_1 + \log n_2)$ para, em seguida, determinar-se os pontos $p_1 \in P_1$ e $p_2 \in P_2$ tais que $d(p_1, p_2) = d(P_1, P_2)$ através de uma pesquisa binária em cada cadeia.

Edelsbrunner [EDEL85] propôs independentemente um algoritmo semelhante, de mesma complexidade, porém utilizando uma estratégia de determinação das cadeias e execução da pesquisa binária diferentes. No mesmo artigo, apresenta-se um algoritmo para o cálculo dos pontos que verificam a distância máxima entre P_1 e P_2 , definida de forma análoga à distância mínima.

Os algoritmos propostos em [CHIN83] e em [EDEL85] podem ser

aproveitados para a determinação da projeção de um ponto qualquer do plano em um polígono convexo, considerando-se o caso limite de um dos polígonos ser formado por um único vértice. Logo, o cálculo de $D_{\text{CMMR}}(P_1, P_2)$ poderia ser feito em $O(n_1 \log n_2 + n_2 \log n_1)$, com menor complexidade do que a do algoritmo trivial utilizado em [COX89].

O algoritmo de complexidade linear para o cálculo da distância de Hausdorff entre polígonos disjuntos proposto por Atallah [ATAL83] será discutido no próximo capítulo. Este algoritmo pode ser adaptado para o cálculo de $D_{\text{CMMR}}(P_1, P_2)$, também para polígonos disjuntos. Para o caso geral onde os polígonos podem se interceptar, um novo algoritmo será apresentado no capítulo III.

4. ORGANIZAÇÃO DO TEXTO

O texto desta dissertação está organizado da seguinte maneira: No próximo capítulo, apresenta-se o algoritmo para o cálculo de $D_{\text{CMMR}}(P_1, P_2)$ no caso em que P_1 e P_2 são disjuntos. Este algoritmo é uma adaptação daquele apresentado em [ATAL83], que também pode ser facilmente adaptado para outras funções distância. Resultados computacionais são apresentados.

No capítulo III, propõe-se e discute-se em detalhes o algoritmo para o caso em que P_1 e P_2 se interceptam. Resultados computacionais comparativos com o algoritmo utilizado em [COX89] são apresentados e comentados.

O capítulo IV é dedicado à apresentação de conclusões e extensões. As descrições detalhadas dos outros algoritmos referenciados no texto são apresentadas nos apêndices.

CAPÍTULO II

CÁLCULO DA DISTÂNCIA ENTRE DOIS POLÍGONOS CONVEXOS
QUE NÃO SE INTERSEPTAM

1. INTRODUÇÃO

Considere o problema da determinação da distância entre dois polígonos disjuntos P_1 e P_2 , isto é, que têm interseção vazia. O cálculo de $D_{\text{CMR}}(P_1, P_2)$ em [COX89] é feito em $O(n_1 n_2)$, porém, com os algoritmos apresentados em [CHIN83] e [EDEL85], utilizados para a determinação da projeção de um ponto em um polígono convexo em $O(\log n)$, a complexidade deste cálculo pode ser reduzida para $O(n_1 \log n_2 + n_2 \log n_1)$.

Para o problema do cálculo da distância de Hausdorff [GRUN67] entre dois polígonos convexos, Atallah [ATAL83] propôs um algoritmo linear que pode ser facilmente adaptado para calcular $D_{\text{CMR}}(P_1, P_2)$, também em tempo linear, quando os polígonos são disjuntos. Tal algoritmo tira proveito da convexidade dos polígonos para calcular as projeções de cada vértice de um polígono no outro, utilizando como referência na determinação de cada projeção a última projeção calculada.

Cabe observar que o algoritmo proposto por Atallah não deve ser aproveitado para o cálculo da projeção de um ponto em um polígono convexo, pois neste caso a complexidade seria linear, superior portanto à dos algoritmos de complexidade logarítmica relacionados anteriormente.

2. UM ALGORITMO TRIVIAL

Em [COX89], o cálculo da distância entre dois polígonos convexos é feito por um algoritmo trivial de complexidade quadrática, descrito a seguir. Para determinar a distância de P_1 até P_2 , o que o algoritmo basicamente faz é calcular, para cada vértice de P_1 , as distâncias até todos os vértices e até todas as projeções nas arestas de P_2 (considerando-se apenas as projeções válidas em que o pé da perpendicular do vértice de P_1 à reta suporte da aresta encontra-se entre os vértices extremos da mesma), somando-se os quadrados das menores distâncias encontradas [Fig. II-1]. Completa-se o cálculo de $D_{\text{CMMR}}(P_1, P_2)$ determinando-se a distância de P_2 até P_1 de forma análoga, invertendo-se o papel dos polígonos. Apenas a primeira parte do cálculo de $D_{\text{CMMR}}(P_1, P_2)$ é apresentada, isto é, o cálculo da distância de P_1 até P_2 . A segunda parte é omitida, por ser análoga.

ALGORITMO [COX89]

PASSO 0: (inicialização)

Fazer $i \leftarrow 1$ e $D_{\text{CMMR}}(P_1, P_2) \leftarrow 0$.

PASSO 1: (distância até os vértices de P_2)

Determinar a distância de v_i^1 até todos os vértices de P_2 . Seja d_v o menor valor encontrado.

PASSO 2: (distância até as arestas de P_2)

Determinar a distância de v_i^1 até todas as arestas de P_2 . Seja d_a o menor valor encontrado.

PASSO 3: (atualização de D_{CMMR} e condição de parada)

Fazer $D_{\text{CMMR}}(P_1, P_2) \leftarrow D_{\text{CMMR}}(P_1, P_2) + \left[\min(d_v, d_a) \right]^2$ e

$i \leftarrow \text{ind}(i+1)$. Se $i = 1$ então PARAR; senão, voltar ao passo 1.

FIGURA [II-11]: d_a e d_v são as menores distâncias encontradas até o polígono P_2 a partir do vértice v_3^1 de P_1 .

A complexidade dos passos 1 e 2 é $O(n_2)$. Como os mesmos são executados uma vez para cada um dos n_1 vértices de P_1 , a primeira parte do algoritmo tem complexidade $O(n_1 n_2)$. A complexidade da segunda parte é análoga e, portanto, o algoritmo trivial tem complexidade total $O(n_1 n_2)$.

3. ALGORITMO DE COMPLEXIDADE LINEAR

Considera-se a determinação da primeira parte de $D_{\text{CMMR}}(P_1, P_2)$, isto é, a soma dos quadrados das distâncias euclidianas de cada

vértice v_j^1 de P_1 até sua projeção em P_2 . Representando-se por d_j a distância do vértice v_j^1 a P_2 ($d_j = \delta_{P_2}(\alpha_j^1, b_j^1)$), cada d_j poderia ser calculado separadamente em $O(n_2)$, o que resultaria em um tempo total de $O(n_1 n_2)$, complexidade equivalente à do algoritmo trivial. Mostra-se, em seguida, como calcular todas as distâncias d_j (e consequentemente $D_{\text{CMMR}}(P_1, P_2)$) em $O(n_1 + n_2)$, observando-se primeiramente alguns fatos relativos à convexidade dos polígonos.

FIGURA [II-2]: Exemplos de retas suporte (r_1 e r_2) do polígono P .

3.1. Considerações Preliminares

Define-se uma reta suporte de um polígono convexo P como uma

reta r que passa por um vértice de P e é tal que o interior de P se localiza inteiramente em um mesmo lado de r [PREP85]. r também pode ser uma reta suporte sendo colinear com uma aresta de P [Fig. II-2].

LEMA 1 [ATAL83]

Seja $x = (u, v)$ um vértice de P_2 (resp. P_1) e $y = Proj_{P_1}(u, v)$ (resp. $Proj_{P_2}(u, v)$) sua projeção em P_1 (resp. P_2). Então, a perpendicular à reta xy em y é uma reta suporte de P_1 (resp. P_2) e x e P_1 (resp. P_2) estão em lados diferentes com relação a esta reta [Fig. II-3].

O lema 1 será útil quando, no algoritmo que será descrito a seguir, se desejar verificar se um dado ponto de um dos polígonos é de fato a projeção de um vértice do outro polígono. Para todo $j = 1, 2, \dots, n_1$, seja y_j o ponto de P_2 para o qual $d_j = \|v_j^1 - y_j\|$, isto é, $y_j = Proj_{P_2}(a_j^1, b_j^1)$. Nota-se que, ou y_j é um vértice de P_2 , ou então é o pé da perpendicular de v_j^1 a alguma das arestas de P_2 .

Seja σ a sequência $y_1, y_2, y_3, \dots, y_{n_1}, y_1$ dos pontos de P_2 que são projeções dos vértices de P_1 . Diz-se que σ está se estiver à direita, diz-se que σ está se movendo no sentido horário em y_j . Se v_{j+1}^1 pertence à reta $y_j v_j^1$, diz-se então que σ é estacionária em y_j [Fig. II-4].

Diz-se que a sequência σ muda de direção em y_j se:

- (i) σ é estacionária ou estiver se movendo em sentido horário em y_{j-1} e estiver se movendo em sentido anti-horário

FIGURA [II-3]: A reta perpendicular à reta xy em y ($x = v_2^1$ e $y = v_2^2$) é suporte de P_2 e separa x e P_2 em lados diferentes.

em y_j , ou

(ii) σ é estacionária ou estiver se movendo em sentido anti-horário em y_{j-1} e estiver se movendo em sentido horário em y_j .

LEMA 2 [ATAL83]

A sequência σ muda de direção exatamente duas vezes.

FIGURA [II-4]: A sequência σ está se movendo no sentido horário em y_1 e é estacionária em y_2 . A sequência σ muda duas vezes de direção, nos pontos y_3 e y_5 .

O fato descrito no lema 2 pode ser verificado pela propriedade unimodal de P_1 e P_2 , definida na seção 3 do capítulo anterior. Pode-se dividir o polígono P_2 em dois conjuntos, um contendo a cadeia de arestas de P_2 visíveis a partir dos pontos de P_1 e o outro com as demais arestas. Todas as projeções dos vértices de P_1 no polígono P_2 pertencem ao primeiro conjunto e as duas projeções extremas são aquelas onde σ muda de direção, isto é, ocorrem exatamente duas mudanças de direção [Fig II-4].

3.2. Descrição do Algoritmo Linear

O algoritmo para a determinação de $D_{\text{CMMR}}(P_1, P_2)$ descrito a seguir é uma adaptação do algoritmo proposto por Atallah para o cálculo da distância de Hausdorff [ATAL83]. A primeira parte do algoritmo consiste no cálculo da soma dos quadrados das distâncias de cada vértice de P_1 até sua projeção em P_2 . A segunda parte é análoga, invertendo-se os papéis dos polígonos, isto é, consiste no cálculo da soma dos quadrados das distâncias de cada vértice de P_2 até sua projeção em P_1 . Apenas a primeira parte será apresentada aqui.

A idéia básica do algoritmo consiste em, inicialmente, determinar a distância d_1 e a projeção y_1 de v_1 em P_2 ($y_1 = \text{Proj}_{P_2}(a_1^1, b_1^1)$) em tempo linear, por qualquer algoritmo trivial (por exemplo, calculando-se todas as distâncias de v_1 até todos os vértices e até todas as arestas de P_2 e determinando-se a menor distância encontrada). Em seguida, calcula-se cada projeção y_{k+1} , $k = 1, 2, \dots, n_1 - 1$ (e consequentemente todas as demais distâncias que compõem D_{CMMR}), considerando-se a posição de v_{k+1} em relação ao segmento de reta orientado $\overrightarrow{y_k v_k}$ na determinação do sentido de percurso da fronteira de P_2 .

ALGORITMO $D_{\text{CMMR}}(P_1, P_2)$ polígonos disjuntos

PASSO 0: (determinação de y_1 e d_1)

Calcular d_1 e encontrar y_1 utilizando um algoritmo linear trivial. Fazer $D_{\text{CMMR}}(P_1, P_2) \leftarrow d_1$ e $k \leftarrow 1$.

PASSO 1: (determinação de y_{k+1} e d_{k+1})

Caso (i) v_{k+1}^1 está à esquerda de $y_k v_k^1$:

Obter y_{k+1} percorrendo o polígono P_2 em sentido anti-horário, começando em y_k , até que uma das seguintes condições seja satisfeita:

- (a) A aresta $v_q^2 v_{q+1}^2$ de P_2 em consideração é tal que o pé da perpendicular z de v_{k+1}^1 até $v_q^2 v_{q+1}^2$ está entre v_q^2 e v_{q+1}^2 . Neste caso, fazer $y_{k+1} \leftarrow z$.
- (b) O vértice v_q^2 de P_2 em consideração é tal que a perpendicular a $v_{k+1}^1 v_q^2$ em v_q^2 é uma reta suporte de P_2 . Neste caso, fazer $y_{k+1} \leftarrow v_q^2$.

Caso (ii) v_{k+1}^1 está à direita de $y_k v_k^1$:

Obter y_{k+1} percorrendo o polígono P_2 em sentido horário, começando em y_k , até que uma das seguintes condições seja satisfeita:

- (a) A aresta $v_q^2 v_{q+1}^2$ de P_2 em consideração é tal que o pé da perpendicular z de v_{k+1}^1 até $v_q^2 v_{q+1}^2$ está entre v_q^2 e v_{q+1}^2 . Neste caso, fazer $y_{k+1} \leftarrow z$.
- (b) O vértice v_q^2 de P_2 em consideração é tal que a perpendicular a $v_{k+1}^1 v_q^2$ em v_q^2 é uma reta suporte de P_2 . Neste caso, fazer $y_{k+1} \leftarrow v_q^2$.

Caso (iii) v_{k+1}^1 está sobre $y_k v_k^1$:

Neste caso, fazer $y_{k+1} \leftarrow y_k$.

PASSO 2: (atualização de D_{CMMR} e condição de parada)

Fazer $D_{\text{CMMR}}(P_1, P_2) \leftarrow D_{\text{CMMR}}(P_1, P_2) + (\|v_{k+1}^1 - y_{k+1}\|)^2$ e $k \leftarrow \text{ind}(k+1)$. Se $k = 1$ então PARAR; senão, voltar ao passo 1.

Como todas as distâncias de todos os vértices de P_1 até P_2 e vice-versa são calculadas, fica mostrada a correção do algoritmo. A complexidade do passo 0 é $O(n_2)$. O lema 2 garante que o polígono P_2 não será percorrido mais do que duas vezes durante toda a execução do algoritmo, isto é, durante as n_1

iterações controladas pelo passo 2. Logo, no pior caso, a complexidade global do passo 1 é $O(n_2)$ e a do passo 2 é $O(n_1)$. Para o algoritmo como um todo, tem-se $O(n_2) + O(n_1 + n_2)$ na primeira parte e $O(n_1) + O(n_2 + n_1)$ na segunda parte, ou seja, $O(n_1 + n_2)$.

Cabe observar que, no passo 1 do algoritmo, na condição (b) dos casos (i) e (ii), a definição de reta suporte a um polígono convexo deve ser considerada obrigatoriamente de acordo com o lema 1. Isto é, não basta apenas ser reta suporte como definido na seção 3.1 deste capítulo pois, tomando-se como exemplo os polígonos da figura [II-5], poderia ocorrer a determinação de uma projeção incorreta. De fato, executando-se o algoritmo, tem-se que:

FIGURA [II-5]: Situação em que a perpendicular à reta $v_1^1 v_2^2 v_3^2$ em v_3^2 é reta suporte do polígono P_2 mas v_1^2 e P_2 estão do mesmo lado relativo a esta reta.

- (1) Determina-se primeiramente d_1 , distância de v_1^1 até $y_1 = v_3^2$.
- (2) Como v_2^1 está à direita de $y_1 v_1^1$ ($v_3^2 v_1^1$), y_2 será determinado percorrendo-se P_2 em sentido horário a partir de $y_1 = v_3^2$. A condição (b) é verificada e, pela própria figura, pode-se observar que v_3^2 não é a projeção correta de v_1^1 (na realidade, v_2^2 é a projeção correta).

Esta situação ocorre pois, apesar da reta perpendicular a $v_2^1 v_3^2$ em v_3^2 ser suporte de P_2 , ela não verifica o lema 1 pois v_2^1 e P_2 estão do mesmo lado relativo àquela reta (na verdade, a reta $v_2^1 v_3^2$ já é por si só reta suporte de P_2).

4. RESULTADOS COMPUTACIONAIS

O algoritmo linear e o algoritmo trivial foram implementados e comparados em setenta problemas-teste, dez para cada número de vértices $n = n_1 = n_2 \in \{5, 10, 15, 20, 25, 30, 35\}$. Cada problema é definido por dois polígonos convexos gerados aleatoriamente. Os vértices de P_1 são gerados no quadrado unitário com diagonal $((0,0), (1,1))$ e os de P_2 gerados no quadrado unitário com diagonal $((0,0), (1,-1))$. Os vértices de cada polígono são gerados um a um, de forma a garantir sua convexidade (Apêndice 1) e são armazenados em uma lista circular duplamente encadeada, que facilita o acesso de um determinado vértice a seus vértices adjacentes, permitindo o percurso sequencial eficiente dos vértices do polígono, em sentido horário e anti-horário. Esta será a estrutura de dados utilizada para a representação dos polígonos em todo o trabalho.

Os tempos médios dos dez problemas-teste para cada n (dados em segundos, observados em um microcomputador IBM PC/AT 12 MHz

compatível com co-processador aritmético 80287) e o gráfico comparativo dos valores obtidos são apresentados a seguir, respectivamente nas figuras [II-6] e [II-7].

n	Alg. $O(n_1 n_2)$	Alg. $O(n_1 + n_2)$
5	0.203	0.099
10	0.763	0.226
15	1.688	0.345
20	2.989	0.490
25	4.632	0.615
30	6.675	0.714
35	9.081	0.855

FIGURA [II-6]: Tabela com os resultados computacionais obtidos pelos algoritmos $O(n_1 n_2)$ e $O(n_1 + n_2)$ quando os polígonos são disjuntos.

FIGURA [II-7]: Gráfico comparativo do desempenho dos algoritmos $O(n_1n_2)$ e $O(n_1 + n_2)$ para polígonos disjuntos.

CAPÍTULO III

UM ALGORITMO LINEAR PARA O CASO GERAL DE
POLÍGONOS CONVEXOS

1. INTRODUÇÃO

Neste capítulo considera-se o problema do cálculo da distância no caso geral em que a interseção de P_1 e P_2 pode ser não vazia. Caso um dos polígonos contenha o outro, a solução é trivial, utilizando-se o algoritmo linear descrito no capítulo anterior e calculando-se apenas as distâncias a partir dos vértices do polígono externo (para os vértices do polígono interno, as distâncias são nulas por definição). Neste caso, percorrer completamente uma única vez o polígono interior é suficiente para determinar a distância desejada. Caso contrário, o algoritmo apresentado no capítulo II não pode ser utilizado pois, como mostrado no exemplo da figura [III-1], após a determinação das projeções dos vértices v_1^1 , v_2^1 e v_3^1 , respectivamente y_1 , y_2 e y_3 , deve-se percorrer o polígono P_2 no sentido anti-horário para se determinar a projeção do vértice v_4^1 e, no lugar do ponto y_4 indicado na figura, obtém-se o ponto $\bar{y}_4$ como projeção. Por isto, um novo algoritmo linear deve ser utilizado.

A idéia básica do algoritmo linear a ser proposto consiste em identificar em tempo linear os "pedaços" dos polígonos P_1 e P_2 que são relevantes para a determinação de $D_{\text{CMMR}}(P_1, P_2)$. Tais "pedaços" são compostos por uma cadeia poligonal externa formada pelos vértices de P_1 e P_2 que contribuem com distância não nula para $D_{\text{CMMR}}(P_1, P_2)$ e por uma cadeia poligonal interna, formada pelos vértices e arestas onde se encontrarão as projeções dos vértices da cadeia poligonal externa. Em seguida a este

FIGURA [III-1]: Situação em que a execução do algoritmo de Atallah resulta na determinação de uma projeção incorreta para v_4^1 , obtendo-se $\bar{y}_4$ e não y_4 .

pré-processamento, a identificação das projeções desejadas poderá ser efetuada em tempo linear, adaptando-se o algoritmo apresentado no último capítulo e executando-se uma espécie de merge entre as cadeias externa e interna.

2. UM ALGORITMO TRIVIAL

O algoritmo trivial aqui difere do algoritmo descrito para o caso de polígonos disjuntos apenas no que se refere aos vértices

de cada polígono que são interiores ao outro, pois, por definição, as distâncias nestes casos são nulas [Fig. III-2]. Daí, o que se faz é realizar um pré-processamento para identificar os vértices de cada polígono que são interiores ao outro e, posteriormente, aplica-se o mesmo algoritmo trivial apresentado no capítulo anterior apenas para aqueles vértices que contribuem com distância positiva para $D_{\text{CMMR}}^{(P_1, P_2)}$.

FIGURA [III-2]: O vértice $v_2^3 \in P_2$ é interno ao polígono P_1 e $v_1^6 \in P_1$ é interno à P_2 . Logo, não contribuem com distância positiva no cálculo de $D_{\text{CMMR}}^{(P_1, P_2)} = d_1^2 + d_2^2 + \dots + d_8^2$.

A identificação dos vértices relevantes de cada polígono para o cálculo de $D_{\text{CMMR}}^{(P_1, P_2)}$ pode ser obtida utilizando-se um

algoritmo que determina em tempo $O(\log n)$ se um dado ponto do plano é ou não interior a um polígono convexo, com pré-processamento em tempo $O(n)$ [PREP85] (este algoritmo é descrito no Apêndice 2). Tais pontos interiores não contribuem com distância positiva para $D_{\text{CMMR}}(P_1, P_2)$ e, portanto, não serão considerados na execução do algoritmo trivial descrito anteriormente. A complexidade total do algoritmo é $O(n_2 \log n_1) + O(n_1 \log n_2)$ para o pré-processamento e, no pior caso - cada polígono tendo apenas um vértice interior ao outro - $O(n_1 n_2)$ também na execução do algoritmo trivial.

3. ALGORITMO DE COMPLEXIDADE LINEAR

Antes de se descrever o novo algoritmo de complexidade linear para o cálculo de $D_{\text{CMMR}}(P_1, P_2)$, considera-se alguns fatos relativos à geometria do problema, detalhando-se a estratégia utilizada pelo algoritmo.

3.1. Considerações Preliminares

Dados P_1 e P_2 , seja $P = (P_1 \cup P_2) - \text{Interior}(P_1 \cap P_2)$ a região do plano em P_1 ou P_2 mas não no interior da interseção dos mesmos. Tanto a união como a interseção de dois polígonos convexos podem ser calculadas em tempo linear [ROUR82] (este algoritmo é descrito no Apêndice 3). Logo, uma descrição da região P também pode ser obtida em tempo linear.

A região P consiste de um número $m = O(n)$ de pedaços, onde $n = \max\{n_1, n_2\}$. Cada pedaço é uma região poligonal tipicamente não convexa. A fronteira de cada região consiste de duas cadeias poligonais: uma vindo da fronteira de P_1 , dita cadeia- P_1 , e a outra vindo da fronteira de P_2 , dita cadeia- P_2 . Suponha que os

FIGURA [III-3]: Região P com $m = 4$ pedaços.

pedaços que aparecem na decomposição de P são numerados em sentido anti-horário [Fig. III-3]. Seja cadeia $_{\mathcal{A}}-P_1$ (resp. cadeia $_{\mathcal{A}}-P_2$) a cadeia- P_1 (resp. cadeia- P_2) no $\mathcal{A}$ -ésimo pedaço da decomposição de P com $n_1^{\mathcal{A}}$ (resp. $n_2^{\mathcal{A}}$) vértices [Fig. III-4]. Os pedaços *predecessor* e *sucessor* do $\mathcal{A}$ -ésimo pedaço são, respectivamente, o $\text{ind}(\mathcal{A}-1)$ -ésimo e o $\text{ind}(\mathcal{A}+1)$ -ésimo pedaços da decomposição de P [Fig. III-5].

Dado o $\mathcal{A}$ -ésimo pedaço, suponha que cadeia $_{\mathcal{A}}-P_1$ (resp. cadeia $_{\mathcal{A}}-P_2$) é a cadeia externa (resp. interna), isto é, todos os pontos da cadeia $_{\mathcal{A}}-P_2$ são interiores ou estão na fronteira de P_1 . Logo, as distâncias de todos os vértices - se houverem - da cadeia $_{\mathcal{A}}-P_2$ (isto é, da cadeia interna no caso geral) até P_1 são nulas e não precisam ser consideradas no cálculo de $D_{\text{CMMR}}^{(P_1, P_2)}$. Por

FIGURA [III-4]: Polígonos P_1 e P_2 , respectivamente com $n_1 = 6$ e $n_2 = 4$ vértices. A decomposição de P fornece $m = 4$ pedaços, formados pelas seguintes cadeias:

Pedaço 1: cadeia $_1$ - $P_1 = \langle A, v_1^1, v_2^1, B \rangle$ e cadeia $_1$ - $P_2 = \langle A, B \rangle$

Pedaço 2: cadeia $_2$ - $P_1 = \langle B, C \rangle$ e cadeia $_2$ - $P_2 = \langle B, v_2^2, C \rangle$

Pedaço 3: cadeia $_3$ - $P_1 = \langle C, v_3^1, v_4^1, v_5^1, D \rangle$ e cadeia $_3$ - $P_2 = \langle C, v_3^2, D \rangle$

Pedaço 4: cadeia $_4$ - $P_1 = \langle D, v_6^1, A \rangle$ e cadeia $_4$ - $P_2 = \langle D, v_4^2, v_1^2, A \rangle$.

convexidade, tem-se que, para qualquer vértice v_j^1 pertencente à cadeia $_k$ - P_1 , o ponto y_j de P_2 mais próximo de v_j^1 (isto é, $y_j = \text{Proj}_{P_2}(\alpha_j^1, b_j^1)$) está ou em cadeia $_k$ - P_2 (que é uma cadeia interna), ou em cadeia $_{\text{ind}(k-1)}$ - P_2 ou cadeia $_{\text{ind}(k+1)}$ - P_2 (que são cadeias

FIGURA [III-5]: Pedaço 3 e respectivo pedaço predecessor (pedaço 2) e sucessor (pedaço 4).

externas). No caso geral, as contribuições para $D_{\text{CMMR}}(P_1, P_2)$ dos vértices de qualquer das cadeias externas que aparecem na decomposição em pedaços de P podem ser calculadas considerando-se apenas suas distâncias até a cadeia interna do mesmo pedaço e até as cadeias externas do pedaço predecessor e do pedaço sucessor.

Sem perda de generalidade, considera-se o cálculo da distância de qualquer vértice v_j^1 da cadeia externa $\text{cadeia}_k - P_1$ até P_2 (isto é, envolvendo as cadeias $\text{cadeia}_k - P_2$, $\text{cadeia}_{\text{ind}(k-1)} - P_2$ e $\text{cadeia}_{\text{ind}(k+1)} - P_2$). Considere-se primeiramente o cálculo até as cadeias externas $\text{cadeia}_{\text{ind}(k-1)} - P_2$ ou $\text{cadeia}_{\text{ind}(k+1)} - P_2$, por

exemplo, até esta última. Para isto, a cadeia $_{ind(k+1)}-P_2$ pode ser vista como um polígono convexo criando-se uma aresta artificial ligando seu primeiro e último vértices. Novamente, por convexidade, a distância que deseja-se calcular é a mesma distância de v_j^1 até o polígono recém-construído. O mesmo artifício pode ser aplicado à cadeia $_k-P_1$. Como os polígonos

FIGURA [III-6]: Polígonos convexas formados pelas cadeias $_4-P_2$, $_3-P_1$ e $_1-P_1$ para a utilização do algoritmo linear do capítulo II na determinação das distâncias dos vértices v_4^2 e v_1^2 da cadeia externa do pedaço 4 com relação às cadeias externas dos pedaços 3 (predecessor) e 1 (sucessor). As arestas CD e DA são artificiais.

formados a partir de cadeia $_{\mathcal{K}}-P_1$ e de cadeia $_{ind(\mathcal{K}+1)}-P_2$ são disjuntos, exceto pelo último vértice de cadeia $_{\mathcal{K}}-P_1$ e pelo primeiro vértice de cadeia $_{ind(\mathcal{K}+1)}-P_2$ (o qual é a interseção das linhas poligonais associadas com P_1 e P_2 , não sendo portanto necessariamente um vértice de P_1 ou P_2), o algoritmo linear apresentado no capítulo anterior pode agora ser utilizado para calcular em $O(n_1^{\mathcal{K}} + n_2^{ind(\mathcal{K}+1)})$ a distância de todos os vértices da cadeia $_{\mathcal{K}}-P_1$ até cadeia $_{ind(\mathcal{K}+1)}-P_2$ (e não somente a distância de v_j^1 até P_2) [Fig III-6]. O mesmo raciocínio pode ser usado para o cálculo das distâncias de todos os vértices da cadeia $_{\mathcal{K}}-P_1$ à cadeia $_{ind(\mathcal{K}-1)}-P_2$ em $O(n_1^{\mathcal{K}} + n_2^{ind(\mathcal{K}-1)})$.

Com relação ao cálculo da distância de v_j^1 até a cadeia interna cadeia $_{\mathcal{K}}-P_2$, considerar-se-á analogamente o cálculo de todas as distâncias de todos os vértices da cadeia $_{\mathcal{K}}-P_1$ até a cadeia $_{\mathcal{K}}-P_2$. No caso, a idéia é percorrer ambas as cadeias de forma análoga a um **merge** de duas listas ordenadas, calculando-se as distâncias de cada vértice da cadeia $_{\mathcal{K}}-P_1$ à medida em que se encontra o ponto da cadeia $_{\mathcal{K}}-P_2$ que lhe é mais próximo [Fig. III-7]. O que garante o funcionamento deste **merge** é o fato de que, na medida em que se percorre a cadeia $_{\mathcal{K}}-P_1$ monotonicamente, também percorre-se a cadeia $_{\mathcal{K}}-P_2$ monotonicamente. Logo, o cálculo das distâncias de todos os vértices da cadeia $_{\mathcal{K}}-P_1$ até a cadeia $_{\mathcal{K}}-P_2$ é feito em $O(n_1^{\mathcal{K}} + n_2^{\mathcal{K}})$.

3.2. Descrição do Algoritmo

O algoritmo para o cálculo de $D_{\text{CMR}}(P_1, P_2)$ (no caso geral em que os polígonos podem se interceptar) proposto a seguir é composto por um pré-processamento onde, primeiramente, se identifica se a

FIGURA [III-7]: Realiza-se um **merge** quando da determinação das distâncias dos vértices v_4^2 e v_1^2 da cadeia externa ($cadeia_4-P_2$) do pedaço 4 com relação à cadeia interna ($cadeia_4-P_1$) do mesmo pedaço.

interseção dos dois polígonos é ou não vazia. Em caso afirmativo, o próximo passo consiste em se executar o algoritmo descrito no capítulo anterior. Caso contrário, calcula-se a união de P_1 e P_2 , a fim de se obter a decomposição em pedaços dos dois polígonos. Finalmente, para cada cadeia externa de cada pedaço determinado, faz-se o **merge** com a cadeia interna do mesmo pedaço e aplica-se o algoritmo do último capítulo para as cadeias externas (transformadas em polígonos) predecessora e sucessora.

ALGORITMO $D_{\text{CMMR}}^{(P_1, P_2)}$ Caso geral

PASSO 0: (pré-processamento e inicialização)

Calcular $P_1 \cap P_2$ utilizando um algoritmo linear. Se $P_1 \cap P_2 = \emptyset$, executar o algoritmo do capítulo anterior e PARAR. Senão, calcular também $P_1 \cup P_2$, determinando e armazenando em seguida a decomposição em pedaços de P .

Sejam m o número de pedaços na decomposição de P , n_1^k (resp. n_2^k) o número de vértices de P_1 (resp. P_2) na cadeia $_k-P_1$ (resp. cadeia $_k-P_2$) e externa_k (resp. interna_k) a cadeia externa (resp. interna) do pedaço k , $k = 1, 2, \dots, m$.

Fazer $D_{\text{CMMR}} \leftarrow 0$ e $k \leftarrow 1$.

PASSO 1: (processamento principal)

Para cada vértice j de externa_k , calcular suas distâncias $d_j^{\text{ind}(k-1)}$, d_j^k e $d_j^{\text{ind}(k+1)}$, respectivamente até as cadeias $\text{externa}_{\text{ind}(k-1)}$, interna_k e $\text{externa}_{\text{ind}(k+1)}$.

PASSO 2: (atualização de D_{CMMR} e condição de parada)

Fazer $D_{\text{CMMR}} \leftarrow D_{\text{CMMR}} + \sum_{j \in \text{externa}_k} \left[\min(d_j^{\text{ind}(k-1)}, d_j^k, d_j^{\text{ind}(k+1)}) \right]^2$

$k \leftarrow k + 1$. Se $k > m$, PARAR; senão, voltar ao passo 1.

Também para o armazenamento dos pedaços da decomposição utilizou-se uma lista duplamente encadeada e circular. Os elementos desta lista são, por sua vez, listas lineares. O primeiro elemento de cada uma delas corresponde a um ponto pertencente à interseção dos dois polígonos, que pode ter sido determinado por uma aresta de cada um ou ser originalmente um vértice do polígono relativamente mais interno. No primeiro

caso, os demais elementos da lista são os vértices da cadeia externa do pedaço definido pelo ponto de interseção correspondente ao primeiro elemento. Já no segundo caso, não há mais elementos na lista pois, sendo um vértice original de um dos polígonos, este ponto de interseção corresponde a um ponto

FIGURA [III-8]: Lista circular duplamente encadeada utilizada como estrutura de dados para o armazenamento dos pedaços da decomposição da região $P = (P_1 \cup P_2) - \text{Interior}(P_1 \cap P_2)$.

da cadeia interna do pedaço definido por um ponto de interseção anterior.

Seja o exemplo ilustrado na figura [III-8] com relação aos polígonos da figura [III-4]. O pedaço 3 é definido pelo ponto de interseção C (que é determinado pela interseção das arestas (v_2^1, v_3^1) e (v_2^2, v_3^2)), com cadeia externa (cadeia₃-P₁) definida na fronteira de P₁, de C até o ponto de interseção D e contendo os vértices v_3^1 , v_4^1 e v_5^1 e com cadeia interna (cadeia₃-P₂) definida na fronteira de P₂, também de C até D, contendo o vértice v_3^2 .

Esta estrutura de dados é adequada não só para, a partir de qualquer pedaço, se obter seus respectivos pedaços predecessor e sucessor como também para se construir os polígonos convexos a partir das cadeias externas e arestas artificiais. Além disso, é facilitada a execução do *merge* entre as cadeias externa e interna de um mesmo pedaço

TEOREMA 1:

O algoritmo proposto nesta seção tem complexidade $O(n)$, onde $n = \max \{n_1, n_2\}$.

Prova:

Seja n_{ext}^k (resp. n_{int}^k) o número de vértices de externa_k (resp. interna_k). O cálculo da distância de todos os vértices de externa_k até o outro polígono leva $O(n_{ext}^k + n_{int}^k + n_{ext}^{ind(k-1)} + n_{ext}^{ind(k+1)})$. Logo, a complexidade total do algoritmo é

$$\sum_{k=1}^m O(n_{ext}^k + n_{int}^k + n_{ext}^{ind(k-1)} + n_{ext}^{ind(k+1)}) =$$

$$\begin{aligned}
&= \sum_{k=1}^m \alpha(n_{\text{ext}}^k + n_{\text{int}}^k) + \sum_{k=1}^m \alpha(n_{\text{ext}}^{\text{ind}(k-1)}) + \sum_{k=1}^m \alpha(n_{\text{ext}}^{\text{ind}(k+1)}) = \\
&= \alpha(n_1 + n_2) + \alpha(n) + \alpha(n) = \alpha(n). \quad \blacksquare
\end{aligned}$$

4. RESULTADOS COMPUTACIONAIS

O algoritmo descrito na seção anterior foi implementado e seu desempenho comparado com o do algoritmo trivial, também para dez problemas-teste, cada qual composto de polígonos gerados aleatoriamente, desta vez no mesmo quadrado unitário. Os tempos médios obtidos (no mesmo microcomputador IBM PC/AT 12 Mhz com co-processador aritmético 80287 utilizado no caso de polígonos disjuntos) são dados em segundos na tabela seguinte [Fig III-9]:

n	Alg. $\alpha(n_1, n_2)$	Alg. $\alpha(n)$
5	0.252	0.254
10	0.786	0.468
15	1.436	0.615
20	2.591	0.853
25	3.587	0.951
30	5.174	1.121
35	6.532	1.593

FIGURA [III-9]: Tabela com os resultados computacionais obtidos pelos algoritmos $\alpha(n_1, n_2)$ e $\alpha(n_1 + n_2)$ para o caso geral onde em que os polígonos podem se interseparar.

O gráfico resultante dos dados da tabela anterior é apresentado a seguir [Fig. III-10]:

FIGURA [III-10]: Gráfico comparativo do desempenho dos algoritmos $O(n_1 n_2)$ e $O(n_1 + n_2)$ para o caso geral em que os polígonos podem se interceptar.

CAPÍTULO IV

CONCLUSÕES

Foi proposto nesta dissertação um novo algoritmo de complexidade linear para o cálculo da distância entre polígonos convexos, no caso geral em que os polígonos podem se interceptar. Cabe observar que o melhor algoritmo disponível na literatura até o momento tinha complexidade $O(n_1 \log n_2 + n_2 \log n_1)$ e que os resultados computacionais obtidos indicam que, na prática, o novo algoritmo também apresenta um bom desempenho.

Em [COX89], apresentou-se um algoritmo para resolver um problema de superposição ótima de polígonos convexos e observou-se em testes computacionais que mais de 95% do tempo necessário para se obter a solução correspondia a cálculos da distância entre os dois polígonos, feitos por um algoritmo trivial de complexidade $O(n_1 n_2)$. Logo, o algoritmo linear aqui proposto poderá ser utilizado para se obter resultados consideravelmente melhores.

O problema discutido neste trabalho poderá eventualmente ser estendido para o caso tridimensional, desenvolvendo-se um algoritmo de complexidade linear para o cálculo de certas funções distância entre dois poliedros convexos. No caso, o resultado obtido recentemente por Chazelle [CHAZ89] relativo à determinação em tempo linear da interseção de poliedros convexos tridimensionais seria provavelmente de grande utilidade.

Outras extensões possíveis são as apresentadas em [ATAL90], onde são propostos um algoritmo paralelo ótimo (modelo EREW-PRAM) para o cálculo da distância entre polígonos convexos e também um algoritmo de complexidade $O(n_1 n_2 + n_1 \log n_1 + n_2 \log n_2)$ para o caso em que os polígonos não são necessariamente convexos.

REFERÊNCIAS BIBLIOGRÁFICAS

- [ATAL83] M.J. ATALLAH, "A Linear Time Algorithm for the Hausdorff Distance between Convex Polygons", *Information Processing Letters* 17 , 207-209, 1983.
- [ATAL90] M.J. ATALLAH, C.C. RIBEIRO E S. LIFSCHITZ, "Computing Some Distance Functions between Polygons", Les Cahiers du GERAD G-90-24, École des Hautes Études Commerciales, Montreal, Canada, May 1990.
- [CHAZ89] B.M. CHAZELLE, "An Optimal Algorithm for Intersecting Three-Dimensional Convex Polyhedra", *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, Research Triangle, NC, 586-591, 1989.
- [COX89] P. COX, H. MAITRE, M. MINOUX e C.C. RIBEIRO, "Optimal Matching of Convex Polygons", *Pattern Recognition Letters* 9, 327-334, 1989.
- [CHIN83] F. CHIN e C.A. WANG, "Optimal Algorithms for the Intersection and the Minimum Distance Problems between Planar Polygons", *IEEE Transactions on Computers* C-32(12), 1203-1207, 1983.
- [CHIN84] F. CHIN e C.A. WANG, "Minimum Vertex Distance between Separable Convex Polygons", *Information Processing Letters* 18, 41-45, 1984.
- [EDEL85] H. EDELSBRUNNER, "Computing the Extreme Distances between Two Convex Polygons", *Journal of Algorithms* 6, 213-224, 1985.
- [GRUN67] B. GRUNBAUM, *Convex Polytopes*, Wiley, New York, 1967.

- [PREP85] F.P. PREPARATA e M.I. SHAMOS, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.
- [ROUR82] J. O'ROURKE, C.-B. CHIEN, T. OLSON e D. NADDOR, "A New Linear Time Algorithm for Intersecting Convex Polygons", *Computer Graphics and Image Processing* 19, 384-391, 1982.
- [SCHW81] J.T. SCHWARTZ, "Finding the Minimum Distance between Two Convex Polygons", *Information Processing Letters* 13(4), 168-170, 1981.

APÊNDICE 1

GERAÇÃO ALEATÓRIA DE POLÍGONOS CONVEXOS

Para se verificar o desempenho dos algoritmos apresentados nos capítulos anteriores, foi desenvolvido um algoritmo de construção de polígonos convexos, definidos por vértices cujas coordenadas são determinadas por um gerador de números aleatórios.

A idéia básica do algoritmo consiste em, a partir de um polígono de três vértices gerado inicialmente, construir a cada passo um novo polígono convexo com um vértice a mais do que no passo anterior, até que o polígono com o número de vértices desejado seja obtido. O algoritmo é descrito a seguir e sua execução é ilustrada na figura [A1-1]:

ALGORITMO Geração aleatória de polígonos

PASSO 0: (determinação dos três primeiros vértices)

Gerar inicialmente dois pontos distintos do plano. Em seguida, gerar um terceiro ponto que não se encontre sobre a reta passando pelos dois primeiros. Considerar estes pontos como os três primeiros vértices do polígono e fazer $n \leftarrow 4$.

PASSO 1: (determinação do próximo vértice)

Gerar um novo ponto p_n tal que a envoltória convexa do conjunto formado por este ponto e pelos $n-1$ vértices já determinados defina um polígono convexo com n vértices. Considerar p_n como o n -ésimo vértice.

PASSO 2: (condição de parada)

Se n for igual ao número de vértices desejado para o polígono,
PARAR; senão, fazer $n \leftarrow n + 1$ e voltar ao passo 1.

FIGURA [A1-1]: Exemplo de execução do algoritmo de geração aleatória de polígonos convexos. Os índices i ($i = 1, 2, \dots, 8$) dos vértices p_i indicam a ordem em que os vértices foram gerados e os segmentos de reta interiores ao polígono representam arestas dos polígonos convexos construídos nas iterações intermediárias.

No caso da construção de polígonos disjuntos, um dos polígonos é gerado no quadrado unitário com diagonal $((0,0),(1,1))$ e o outro é gerado no quadrado unitário com diagonal $((0,0),(1,-1))$. No

caso geral em que os polígonos podem se interceptar, os vértices dos dois polígonos são gerados no mesmo quadrado unitário.

Foi utilizada uma lista circular duplamente encadeada como estrutura de dados para o armazenamento dos vértices dos polígonos. Desta forma, o acesso a qualquer vértice do polígono e a seus respectivos vértices predecessor e sucessor é facilitado, permitindo o percurso sequencial eficiente tanto em sentido horário como em sentido anti-horário. Na implementação do algoritmo, cada novo vértice gerado foi incluído na lista de forma que, ao final da execução, os vértices do polígono estejam ordenados em sentido anti-horário.

O algoritmo foi utilizado para a geração de polígonos de no mínimo 5 (cinco) e no máximo 35 (trinta e cinco) vértices. Este limite máximo foi determinado pelas dificuldades verificadas durante os testes da implementação do algoritmo. Em certos casos, como no exemplo ilustrado na figura [A1-2], gerar aleatoriamente um novo vértice que verifique a condição do passo 1 pode ser uma tarefa muito difícil. De fato, para ser considerado como um novo vértice do polígono, o novo ponto tem que pertencer às regiões sombreadas indicadas na figura, que representam no total um conjunto de pontos relativamente pequeno comparado ao quadrado unitário onde os pontos são gerados.

Portanto, foi considerado na implementação do algoritmo um número máximo de tentativas para o passo 1, recomeçando-se a execução do algoritmo caso este limite fosse atingido. A construção de polígonos convexos com mais de 35 (trinta e cinco) vértices se mostrou inviável mesmo com esta alteração. A técnica de backtracking, isto é, a retirada do(s) vértice(s) que estaria(m) impedindo a verificação da condição do passo 1, também foi testada como alternativa de solução para o problema, mas os resultados observados não foram melhores do que aqueles obtidos com o reinício da execução do algoritmo.

FIGURA [A1-2]: Situação em que a geração aleatória de um próximo vértice pode ser muito difícil. Para manter a convexidade do polígono, o novo ponto gerado deve pertencer às regiões sombreadas, que representam um conjunto de pontos pequeno com relação ao quadrado unitário.

APÊNDICE 2

DETERMINAÇÃO DOS PONTOS INTERIORES A UM POLÍGONO CONVEXO

O algoritmo trivial descrito no capítulo III - quando os polígonos podem se interceptar - utiliza um pré-processamento com o objetivo de identificar os vértices de P_1 que são interiores a P_2 e os vértices de P_2 que são interiores a P_1 . Estes vértices não contribuem com distância positiva no cálculo de $D_{\text{CMR}}^{(P_1, P_2)}$ e podem ser ignorados durante a execução do algoritmo.

Este pré-processamento é baseado em um algoritmo de complexidade $O(\log n)$ ($O(n)$ de pré-processamento) proposto em [PREP85], que verifica se um dado ponto do plano é ou não interior a um polígono convexo de n vértices. Este algoritmo é descrito a seguir. Posteriormente, será apresentado o algoritmo que determina os vértices de P_1 que são interiores a P_2 . Antes porém, apresenta-se as seguintes definições:

Definição A2-1: O centróide de um conjunto finito de pontos do plano $\{p_1, p_2, \dots, p_n\}$ é o ponto definido pela média aritmética das coordenadas de $p_1, p_2, \dots, p_n$.

Se $n \geq 3$ e $p_1, p_2, \dots, p_n$ formam um polígono convexo, tem-se que o centróide é interior a este polígono.

Definição A2-2: O ângulo (p_1, p_2, p_3) é dito um ângulo anti-horário (resp. ângulo horário) se os pontos p_1, p_2 e p_3 , nesta ordem, formam um ciclo anti-horário (resp. ciclo horário).

Uma das formas de se determinar se o ângulo $(p_1 p_2 p_3)$ é um ângulo anti-horário ou horário consiste em se verificar o produto vetorial de $\overrightarrow{p_1 p_2}$ por $\overrightarrow{p_1 p_3}$: se positivo, é um ângulo anti-horário; se negativo, um ângulo horário.

O algoritmo a ser descrito aproveita a propriedade de que os vértices de um polígono convexo são ordenados angularmente com relação a qualquer ponto interior. Tomando-se, por exemplo, o centróide do triângulo formado por três dos vértices do polígono, os raios originados neste ponto até cada vértice particionam o plano em n cunhas. Cada cunha é dividida em duas

FIGURA [A2-11]: Os raios originados no centróide q do polígono definem o plano em $n = 8$ cunhas. O ângulo $(p_1 x_1 p_2)$ é um ângulo anti-horário e, portanto, x_1 é externo ao polígono. Já x_2 é um ponto interno pois o ângulo $(p_2 x_2 p_3)$ é um ângulo horário.

regiões distintas por uma aresta do polígono: uma delas totalmente interna e a outra totalmente externa ao polígono. Determinando-se a cunha - e consequentemente os vértices que a definem - à qual um ponto qualquer do plano pertence, uma simples verificação da posição relativa deste ponto (em cuja região da cunha ele está) possibilita saber se o ponto é interior ou não ao polígono [Fig. A2-1].

Mais especificamente, tomando-se o centróide como a origem em coordenadas polares, a determinação da cunha a qual um ponto qualquer do plano pertence pode ser feita por pesquisa binária, já que os raios da origem até os vértices do polígono ocorrem em ordem angular. Este fato garante a complexidade logarítmica do algoritmo.

ALGORITMO [PREP85]

PASSO 0: (pré-processamento)

Determinar o centróide q de três vértices de P e armazenar os vértices $p_1, p_2, \dots, p_n$ de P em uma estrutura de dados que permita uma pesquisa binária eficiente.

PASSO 1: (determinação da cunha)

Determinar por pesquisa binária em qual cunha um dado ponto x está localizado. O ponto x está localizado entre os raios definidos por p_i e p_{i+1} se e somente se o ângulo $(x \ q \ p_{i+1})$ é um ângulo horário e o ângulo $(x \ q \ p_i)$ é um ângulo anti-horário.

PASSO 2: (verificação se o ponto é interior ou não)

Sejam p_i e p_{i+1} os vértices que definem a cunha determinada. O ponto x é interior ao polígono P se e somente se o ângulo $(p_i p_{i+1} x)$ é um ângulo anti-horário.

Com relação ao passo 0, um vetor é a estrutura de dados adequada para uma pesquisa binária. Já o passo 1 sugere que se utilize a verificação de ângulo horário ou anti-horário na determinação da cunha onde o ponto x está localizado. Porém, esta técnica não pode ser aplicada para se determinar, a cada iteração da pesquisa binária, em qual metade do conjunto de vértices de P ($p_1, p_2, \dots, p_n$ inicialmente) se procederá a nova iteração.

Logo, algumas alterações tiveram que ser feitas neste algoritmo mantendo-se porém sua idéia original e, principalmente, sua complexidade. O algoritmo que verifica quais são os vértices do polígono P_1 interiores ao polígono P_2 descrito a seguir utiliza o algoritmo [PREP85] já com as modificações necessárias efetuadas. A determinação dos vértices de P_2 externos a P_1 é análoga e portanto omitida.

ALGORITMO Vértices de P_1 externos a P_2

PASSO 0: (pré-processamento)

Determinar o centróide q de três vértices de P_2 e calcular os ângulos formados por cada raio qv_i^2 , $i \in \{2, \dots, n_2\}$, com o eixo qv_1^2 tomado como referência, e armazenar esses ângulos em um vetor.

PASSO 1: (inicialização)

Fazer $j \leftarrow 1$ e inicializar a lista *Externos* como vazia.

PASSO 2: (determinação da cunha onde se localiza v_j^1)

Determinar o ângulo formado pelo raio qv_j^1 com o eixo qv_1^2 . Em seguida, determinar por pesquisa binária a qual cunha pertence o vértice v_j^1 . Sejam v_i^2 e v_{i+1}^2 os vértices de P_2 que definem a cunha determinada.

PASSO 3: (verificar se v_j^1 é ou não externo e condição de parada)

Se o ângulo $(v_i^2, v_j^1, v_{i+1}^2)$ é um ângulo anti-horário, então incluir

v_j^1 na lista *Externos*. Se $j = n_1$, PARAR; senão, fazer $j \leftarrow j + 1$ e voltar ao passo 2.

O passo 0 do algoritmo tem complexidade $O(n_2)$ e o passo 2 $O(\log n_2)$. Para as n_1 iterações do passo 3, a complexidade total, quando da determinação dos vértices de P_1 externos a P_2 , é $O(n_1 \log n_2)$ ($O(n_2)$ de pré-processamento) e $O(n_2 \log n_1)$ ($O(n_1)$ de pré-processamento) quando invertem-se os papéis dos polígonos.

Por construção, os ângulos calculados no passo 0 têm valores crescentes e podem ser diretamente armazenados em um vetor (ou em uma estrutura de dados que permita pesquisa binária). Um vetor somente com os vértices de P_2 , como no algoritmo anterior, não pode ser utilizado na determinação da cunha por pesquisa binária, a não ser que os vértices estejam associados a informações adicionais como, por exemplo, seus ângulos com relação a um determinado eixo.

A estrutura de dados utilizada para a lista *Externos* é idêntica à estrutura de dados utilizada para a representação dos polígonos em todo este trabalho, isto é, uma lista circular duplamente encadeada para os vértices de P_1 externos a P_2 e outra para os vértices de P_2 externos a P_1 .

APÊNDICE 3

ALGORITMO LINEAR PARA A DETERMINAÇÃO DA INTERSEÇÃO ENTRE DOIS POLÍGONOS CONVEXOS

O algoritmo linear apresentado no capítulo III para o cálculo de $D_{\text{CMMR}}^{(P_1, P_2)}$ tem como componente principal a construção da região $P = (P_1 \cup P_2) - \text{interior}(P_1 \cap P_2)$. Usando-se o algoritmo proposto em [ROUR82], tanto a interseção como a união de dois polígonos convexos podem ser calculadas em tempo linear e, conseqüentemente, a região P também.

A idéia básica do algoritmo consiste em, no cálculo da interseção de P_1 e P_2 , fazer dois ponteiros especiais - cada qual apontando para uma aresta de cada polígono - percorrer sequencialmente os dois polígonos à procura de pontos de interseção. Todos os pontos de interseção existentes podem ser determinados, no pior caso, em duas voltas ao redor dos polígonos. No caso específico do cálculo de $P_1 \cap P_2$, a complexidade do algoritmo é dada por $O(n_1 + n_2)$. O algoritmo para a determinação da união de P_1 e P_2 é análogo e apenas pequenas adaptações são necessárias.

Cada polígono é representado pelas coordenadas de seus vértices, representadas em duas listas circulares duplamente encadeadas com orientação anti-horária, isto é, ao percorrer-se sequencialmente as arestas dos polígonos representadas pelos vértices que a definem, o interior dos polígonos estará sempre à esquerda das mesmas. O duplo encadeamento das listas não é aqui necessário, porém foi utilizado para se manter a mesma estrutura de dados utilizada no restante desta dissertação. Antes de se descrever o algoritmo, apresenta-se as seguintes notações e definições:

FIGURA [A3-1]: Vértice v_1 e respectivo vértice predecessor (v_{1-}) e sucessor (v_{1+}). A região hachurada indica o semi-plano definido pelo vetor $\dot{v}_1 = \overrightarrow{v_{1-}v_1}$.

Para cada vértice v_1 (resp. v_2) de P_1 (resp. P_2), representa-se por v_{1+} (resp. v_{2+}) seu vértice posterior e por v_{1-} (resp. v_{2-}) seu vértice anterior. Representa-se também por $\dot{v}_1$ (resp. $\dot{v}_2$), o vetor $\overrightarrow{v_{1-}v_1}$ (resp. $\overrightarrow{v_{2-}v_2}$) e por $\partial\pi(\dot{v}_1)$ o semi-plano definido por $\dot{v}_1$ que contém os pontos do plano à esquerda ou sobre a reta passando por v_{1-} e v_1 , isto é, $\partial\pi(\dot{v}_1) = \{x \in \mathbb{R}^2 \mid \dot{v}_1 \times (x - v_{1-}) \geq 0\}$. O semi-plano $\partial\pi(\dot{v}_2)$ é definido de forma análoga [Fig. A3-1].

O algoritmo proposto em [ROUR82] e descrito a seguir determina inicialmente um ponto de interseção gerado por uma aresta de

cada polígono (se não existir um ponto com esta característica significa que ou os polígonos são disjuntos ou um dos polígonos contém o outro). A partir deste ponto, os dois polígonos são percorridos simultaneamente, procurando-se a cada iteração um

FIGURA [A3-2]: Exemplo de execução do algoritmo. As arestas de P_1 e P_2 escolhidas inicialmente estão indicadas por v_1 inicial e v_2 inicial e os números associados às arestas indicam a iteração em que estas foram percorridas. Ao final, a lista *Interseção* conterá os vértices marcados com um círculo. Os pontos de interseção i_1 , i_2 , i_3 , i_4 , i_5 e i_6 são aqueles definidos por duas arestas de cada polígono e os demais são vértices originais de P_1 ou P_2 .

novo ponto de interseção, dado ou por nova interseção de arestas dos polígonos ou por um vértice do polígono relativamente mais interno. Obtém-se ao final da execução do algoritmo uma nova lista circular duplamente encadeada com os vértices do polígono interseção. Um exemplo de execução do algoritmo é ilustrado na figura [A3-2].

ALGORITMO: [ROUR82]

PASSO 0: (inicialização)

Escolher $v_1 \in P_1$ e $v_2 \in P_2$ arbitrários, inicializar a lista *Interseção* como vazia e fazer *contador* $\leftarrow 0$

PASSO 1: (determinação dos pontos de interseção)

Se ($\dot{v}_1$ e $\dot{v}_2$ se interceptam) E (este ponto de interseção é igual ao primeiro ponto de interseção determinado)

então ir para o passo 3

senão Incluir este ponto de interseção na lista *Interseção* e

Se ($v_1 \in op(\dot{v}_2)$)

então P_1 é o polígono mais interno

senão P_2 é o polígono mais interno

Fim-Se

Fim-Se

Se (($\dot{v}_2 \times \dot{v}_1 \geq 0$) E ($v_1 \in op(\dot{v}_2)$)) OU

(($\dot{v}_2 \times \dot{v}_1 < 0$) E ($v_2 \in op(\dot{v}_1)$)))

então Se (P_2 é o polígono mais interno)

então Incluir v_2 na lista *Interseção*

Fim-Se e

fazer $v_2 \leftarrow v_{2+}$

senão Se (P_1 é o polígono mais interno)

então Incluir v_1 na lista *Interseção*

Fim-Se e

fazer $v_1 \leftarrow v_{1+}$

Fim-Se

PASSO 2: (condição de parada)

Se ($\text{contador} \leq 2 \cdot (|P_1| + |P_2|) \cdot 2$)

então fazer $\text{contador} \leftarrow \text{contador} + 1$ e voltar ao passo 1

senão PARAR

Fim-Se

PASSO 3: (verificação do resultado)

Se (a lista *Interseção* não estiver vazia)

então existe interseção entre P_1 e P_2 e os vértices do polígono interseção estão na lista *Interseções*

senão Escolha $v_1 \in P_1$ e $v_2 \in P_2$ arbitrários

Se ($v_1 \in P_2$)

então $P_1 \subset P_2$: P_1 é o polígono interseção

senão Se ($v_2 \in P_1$)

então $P_2 \subset P_1$: P_2 é o polígono interseção

senão P_1 e P_2 são disjuntos

Fim-Se

Fim-Se

Fim-Se

Este algoritmo não prevê a ocorrência de pontos de interseção degenerados, em que um vértice de um polígono coincide com vértice ou pertence a uma aresta do outro polígono ou ainda, quando existirem arestas colineares [Fig A3-3]. Estes casos podem ser solucionados com as seguintes alterações no algoritmo:

- (a) Não se considerar v_1 e v_2 se interceptando quando forem colineares.
- (b) Modificando-se a condição de suspensão da execução do algoritmo (passo 1) para:

Se (o primeiro ponto não foi encontrado na iteração anterior)

E (este ponto de interseção é igual ao primeiro ponto de interseção determinado)

FIGURA [A3-3]: Casos degenerados:

- (a) Um vértice de P_1 coincide com uma aresta de P_2 .
- (b) P_1 e P_2 têm um vértice em comum.
- (c) P_1 e P_2 têm arestas colineares.

então ir para o passo 3

No caso da determinação da união de P_1 e P_2 , as seguintes adaptações devem ser feitas:

- (a) No passo 1, manter controle do polígono mais externo e não do mais interno.
- (b) No passo 3, se $P_1 \subset P_2$ tem-se P_2 como polígono união e, caso $P_1 \supset P_2$, P_1 como polígono união.

A representação dos pedaços (região P) na estrutura de dados

descrita na seção 3.2 do capítulo III é facilmente obtida ao se percorrer simultaneamente as listas *União* e *Interseção* determinadas pelo algoritmo descrito neste apêndice.

UM ALGORITMO LINEAR PARA O CÁLCULO DE ALGUMAS
FUNÇÕES DISTÂNCIA ENTRE POLÍGONOS CONVEXOS

Dissertação de Mestrado apresentada por SÉRGIO LIFSCHITZ em 9 de agosto de 1990 ao Departamento de Engenharia Elétrica da PUC/RJ, e aprovada pela Comissão Julgadora, formada pelos seguintes professores:

PROF. REINALDO CASTRO SOUZA (ORIENTADOR)
DEE-PUC/RJ

PROF. ANTONIO ALBERTO FERNANDES DE OLIVEIRA
COPPE-UFRJ

PROF. MARCELO GATTASS
DEC-PUC/RJ

PROF. NELSON MACULAN FILHO
COPPE-UFRJ

PROF. RONALDO CESAR MARINHO PERSIANO
COPPE-UFRJ

Vista e permitida a impressão
Rio de Janeiro, 24/08/90

PROFa. BELITA KOILLER

Coordenadora dos Programas de Pós-Graduação do
CENTRO TÉCNICO CIENTÍFICO