Este exemplar corresponde à redação final da Tese/Dissertação devidamente corrigida e defendida por: FABIO DALLA COSTA FIGUEIREDO
e aprovada pela Banca Examinadora.
Campinas, 11 de ABRIL de 2006
COORDENADOR DE PÓS-GRADUAÇÃO
CPG-IC

Alguns Problemas em Geometrias de Curvas

Fábio Dalla Costa Figueiredo

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

Alguns Problemas em Geometrias de Curvas

Fábio Dalla Costa Figueiredo

Dezembro de 2005

Banca Examinadora:

- Pedro Jussieu de Rezende (Orientador)
- Helena Cristina da Gama Leitão
  Instituto de Computação - Universidade Federal Fluminense
- Siome Klein Goldenstein
  Instituto de Computação - Universidade Estadual de Campinas
- Cid Carvalho de Souza (Suplente)
  Instituto de Computação - Universidade Estadual de Campinas

UNIDADE BE
Nº CHAMADA +UNICAMP
F469a
V EX
TOMBO BC/ 68315
PROC. 16.123.06
C D
PREÇO 11,00
DATA 10/5/06
BIBID - 379308

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecário: Maria Júlia Milani Rodrigues CRB8a / 2116

Figueiredo, Fábio Dalla Costa
F469a Alguns problemas em geometrias de curvas / Fábio Dalla Costa Figueiredo -- Campinas, [S.P. :s.n.], 2005.

Orientador : Pedro Jussieu de Rezende
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Geometria computacional. 2. Geometria não-euclideana. 3. Algoritmos. I. Rezende, Pedro Jussieu de. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Título em inglês: Various problems in geometries of curves

Palavras-chave em inglês (Keywords): 1. Computational geometry. 2. Non-euclidean geometry. 3. Algorithms.

Área de concentração: Teoria da Computação

Titulação: Mestre em Ciência da Computação

Banca examinadora: Prof. Dr. Pedro Jussieu de Rezende (IC-UNICAMP)
Profa. Dra. Helena Cristina da Gama Leitão (IC-UFF)
Prof. Dr. Siome Klein Goldenstein (IC-UNICAMP)

Data da defesa: 28/11/2005

Alguns Problemas em Geometrias de Curvas

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Fábio Dalla Costa Figueiredo e aprovada pela Banca Examinadora.

Campinas, 2 de dezembro de 2005.

Pedro Jussieu de Rezende (Orientador)

200609412

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

TERMO DE APROVAÇÃO

Tese defendida e aprovada em 28 de novembro de 2005, pela Banca examinadora composta pelos Professores Doutores:

Profa. Dra. Helena Cristina da Gama Leitão
IC – UFF

Prof. Dr. Siome Klein Goldenstein
IC - UNICAMP

Prof. Dr. Pedro Jussieu de Rezende
IC - UNICAMP

© Fábio Dalla Costa Figueiredo, 2005.
Todos os direitos reservados.

Resumo

Problemas de natureza geométrica são encontrados em diversas áreas e, portanto, a análise dos mesmos sob uma ótica algorítmica é imprescindível. Não obstante um amplo tratamento de problemas na geometria euclidiana, relativamente poucos estudos foram feitos em outras geometrias.

Em particular, tomando-se como retas sobre o plano as curvas de uma família $\mathcal{F}$ que satisfaçam um pequeno conjunto de propriedades, pode-se definir uma geometria de curvas, denotada por $\mathcal{G}_{\mathcal{F}}$, a qual foi inicialmente estudada, sob o ponto de vista algorítmico, por Harada [Har00].

Nesta dissertação, estudamos características de famílias de curvas que podem formar uma geometria $\mathcal{G}_{\mathcal{F}}$, e sobre ela, primitivas e algoritmos para soluções de problemas. Desenvolvemos ainda um visualizador gráfico, denominado `GFViewer`, através do qual é possível aprimorar a intuição quanto à forma de construções geométricas, como $\mathcal{G}_{\mathcal{F}}$-retas, $\mathcal{G}_{\mathcal{F}}$-segmentos, $\mathcal{G}_{\mathcal{F}}$-polígonos, $\mathcal{G}_{\mathcal{F}}$-bissetores, $\mathcal{G}_{\mathcal{F}}$-circunferências, etc. Esse visualizador foi utilizado para testarmos a implementação de alguns algoritmos geométricos sobre certas instâncias de $\mathcal{G}_{\mathcal{F}}$.

Com a caracterização de algumas famílias de curvas, foi possível construir novos exemplos dessas geometrias. Além disso, na análise dos problemas formulados, verificamos ser plausível a adaptação de algoritmos existentes no caso euclidiano, devido à correlação entre as duas geometrias, de diversas primitivas e predicados.

Abstract

Problems of geometric nature are found in many areas, so their study under an algorithmic point of view is indispensable. Even though a vast literature exists on problems for the Euclidian geometry, relatively little has been done on other geometries.

In particular, if one takes as lines on the plane the curves of a family $\mathcal{F}$ that meet a small set of conditions, one may define a geometry of curves, denoted by $\mathcal{G}_{\mathcal{F}}$, which was first studied, under an algorithmic approach, by Harada [Har00].

In this dissertation, we identify characteristics of families of curves that may form a $\mathcal{G}_{\mathcal{F}}$ geometry, over which we study primitives and algorithms for the solution of geometric problems. We also developed a graphical visualizer, known as `GFViewer`, through which it is possible to improve the intuition towards the understanding of $\mathcal{G}_{\mathcal{F}}$-lines, $\mathcal{G}_{\mathcal{F}}$-segments, $\mathcal{G}_{\mathcal{F}}$-poligons, $\mathcal{G}_{\mathcal{F}}$-bissectors, $\mathcal{G}_{\mathcal{F}}$-circles, etc. This visualizer was used to test the implementation of a few geometric algorithms over certain instances of $\mathcal{G}_{\mathcal{F}}$.

With the characterization of some families of curves, it was possible to build new examples of these geometries. Furthermore, in the analysis of the problems studied, we perceived that it is plausible to adapt algorithms that deal with the euclidian case, due to the correlation between the two geometries, of several primitives and predicates.

Agradecimentos

Aos amigos, professores e funcionários do Instituto de Computação, cada qual com a sua contribuição para que esse trabalho fosse concluído.

Ao amigo Flávio pela ajuda na implementação do visualizador.

A meu orientador Pedro J. de Rezende, pela paciência, dedicação e pelos conhecimentos compartilhados ao longo da jornada.

A minha futura esposa Samira, pelo amor e carinho sempre, apoio e palavras de incentivo nas horas mais difíceis, e por não me deixar desistir nunca! Te amo.

Aos meus pais Ivam e Eliza, e a meus irmãos Mateus e Samuel, por estarem sempre ao meu lado, incondicionalmente, durante toda a vida. Amo todos vocês.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), ao Instituto de Computação e a meu pai pelo suporte financeiro, sem o qual esse trabalho não seria possível.

Sumário

Lista de Figuras

Capítulo 1

Introdução

Restringindo-se a história da Geometria Computacional ao período após o advento do computador, já que de outro modo deveríamos voltar a Euclides cujas demonstrações eram, de fato, algoritmos, podemos dizer que seu estudo teve início na tese de doutorado de Michael Shamos em meados da década de 70.

Entretanto, no que diz respeito a aplicações, é fácil perceber que problemas geométricos aparecem por toda parte. Focando-se especialmente nas ciências, há aplicações em arqueologia (re-arranjos de fragmentos), astronomia (identificação automática de constelações), biologia (identificação de clusters em culturas), cristalografia, física, geologia (simulação de superfícies de lençóis subterrâneos), projetos de circuitos (wiring), localização de facilidades, química (visualização de modelos moleculares), robótica (otimização de caminhos), etc. É claro que na própria área de computação, diretamente ou em suas aplicações, a freqüência com que ocorre a necessidade de algoritmos eficientes para problemas geométricos é ainda mais notável, especialmente em computação gráfica, processamento de imagens, reconhecimento de padrões, VLSI, etc.

Geometria computacional é, portanto, a área de computação e de geometria em que se estudam propriedades e técnicas de projeto de algoritmos com o objetivo de se obter soluções eficientes para problemas de natureza geométrica, assim como o estabelecimento de quotas inferiores para esses problemas.

Nos cerca de 30 anos em que esta área vem se desenvolvendo, seu destaque como campo fundamental de estudo foi crescendo e hoje já há pelo menos seis revistas científicas exclusivamente dedicadas à publicação de trabalhos nessa área e cerca de dez congressos internacionais integralmente de geometria computacional ou com pelo menos uma trilha em teoria ou aplicações do tema, além de diversos workshops regionais.

A tese de Shamos [Sha78] assim como seus trabalhos iniciais com Hoey [SH75, SH76] e, posteriormente, o livro clássico de Preparata e Shamos [PS85], trataram de modo exclusivo a geometria euclidiana.

Na década de 80 surgiram alguns artigos em métricas não euclidianas como L_1 e L_∞ — por suas aplicações a projeto de circuitos — como trabalhos de Lee [Lee80], Mitchell [Mit87], Rezende et al [dRLW89], Wu et al [WWSW87], dentre outros. Estes foram seguidos mais tarde por trabalhos em métricas de múltiplas direções fixas (Rawlings [Raw87], Wood [RW91] nos anos 80 e, mais recentemente, Harada [Har00]) e por estudos de problemas sob métricas com pesos (Aurenhammer [Aur87], Mitchell e Papadimitriou [MP91], Pinto e Rezende [PdR00], e Guerra [Gue98]). Além destes trabalhos, houve avanço no estudo de técnicas para abordagem de problemas definidos sobre o plano projetivo orientado (Stolfi [Sto91] e, mais recentemente, Gon et al [dRG96], Pinto et al [PdR00], Westrupp et al [dRW99] e Oliveira, Selmi Dei e Rezende [OdRS05]).

Contudo, pouco se fez no estudo de geometrias não euclidianas sobre espaços hiperbólicos, elípticos e sobre geometrias de curvas sobre o plano cartesiano.

Em sua tese de doutorado, Harada [Har00] tomou os trabalhos de Peixoto [Pei49] e Drandell [Dra52] onde são definidas geometrias de curvas sobre o plano — que consistem em uma generalização da geometria euclidiana em duas dimensões — e formulou alguns problemas computacionais cujas soluções eficientes Harada apresenta em sua tese.

Se $\mathcal{F}$ é uma família de curvas que satisfaz as propriedades descritas por Drandell [Dra52], denota-se por $\mathcal{G}_\mathcal{F}$ a geometria de curvas correspondente. Em $\mathcal{G}_\mathcal{F}$ as retas são as curvas da família $\mathcal{F}$ e os pontos são os de $\mathbb{R}^2$.

Nesta dissertação, são estudados outros problemas geométricos sobre a geometria $\mathcal{G}_\mathcal{F}$, assim como uma caracterização de algumas famílias de curvas que podem formar uma geometria $\mathcal{G}_\mathcal{F}$ e o comportamento dos predicados e primitivas geométricas sobre essa geometria. Além disso, para facilitar a percepção do comportamento das curvas dessa geometria, foi desenvolvido um visualizador, denominado `GFViewer`.

No capítulo 2 é feita, primeiramente, uma introdução à geometria $\mathcal{G}_\mathcal{F}$. A seguir é apresentada uma caracterização de algumas famílias de curvas que satisfazem as condições necessárias para se formar uma geometria $\mathcal{G}_\mathcal{F}$. Neste capítulo, também é feita uma análise, baseada na geometria $\mathcal{G}_\mathcal{F}$, dos predicados e primitivas recorrentes em algoritmos geométricos. Por fim, são dadas algumas definições necessárias para a apresentação dos algoritmos propostos no capítulo seguinte.

O capítulo 3 mostra diversos problemas geométricos, para cada um dos quais é apresentado um ou mais tratamentos propostos para a geometria euclidiana e, então, são propostos algoritmos para a solução dos mesmos sobre $\mathcal{G}_\mathcal{F}$.

O capítulo 4 apresenta o visualizador `GFViewer`. São mostradas suas principais funcionalidadese e sua interface, além de uma breve discussão sobre algumas decisões importantes do projeto.

No capítulo 5 são apresentadas as conclusões deste trabalho e discutidos possíveis extensões do mesmo. Além destes capítulos, esta dissertação ainda contém três apêndices.

No apêndice A é apresentada uma discussão sobre como cada fator que define a geometria $\mathcal{G}_{\mathcal{F}}$ influencia a análise de complexidade de algoritmos para problemas geométricos sobre a mesma. O apêndice B discute a dificuldade do cálculo de distâncias sobre uma geometria $\mathcal{G}_{\mathcal{F}}$. Por fim, o apêndice C contém uma tabela de símbolos que são usados nesta dissertação.

Capítulo 2

A Geometria $\mathcal{G}_\mathcal{F}$

Neste capítulo, faremos uma introdução à geometria $\mathcal{G}_\mathcal{F}$, primeiramente tratada algoritmicamente por Harada [Har00], que obteve resultados computacionais no estudos de alguns problemas nessa geometria.

Na seção 2.1, será feita a descrição da geometria e a apresentação do conceito de convexidade sobre a mesma. Na seção 2.2, são caracterizadas famílias de curvas que podem formar uma geometria $\mathcal{G}_\mathcal{F}$. A seção 2.3 mostra algumas definições de relações de precedência essenciais para o funcionamento de alguns algoritmos sobre a $\mathcal{G}_\mathcal{F}$. Por fim, são discutidas a existência e a verificação de predicados e primitivas básicas para algoritmos geométricos na geometria $\mathcal{G}_\mathcal{F}$, na seção 2.4.

2.1 Geometria $\mathcal{G}_\mathcal{F}$ e $\mathcal{G}_\mathcal{F}$-convexidade

Esta seção apresenta a definição da geometria $\mathcal{G}_\mathcal{F}$, assim como o conceito de convexidade sobre a mesma.

A geometria $\mathcal{G}_\mathcal{F}$ pode ser caracterizada através de uma família de curvas, a qual Drandell [Dra52] estudou como uma extensão do trabalho de Peixoto [Pei49]. Antes de mais nada, considere o plano $\mathbb{R}^2$ estendido por um ponto no infinito, o qual pode, assim, ser identificado com a esfera de Riemann, aqui denotada por $\mathcal{R}^2$. Por projeção estereográfica, podemos considerar que o ponto no infinito corresponde ao pólo norte de $\mathcal{R}^2$ (figura 2.1).

Seja $\mathcal{F}$ uma família de curvas diferenciáveis por partes na esfera de Riemann, que satisfaça as seguintes propriedades:

1. Toda curva $C \in \mathcal{F}$ é uma curva de Jordan[1] que passa pelo pólo norte da esfera de Riemann.

[1] Uma curva de Jordan é uma curva fechada simples

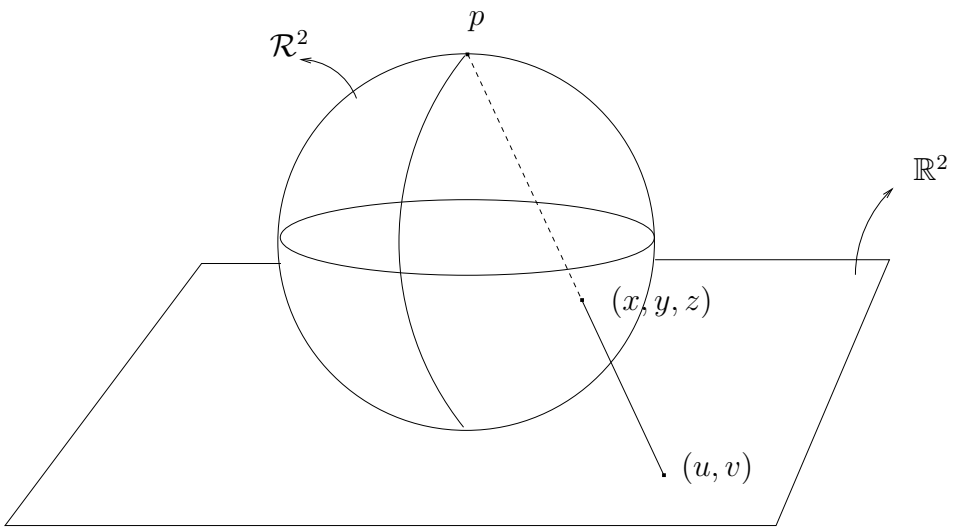


Figura 2.1: Projeção estereográfica de $\mathbb{R}^2$ em na esfera de Riemann ($\mathcal{R}^2$). Um ponto (u, v) em $\mathbb{R}^2$ é mapeado em um ponto (x, y, z) na esfera. O ponto p, pólo norte da esfera, é o ponto no infinito.

2. Dados dois pontos quaisquer no plano, existe apenas uma curva em $\mathcal{F}$ que passa por eles.

3. Existe um limitante superior para o número de partes diferenciáveis das curvas de $\mathcal{F}$.

A partir do item 2 podemos inferir facilmente que, dadas duas curvas distintas, elas podem se interceptar em apenas um ponto. Caso contrário, teríamos duas curvas distintas passando pelos mesmos dois pontos, que seriam os pontos de interseção.

É óbvio que as retas da geometria euclidiana satisfazem a essas condições.

Um exemplo não trivial de uma família de curvas que satisfaz essas condições são as curvas descritas pelas funções $f(x) = ax^2 + bx + c$ para um valor de a fixo, juntamente com as retas verticais, como ilustrado na figura 2.2.

Definição 2.1.1 *Uma* geometria $\mathcal{G}_\mathcal{F}$ *é formada pelo conjunto de pontos de* $\mathcal{R}^2$ *e por um conjunto de curvas* $\mathcal{F}$ *que satisfaça as três condições acima e possui as mesmas regras de incidência da geometria euclidiana.*

Serão utilizados os termos $\mathcal{G}_\mathcal{F}$*-reta* para se referir a uma curva de $\mathcal{F}$, $\mathcal{G}_\mathcal{F}$*-segmento* para designar um arco de uma $\mathcal{G}_\mathcal{F}$-reta com seus extremos em $\mathbb{R}^2$, e $\mathcal{G}_\mathcal{F}$*-semi-reta* ou $\mathcal{G}_\mathcal{F}$*-raio* para designar um arco de uma $\mathcal{G}_\mathcal{F}$-reta que possua um de seus extremos no pólo norte

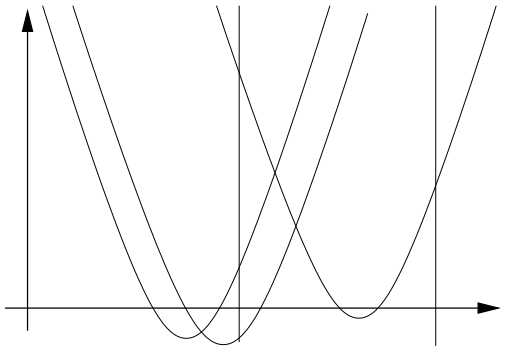


Figura 2.2: Exemplo de uma família de curvas $\mathcal{F}$. Parábolas com uma mesma abertura ($f(x) = ax^2 + bx + c$ com a fixo) e retas verticais.

de $\mathcal{R}^2$. Pontos sobre uma $\mathcal{G}_\mathcal{F}$-reta são ditos $\mathcal{G}_\mathcal{F}$-*colineares*. A notação $d_\mathcal{F}(p_1, p_2)$ representa a distância entre os pontos p_1 e p_2 na geometria $\mathcal{G}_\mathcal{F}$, que é dada pelo comprimento do $\mathcal{G}_\mathcal{F}$-segmento $\widetilde{p_1p_2}$. Denomina-se $\mathcal{G}_\mathcal{F}$-*circunferência* ou $\mathcal{G}_\mathcal{F}$-*círculo* o lugar geométrico dos pontos p de $\mathbb{R}^2$ localizados a uma distância constante $d_\mathcal{F}(p, c)$ de um ponto fixo c, denominado *centro* da $\mathcal{G}_\mathcal{F}$-circunferência.

A partir da definição da geometria $\mathcal{G}_\mathcal{F}$, Harada definiu o conceito de convexidade como:

Definição 2.1.2 *[Har00] Um conjunto de* $\mathcal{R}^2$ *é* $\mathcal{G}_\mathcal{F}$-convexo *se a interseção de qualquer* $\mathcal{G}_\mathcal{F}$*-reta com o mesmo é conexa (ou vazia).*

Essa definição é uma generalização da definição de convexidade da geometria euclidiana, uma vez que se tomarmos como família de curvas $\mathcal{F}$ as retas euclidianas, $\mathcal{G}_\mathcal{F}$ é a geometria euclidiana.

No contexto desta nova geometria, Harada desenvolveu algoritmos para a construção da envoltória convexa e do diagrama de Voronoi abstrato para um conjunto de pontos na geometria $\mathcal{G}_\mathcal{F}$.

2.2 Caracterização das Famílias de Curvas $\mathcal{F}$

Nesta seção, será feita uma análise sobre famílias de curvas que podem satisfazer as três condições que definem uma geometria $\mathcal{G}_\mathcal{F}$. De agora em diante, um conjunto de curvas que satisfaça as condições será denominada simplesmente de *família* $\mathcal{F}$.

Vamos examinar primeiro o exemplo da família de curvas descritas pelas funções $f(x) = ax^2 + bx + c$ para um valor de a fixo, juntamente com as retas verticais, dado na seção anterior. As parábolas, as verticais e as translações delas são curvas que satisfazem

a primeira e terceira condições exigidas pela definição 2.1.1. Assim, para provarmos que esta família de curvas forma uma geometria $\mathcal{G}_\mathcal{F}$ precisamos mostrar que, dados dois pontos quaisquer no plano, existirá apenas uma curva da família passando por eles. O sistema para acharmos a parábola que passa por dois pontos $p_1 = (p_1.x, p_1.y)$ e $p_2 = (p_2.x, p_2.y)$ de $\mathbb{R}^2$ é dado por:

$$\begin{cases} a * p_1.x^2 + b * p_1.x + c = p_1.y \\ a * p_2.x^2 + b * p_2.x + c = p_2.y \end{cases} \Rightarrow \begin{cases} b * p_1.x + c = p_1.y - a * p_1.x^2 \\ b * p_2.x + c = p_2.y - a * p_2.x^2 \end{cases}$$

Como vemos, este sistema pode ter zero (se $p_1.x = p_2.x$ e $p_1.y \neq p_2.y$), uma ou infinitas soluções (se $p_1.x = p_2.x$ e $p_1.y = p_2.y$). No primeiro caso, p_1 e p_2 estão numa mesma vertical e, portanto, existe apenas uma curva da família que passa por eles (a reta $x = p_1.x$). No segundo caso a única parábola que passa por eles é dada pela solução do sistema. No terceiro caso, $p_1 = p_2$, portanto, a segunda condição também é satisfeita, e fica provado que estas curvas formam as retas de uma geometria $\mathcal{G}_\mathcal{F}$.

Podemos observar que, uma vez escolhida uma família de parábolas com o mesmo coeficiente do termo de segundo grau e as retas verticais, qualquer rotação de todas essas por um mesmo ângulo θ, também nos leva a uma família de curvas válida, como ilustrado na figura 2.3. É interessante notar que, neste caso, temos curvas que não são gráficos de funções.

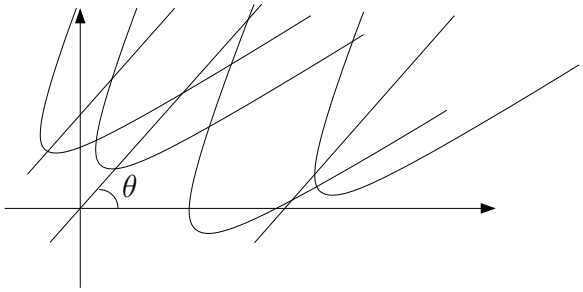


Figura 2.3: Outro exemplo da família de curvas $\mathcal{F}$. Parábolas com uma mesma abertura e retas verticais, todas rotacionadas por um mesmo ângulo θ.

Ainda no exemplo das parábolas, podemos notar que todas as curvas que compõem a família são translações arbitrárias de duas curvas iniciais: uma parábola de equação $ax^2 = 0$ e a reta $x = 0$. Dizemos então, que estas são as *curvas geradoras* desta família. Através de diferentes curvas geradoras podemos obter inúmeras famílias $\mathcal{F}$.

Definição 2.2.1 *Um conjunto de curvas é chamado* conjunto gerador *se suas curvas dão*

origem a uma família $\mathcal{F}$ por translações arbitrárias. Neste caso, as curvas do conjunto são chamadas curvas geradoras.

Na seção 2.2.1, analisaremos possíveis conjuntos geradores e suas características. Já na seção 2.2.2 serão analisados os casos das famílias $\mathcal{F}$ não obtidas a partir de geradoras.

2.2.1 Famílias $\mathcal{F}$ Formadas Através de Conjuntos Geradores

Como as curvas geradoras podem ser transladadas em qualquer direção, por todo o plano, algumas restrições podem ser identificadas em suas formas, como nos mostra os teoremas 2.2.1 e 2.2.2.

Teorema 2.2.1 *Para que uma curva C_0 possa ser uma curva geradora, um dos meios-planos de C_0 deve ser euclidianamente convexo.*

Prova: Por contradição, sejam H_1 e H_2 os dois meios-planos de C_0, e suponha que ambos não sejam euclidianamente convexos. Sejam p_1 e p_2 dois pontos em H_1 tais que o segmento $\overline{p_1p_2}$ não está inteiramente contido em H_1. Estes pontos existem, pois H_1 não é euclidianamente convexo. Seja p_{12} um ponto em $\overline{p_1p_2}$ que está em H_2. Da mesma forma, sejam p_3 e p_4 dois pontos em H_2 tais que o segmento $\overline{p_3p_4}$ não está inteiramente contido em H_2. Seja p_{34} um ponto em $\overline{p_3p_4}$ que está em H_1 [figura 2.4(a)]. Se transladarmos C_0 no sentido do vetor $\overrightarrow{p_{12}p_{34}}$ criaremos curvas com duas interseções com C_0, como a curva C_0' mostrada na figura 2.4(b), o que viola as condições de uma família $\mathcal{F}$. ■

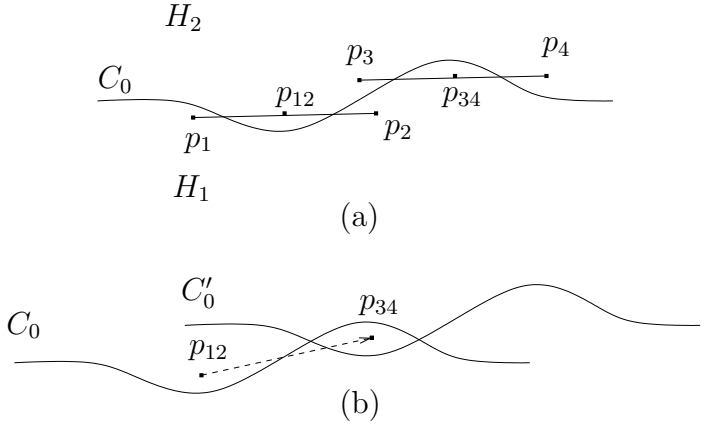


Figura 2.4: Ilustração do teorema 2.2.1. (a) Curva C_0 com dois meios planos euclidianamente não convexos. (b) A translação C_0' de C_0 gera duas interseções com C_0.

Teorema 2.2.2 *Para que uma curva C_0 possa ser uma curva geradora, se qualquer parte de C_0 for um segmento de reta euclidiano, então C_0 só pode ser a reta euclidiana que contém aquele segmento.*

Prova: Suponha que C_0 tenha uma parte que seja um segmento de reta euclidiana. Seja p um ponto num extremo desse segmento [figura 2.5 (a)]. Se transladarmos a curva C_0 na direção do segmento de reta incidente em p, chegaremos a uma outra curva C_0' cujo segmento euclidiano intersecta o da curva C_0 em um intervalo de comprimento positivo [figura 2.5 (b)]. Neste caso, C_0 e C_0' têm mais de um ponto de interseção, e, portanto, C_0 não pode ser uma curva geradora. ■

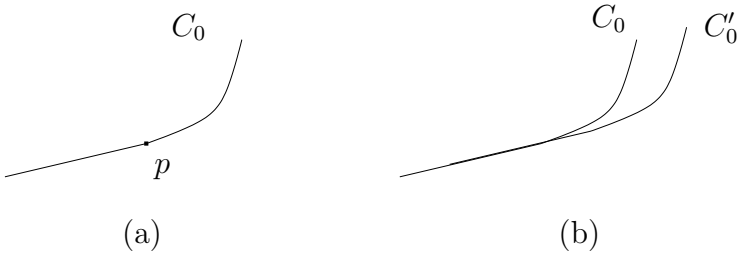


Figura 2.5: Ilustração do teorema 2.2.2. (a) Uma parte da curva C_0. O ponto p está no extremo de um segmento de reta euclidiano. (b) C_0' é uma translação de C_0 na direção do segmento de reta euclidiano contido em C_0.

Os dois teoremas anteriores nos mostram condições necessárias para uma curva ser geradora. Passemos, agora, a analisar como escolher uma família de geradoras de forma que estas gerem uma família $\mathcal{F}$. A seguir, serão discutidas curvas geradoras diferenciáveis, e, logo após, serão caracterizadas curvas não-diferenciáveis que possam formar um conjunto gerador.

Curvas Geradoras Diferenciáveis Um conjunto de curvas diferenciáveis, para ser gerador, precisa que suas curvas possam ser transladadas arbitrariamente pelo plano, gerando assim uma família de curvas $\mathcal{F}$ que satisfaça as três condições que definem uma geometria $\mathcal{G}_{\mathcal{F}}$. Como estamos interessados apenas em curvas geradoras diferenciáveis, qualquer conjunto delas e suas translações vão atender à terceira condição. Além disso, é fácil observar que uma translação em $\mathbb{R}^2$ de uma curva C passa pelo pólo norte de $\mathcal{R}^2$ se, e somente se, C também passa pelo pólo norte. Portanto, podemos escolher para formar um conjunto gerador apenas curvas que sejam curvas de Jordan e passem pelo pólo norte de $\mathcal{R}^2$. Desta forma, a primeira condição estará sempre satisfeita. Convém ressaltar que,

como estamos procurando curvas geradoras, qualquer candidata tem que satisfazer os teoremas 2.2.1 e 2.2.2.

Seja S um conjunto de curvas candidatas a formarem um conjunto gerador, do qual, qualquer curva satisfaz todas as condições mencionadas no parágrafo anterior. Para provarmos que S é um conjunto gerador, precisamos mostrar que, dados dois pontos p_1 e p_2 quaisquer em $\mathbb{R}^2$, sempre vai existir uma só translação de uma única curva em S passando por p_1 e p_2.

Seja l a reta euclidiana que passa por p_1 e p_2, e α a inclinação de l. Considere uma curva $C \in S$ tenha um ponto q no qual a derivada de C seja α. Vamos mostrar que existe uma única translação de C que passa por p_1 e p_2.

Se C for a reta euclidiana de inclinação α, é óbvio que existe uma translação de C que coincide com l e, portanto, passa por p_1 e p_2. Assim, vamos nos concentrar no caso onde C não é uma reta euclidiana.

É fácil perceber que, como C possui um de seus meios planos euclidianamente convexo, pelo teorema 2.2.1, existe uma paralela a l que intercepta C em dois pontos q_1 e q_2 (uma perturbação da paralela a l em torno de q gera tal reta, como ilustrado em 2.6). Vamos observar o que ocorre com os pontos q_1 e q_2 quando movemos esta paralela na direção ortogonal a ela, no sentido que a afasta de q.

O segmento $\overline{q_1 q_2}$ está contido no meio plano convexo definido por C. Como os dois braços de C que partem de q vão para o infinito (já que C passa pelo pólo norte de $\mathcal{R}^2$), e C satisfaz os teoremas 2.2.1 e 2.2.2, então q_1 e q_2 só podem se afastar um do outro. Se não se afastassem, então poderiam permanecer a mesma distância, o que violaria o teorema 2.2.2, ou se aproximariam, o que faria com que C não chegasse ao pólo norte de $\mathcal{R}^2$ ou violasse 2.2.1.

Como a variação da distância entre q_1 e q_2 é estritamente crescente, existe uma paralela a l onde a distância entre q_1 e q_2 é maior que entre p_1 e p_2. Assim, pela continuidade de C podemos afirmar que existe uma única paralela a l onde a distância entre q_1 e q_2 é exatamente igual à distância entre p_1 e p_2. Com isso, C pode ser transladada de forma a passar por p_1 e p_2.

Neste caso, dizemos que C cobre o ângulo α, pois qualquer outro par de pontos, cuja reta euclidiana passando por eles tenha inclinação α, também terá uma única curva, translação de C, passando por eles. Obviamente, não podemos ter mais nenhuma outra curva no conjunto S cobrindo α, pois senão teriamos duas $\mathcal{G}_{\mathcal{F}}$-retas distintas, cada uma criada a partir da translação de uma geradora, passando por p_1 e p_2. Com base nesta afirmação, chegamos a uma condição suficiente para que um conjunto de curvas seja um conjunto gerador:

Teorema 2.2.3 *Dado um conjunto de curvas diferenciáveis, que sejam curvas de Jordan e passem pelo pólo norte de $\mathcal{R}^2$, e que satisfaçam os teoremas 2.2.1 e 2.2.2, este dará*

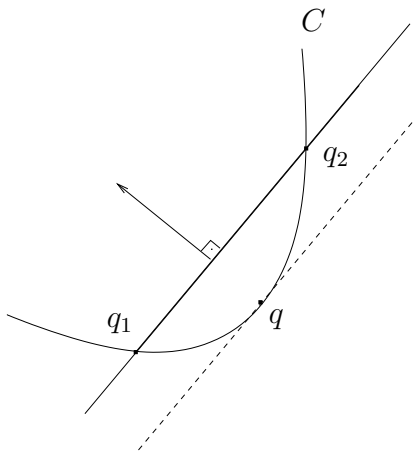


Figura 2.6: Uma perturbação da paralela a l em torno de q gera uma reta que intercepta C em dois pontos q_1 e q_2. A seta indica a direção na qual a paralela será movida.

origem a uma família $\mathcal{F}$ se suas curvas cobrirem de forma unívoca todo o intervalo de ângulos entre $[0, \pi)$.

Prova: Como provado anteriormente, o fato das curvas serem curvas de Jordan diferenciáveis passando pelo pólo norte de $\mathcal{R}^2$ garante que as condições 1 e 3 para se formar uma geometria $\mathcal{G}_{\mathcal{F}}$ serão satisfeitas. Além disso, dados dois pontos quaisquer em $\mathbb{R}^2$, a reta euclidiana que passa por eles tem inclinação entre $[0, \pi)$. Como as curvas do conjunto cobrem de forma unívoca o intervalo $[0, \pi)$, então sempre existe uma única translação de uma curva no conjunto passando por quaisquer dois pontos no plano. Com isso, a segunda condição também fica satisfeita e o conjunto de curva dá origem a uma família $\mathcal{F}$. ■

No exemplo das parábolas, vemos que cada parábola, independente da abertura, cobre o intervalo de pontos dispostos formando retas com inclinação em $[0, \pi/2) \cup (\pi/2, \pi)$. Assim, os únicos pares de pontos que não são cobertos, são aqueles numa mesma vertical (inclinação $\pi/2$). Portanto, uma parábola de função $y = ax^2$ e uma vertical $x = 0$ formam um conjunto gerador.

Através da observação da necessidade de cobertura de um intervalo de inclinações por um conjunto de curvas geradoras, chegamos ao teorema 2.2.4.

Teorema 2.2.4 *Pelo menos uma reta euclidiana deve estar presente em cada conjunto gerador.*

Prova: O intervalo $[0, \pi)$ a ser coberto por todas as curvas geradoras é um intervalo semi-aberto. Cada curva que não seja uma reta euclidiana e passe pelo pólo norte

de $\mathcal{R}^2$ cobre um intervalo aberto, pois esta não pode conter nenhum segmento de reta euclidiana. Qualquer união de intervalos abertos é ainda um intervalo aberto. Assim, para cobrir o intervalo semi-aberto, precisamos de pelo menos uma reta euclidiana no conjunto gerador. ■

Vemos que, no caso onde rotacionamos as parábolas e a reta vertical por um mesmo ângulo θ, a única modificação que estamos fazendo é trocar o intervalo de cobertura de cada uma das duas geradoras.

Com base nessas idéias, chegamos a outros exemplos de famílias de curvas que satisfazem as três condições que definem uma geometria $\mathcal{G}_{\mathcal{F}}$. Entre eles podemos citar:

1.
   - Curva dada pela função $f(x) = a/x$ para $x > 0$ e a constante;
   - Reta $f(x) = bx$ para $b \in [0, \infty)$;
   - Reta $x = 0$. (figura 2.7)

2.
   - Curva dada pela função $f(x) = a/x$ para $x > 0$ e a constante;
   - Curva dada pela função $f(x) = -b/x$ para $x < 0$ e b constante;
   - Reta $f(x) = 0$;
   - Reta $x = 0$. (figura 2.8)

3.
   - Qualquer rotação das geradoras anteriores, todas por um mesmo ângulo.

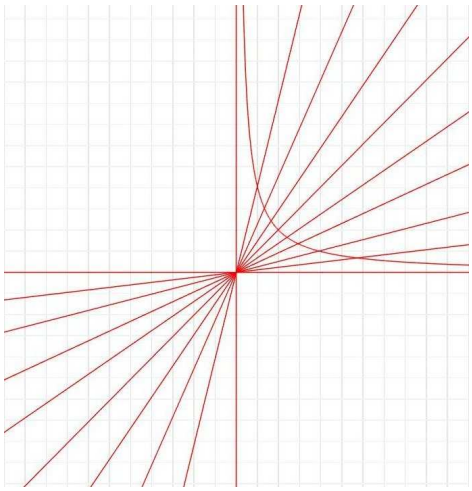


Figura 2.7: Conjunto gerador formado pela curva dada pela função $f(x) = a/x$ para $x > 0$ e a constante, pela reta $f(x) = bx$ para $b \in [0, \infty)$ e pela reta $x = 0$.

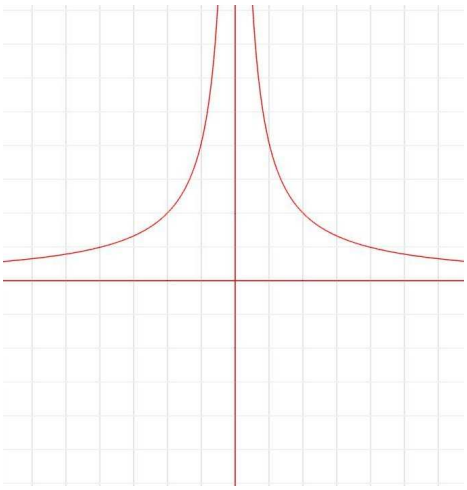


Figura 2.8: Conjunto gerador formado pela curva dada pela função $f(x) = a/x$ para $x > 0$ e a constante, curva dada pela função $f(x) = -b/x$ para $x < 0$ e b constante, pela reta $f(x) = 0$ e pela reta $x = 0$.

Agora passemos a analisar o comportamento de curvas geradoras quando permitimos pontos de não diferenciabilidade nas mesmas.

Curvas Geradoras Não Diferenciáveis Aqui, serão analisadas conjuntos geradores que possuam curvas não diferenciáveis.

Assim como no caso das curvas diferenciáveis, para que um conjunto S de curvas seja gerador, as translações arbitrárias de suas curvas pelo plano devem formar uma família $\mathcal{F}$. Como já argumentado no caso das curvas diferenciáveis, as curvas candidatas devem ser curvas de Jordan que passem pelo pólo norte de $\mathcal{R}^2$, caso contrário, não podem ser geradoras. Além disso, o número de partes diferenciáveis nas curvas de S deve ter um limitante, para que este satisfaça a terceira condição. Apesar da não diferenciabilidade das curvas, os teoremas 2.2.1 e 2.2.2 continuam sendo condições necessárias para que qualquer curva candidata possa fazer parte de um conjunto gerador.

Para verificarmos que a segunda condição da geometria $\mathcal{G}_{\mathcal{F}}$ é satisfeita, voltemos à análise feita no caso das curvas diferenciáveis. Naquele caso, partimos de um ponto q numa curva $C \in S$, e tinhamos que a derivada de C em q era igual à inclinação α da reta euclidiana l que passava por p_1 e p_2. No caso das curvas contendo pontos não diferenciáveis, não podemos tomar o ponto q como o ponto em C cuja derivada seja igual à inclinação α da reta euclidiana passando pelos pontos p_1 e p_2, uma vez que esse ponto q pode não existir. Porém, é fácil observar que todos os argumentos usados no caso anterior são válidos se tomarmos q como o ponto pelo qual passa uma reta de suporte de C de inclinação igual a α (como ilustrado na figura 2.9). Como C atende às condições apresentadas no parágrafo anterior, pelos mesmos argumentos apresentados no caso das curvas diferenciáveis, podemos dizer que C cobre o ângulo α. Portanto, chegamos ao seguinte teorema, que é mais abrangente que o teorema 2.2.3:

Teorema 2.2.5 *Dado um conjunto de curvas que sejam curvas de Jordan e passem pelo pólo norte de $\mathcal{R}^2$, que satisfaçam os teoremas 2.2.1 e 2.2.2, e onde existe um limitante para o número de partes diferenciáveis das curvas do conjunto, este dará origem a uma família $\mathcal{F}$ se suas curvas cobrirem de forma unívoca todo o intervalo de ângulos entre* $[0, \pi)$.

Portanto, qualquer conjunto de curvas que satisfaça o teorema 2.2.5 pode ser usado como um conjunto gerador de uma família de curvas $\mathcal{F}$.

Passemos agora ao caso das família $\mathcal{F}$ que não são obtidas de curvas geradoras.

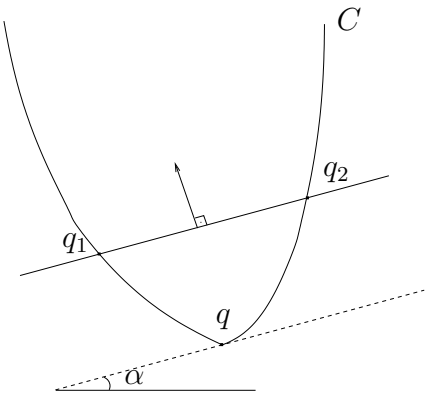


Figura 2.9: A curva C não diferenciável cobre o ângulo α mesmo sem ter pontos de derivada com estes valores.

2.2.2 Famílias $\mathcal{F}$ Arbitrárias

O teorema 2.2.5 nos mostra uma condição suficiente para que formemos uma família $\mathcal{F}$, porém, esta não é a única forma fazê-lo.

Considere o plano $\mathbb{R}^2$ e todas as retas euclidianas contidas nele. A figura 2.10 ilustra uma parte deste plano, com algumas retas verticais e horizontais, mostrado em quatro visões.

Agora, causemos uma deformação no plano, como se o mesmo fosse colocado sobre uma esfera e tomasse o formato do topo da mesma, como ilustrado na figura 2.11.

Uma reta euclidiana no plano, quando deformada gera uma curva em $\mathbb{R}^3$. Considere a projeção vertical, paralela ao eixo z, de cada uma das curvas geradas no espaço, formadas pelas deformações das retas euclidianas, sobre $\mathbb{R}^2$. Um pedaço deste plano, formado pelos pontos de $\mathbb{R}^2$ e por algumas destas projeções, pode ser visto na visão superior da figura 2.11. A geometria formada é uma geometria $\mathcal{G}_\mathcal{F}$. Isto pode ser verificado pois, uma vez que toda curva é formada através de uma deformação das retas euclidianas, estas satisfazem as condições um e três da geometria $\mathcal{G}_\mathcal{F}$, faltando apenas a verificação da segunda condição.

Considere os pontos ${p_1}'$ e ${p_2}'$ em $\mathbb{R}^2$, distorções de dois pontos p_1 e p_2 no plano original. A distorção da reta euclidiana que passava por p_1 e p_2 passa por ${p_1}'$ e ${p_2}'$, assim existe uma curva que passa por ${p_1}'$ e ${p_2}'$. Suponha que esta curva não seja única. Então existem duas curvas diferentes passando por ${p_1}'$ e ${p_2}'$, que são originadas por duas retas diferentes que passavam por p_1 e p_2, o que é um absurdo. Portanto, dados dois pontos quaisquer no plano, vai existir apenas uma curva passando por eles, completando a prova de que temos uma geometria $\mathcal{G}_\mathcal{F}$.

É fácil notar que, para gerarmos a superfície que dará origem às curvas da geometria, podemos ter inúmeras e diferentes distorções no plano original. Cada uma em uma

posição, e com diâmetros diferentes. Isto pode fazer com que tenhamos $\mathcal{G}_{\mathcal{F}}$-retas que possuam os dois $\mathcal{G}_{\mathcal{F}}$-meios-planos não euclidianamente convexos. Pode-se perceber que uma tal geometria não advém de famílias geradas por translações de curvas.

Como os exemplos de geometrias $\mathcal{G}_{\mathcal{F}}$ não se restringem aos casos formados por conjuntos geradores, podemos ter, nesta dissertação, ilustrações que violam os teoremas válidos apenas para as famílias formadas através de geradoras. Por outro lado, as ilustrações estarão sempre de acordo com as propriedades que definem a geometria $\mathcal{G}_{\mathcal{F}}$.

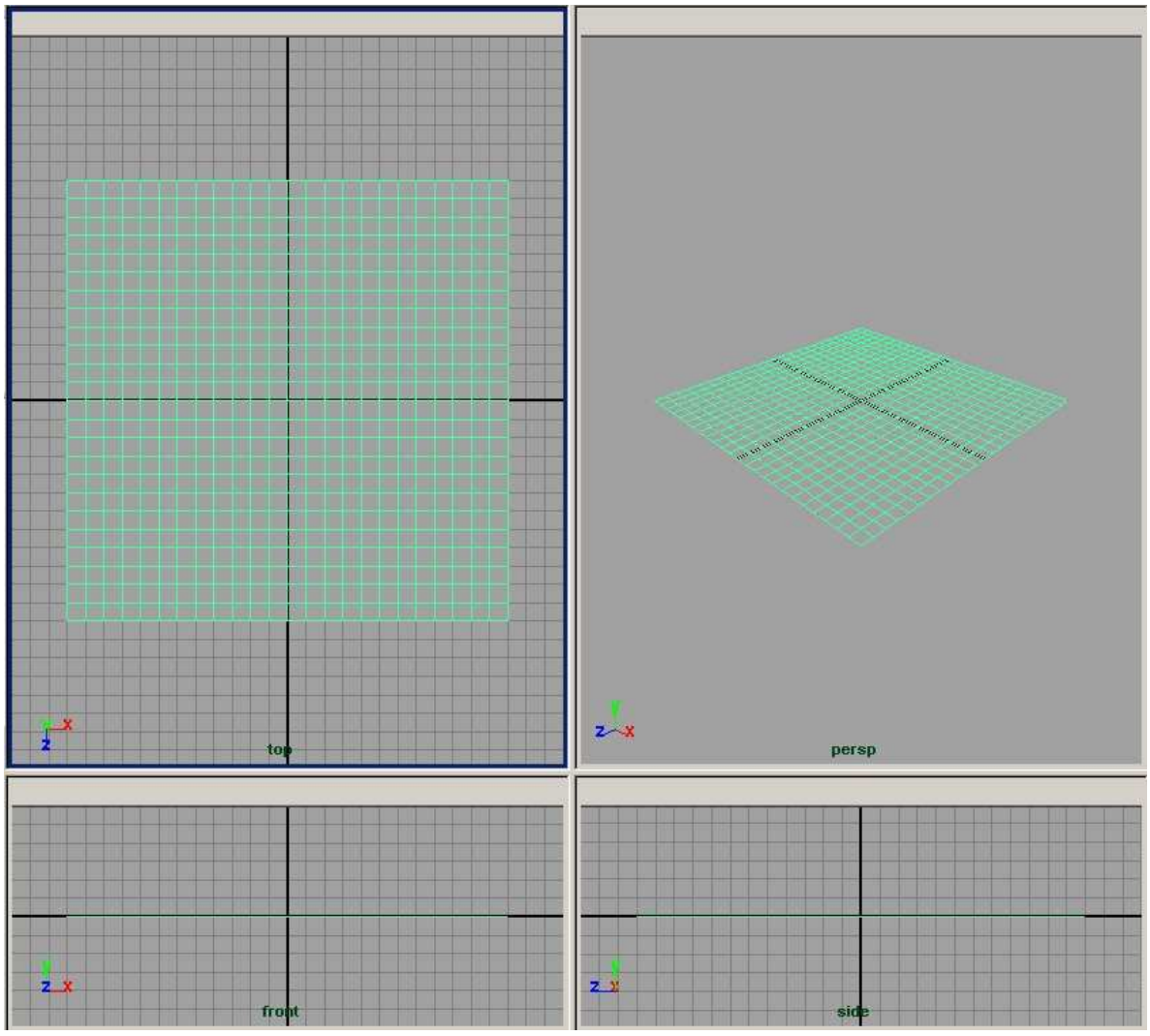


Figura 2.10: Pedaço de $\mathbb{R}^2$ em quatro visões. No sentido horário: superior, perspectiva, lateral e frente.

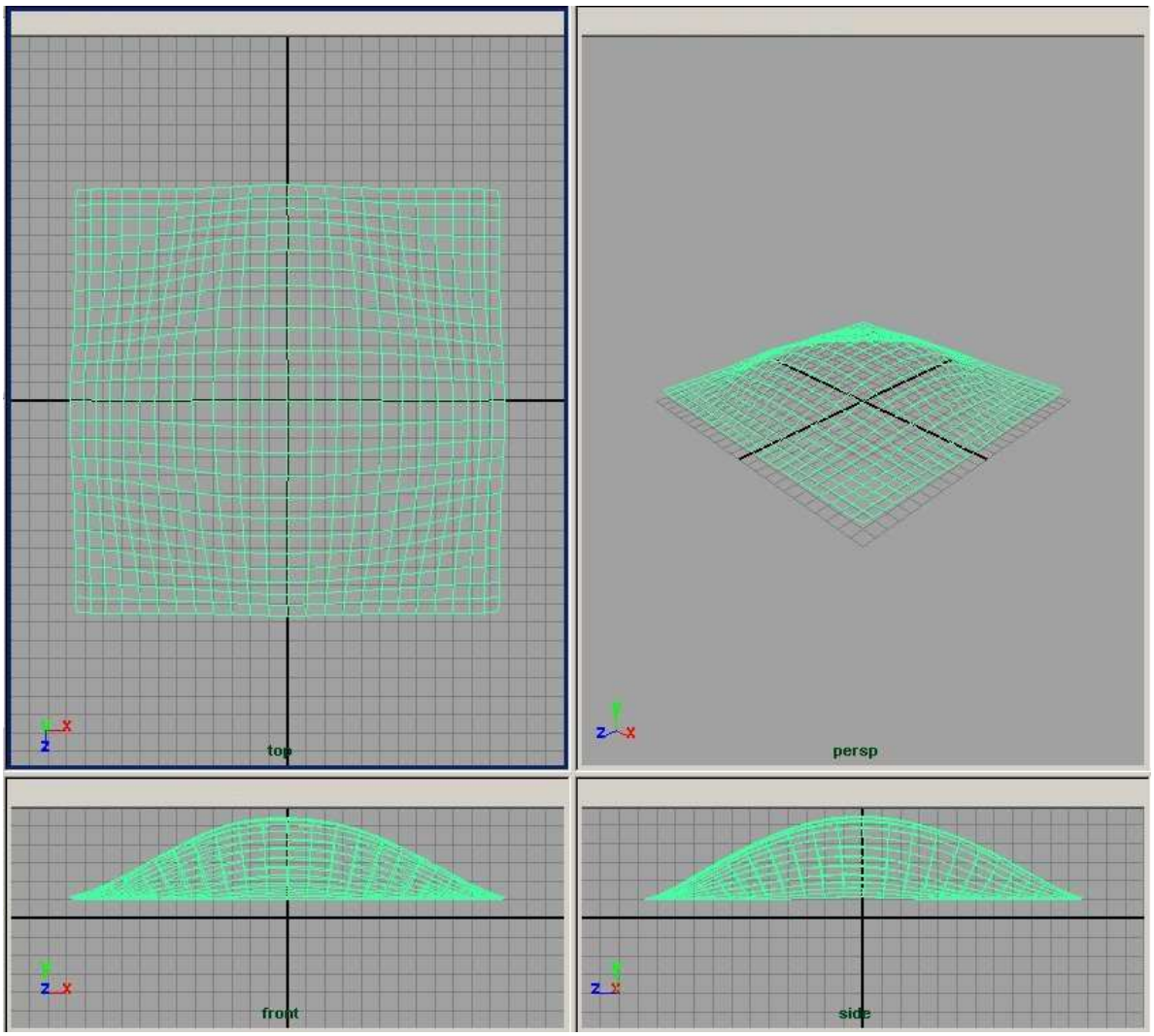


Figura 2.11: Pedaço de $\mathbb{R}^2$ distorcido por uma esfera em quatro visões. No sentido horário: superior, perspectiva, lateral e frente.

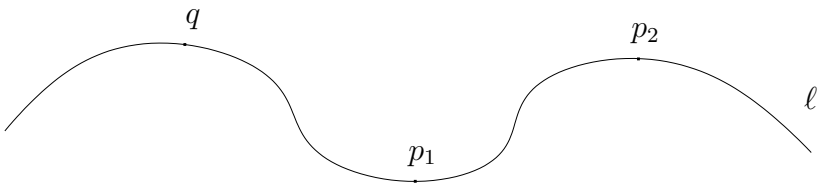


Figura 2.12: O ponto p_1 precede p_2 em ℓ com relação a q.

2.3 Relações de Precedência

Nesta seção, serão definidas algumas relações de precedência na geometria $\mathcal{G}_{\mathcal{F}}$. Essas relações são utilizadas por diversos algoritmos apresentados nesta dissertação.

2.3.1 Precedência entre Pontos sobre uma $\mathcal{G}_{\mathcal{F}}$-reta

Seja $\widetilde{p_1p_2}$ o $\mathcal{G}_{\mathcal{F}}$-segmento de reta que liga p_1 a p_2, mas não contém o ponto no infinito.

Definição 2.3.1 *[Har00] Seja ℓ uma $\mathcal{G}_{\mathcal{F}}$-reta e sejam p_1 e p_2 dois pontos distintos de $\mathbb{R}^2$ em ℓ. Considere um terceiro ponto $q \in \mathbb{R}^2$ tal que $q \notin \widetilde{p_1p_2}$ e $q \in \ell$. Dizemos que p_1* precede *p_2 em ℓ com relação a q se $p_1 \in \widetilde{qp_2}$; em caso contrário, dizemos que p_1* sucede *p_2 em ℓ com relação a q. Denotamos o primeiro caso por $p_1 \prec_q^{\ell} p_2$ e o segundo por $p_2 \prec_q^{\ell} p_1$ (figura 2.12).*

2.3.2 Precedência entre $\mathcal{G}_{\mathcal{F}}$-retas

Nas definições a seguir, considere P um subconjunto de $\mathbb{R}^2$.

Definição 2.3.2 *[Har00] Seja ℓ uma $\mathcal{G}_{\mathcal{F}}$-reta. Dizemos que ℓ é uma $\mathcal{G}_{\mathcal{F}}$-*reta de suporte *de P se P está contido na união de ℓ com um de seus $\mathcal{G}_{\mathcal{F}}$-semi-planos e, além disso, ℓ contém pelo menos um ponto de P.*

Proposição 2.3.1 *(adaptado de [Dra52]) Seja p_0 um ponto fora do casco $\mathcal{G}_{\mathcal{F}}$-convexo de P. Então, existem exatamente duas $\mathcal{G}_{\mathcal{F}}$-retas passando por p_0 que são $\mathcal{G}_{\mathcal{F}}$-retas de suporte de P.*

Definição 2.3.3 *[Har00] Seja p_0 um ponto fora do casco $\mathcal{G}_{\mathcal{F}}$-convexo de P. Sejam ℓ_1 e ℓ_2 $\mathcal{G}_{\mathcal{F}}$-retas de suporte de P passando por p_0 e sejam $\mathcal{H}_1$ e $\mathcal{H}_2$ os $\mathcal{G}_{\mathcal{F}}$-semi-planos limitados por ℓ_1 e ℓ_2, respectivamente, que contêm P. A interseção $\mathcal{H}_1 \cap \mathcal{H}_2$ é denominada* região de interesse *relativa ao ponto p_0 (figura 2.13).*

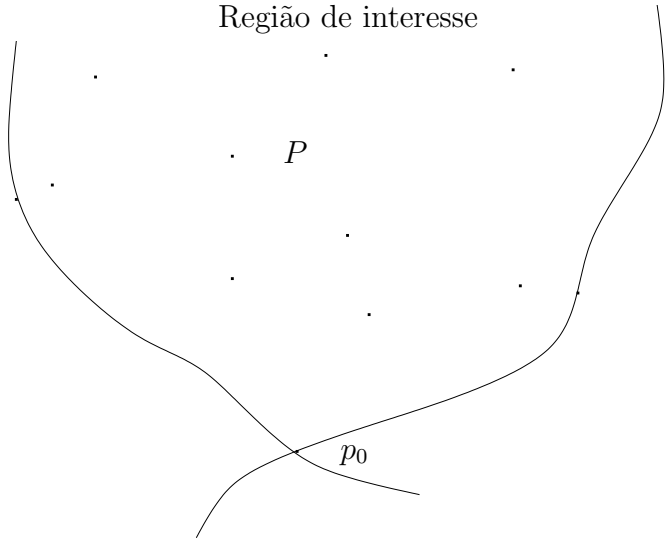


Figura 2.13: Região de interesse relativa ao ponto p_0.

A partir do conceito de região de interesse pode-se definir uma relação entre pontos situados dentro de uma mesma região deste tipo.

Definição 2.3.4 *[Har00] Seja p_0 um ponto fora do casco $\mathcal{G}_\mathcal{F}$-convexo de P e sejam p_1 e p_2 dois pontos distintos de P. Seja $B_\mathcal{F}(p_0, \rho)$ uma $\mathcal{G}_\mathcal{F}$-circunferência com centro em p_0 e raio $\rho > 0$. Escolha um ponto p_e sobre $\partial B_\mathcal{F}(p_0, \rho)$ (fronteira de $B_\mathcal{F}(p_0, \rho)$) fora da região de interesse relativa a p_0. Dizemos que p_1 está à* esquerda (direita) *de p_2 com relação a p_0 se encontrarmos o $\mathcal{G}_\mathcal{F}$-segmento $\widetilde{p_0p_1}$ antes (depois) do $\mathcal{G}_\mathcal{F}$-segmento $\widetilde{p_0p_2}$ quando caminhamos sobre $\partial B_\mathcal{F}(p_0, \rho)$ em sentido horário a partir de p_e (figura 2.14). Denotamos por $p_1 \models_{p_0} p_2$ ($p_2 \models_{p_0} p_1$) quando p_1 está a esquerda (direita) de p_2 com relação a p_0.*

Definição 2.3.5 *[Har00] Seja p_0 um ponto fora do casco $\mathcal{G}_\mathcal{F}$-convexo de P. Dizemos que um ponto $p \in P$ está à* esquerda (direita) *de uma $\mathcal{G}_\mathcal{F}$-reta ℓ que passa por p_0 e que intercepta o interior de sua região de interesse se, para qualquer ponto $q \in \ell$ pertencente à região de interesse com relação a p_0, tivermos que $p \models_{p_0} q$ ($q \models_{p_0} p$).*

Definição 2.3.6 *Seja p_0 um ponto fora do casco $\mathcal{G}_\mathcal{F}$-convexo de P. Sejam ℓ_1 e ℓ_2 $\mathcal{G}_\mathcal{F}$-retas passando por p_0 e pela região de interesse relativa a p_0. Dizemos que ℓ_1 está à* esquerda (direita) de ℓ_2 se qualquer ponto $p \in \ell_1$ e pertencente à região de interesse com relação a p_0 está a esquerda (direita) de ℓ_2.

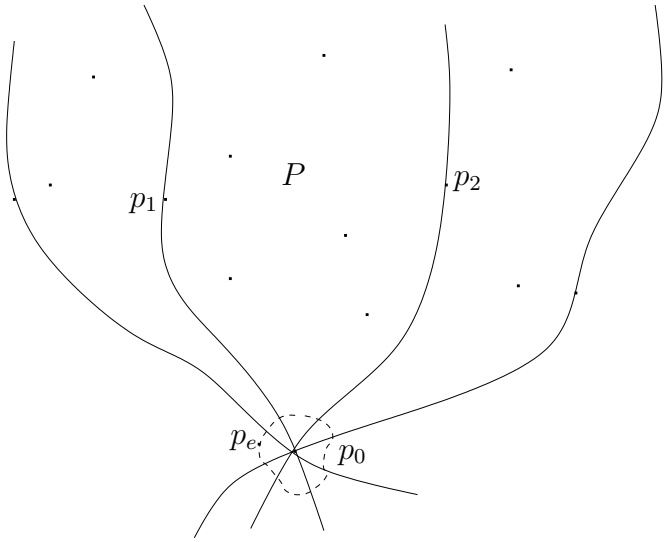


Figura 2.14: O ponto p_1 está a esquerda do ponto p_2 com relação a p_0.

As definições 2.3.4, 2.3.5 e 2.3.6 são utilizadas pelos algoritmos propostos nas seções 3.2.2, 3.3.2 e 3.6.2.

2.4 Predicados e Primitivas

Ao observarmos os algoritmos para os diversos problemas geométricos existentes, vemos que determinados predicados e primitivas são recorrentes em muitos deles. Entre esses predicados podemos citar: incidência de ponto em reta, orientação de pontos e pertinência de ponto a um círculo. Quanto as primititivas, podemos citar a distância entre dois pontos e a precedência entre pontos.

Dado um algoritmo desenvolvido para um problema definido em uma determinada geometria, é possível identificar quais predicados e primitivas são necessários para sua corretude e podemos adaptar o algoritmo para o problema em uma outra geometria, desde que esses predicados e primitivas possam ser avaliados nessa outra geometria.

Nesta seção, faremos uma caracterização de alguns predicados e algumas primitivas que podem ser avaliados na geometria $\mathcal{G}_{\mathcal{F}}$.

- Distância entre pontos

  Temos da definição da geometria $\mathcal{G}_{\mathcal{F}}$ que dados dois pontos, existe apenas uma $\mathcal{G}_{\mathcal{F}}$-reta que passa por eles. Assim, a distância entre pontos pode ser definida como o comprimento do $\mathcal{G}_{\mathcal{F}}$-segmento que passa por eles e que não passa pelo ponto do

infinito. Esta distância pode ser calculada por uma integral de linha sobre a $\mathcal{G}_\mathcal{F}$-reta dada. A notação $d_\mathcal{F}(p_1, p_2)$ representa a distância entre os pontos p_1 e p_2 na geometria $\mathcal{G}_\mathcal{F}$, como apresentada no capítulo 2.

- Precedência entre pontos

  A decisão de precedência entre pontos deve ser feita com relação a um referencial.

  Um primeiro referencial pode ser um terceiro ponto em $\mathbb{R}^2$ colinear aos dois pontos dados. Neste caso, podemos definir a precedência entre pontos como descrito na definição 2.3.1.

  Podemos também definir a precedência entre pontos dentro da região de interesse de um conjunto de pontos, como apresentado na definição 2.3.4.

- Precedência de ponto com relação a reta

  A precedência de ponto com relação a reta pode ser definida dentro de uma região de interesse, como apresentado na definição 2.3.5.

- Precedência entre retas

  A precedência entre retas pode ser definida dentro de uma região de interesse, como apresentado na definição 2.3.6.

- Orientação de pontos (CW ou CCW)

  Dado um trio de pontos (p_1, p_2, p_3), a decisão sobre a orientação dos mesmos nos indica a posição de cada um deles com relação à $\mathcal{G}_\mathcal{F}$-reta que passa pelos outros dois.

  Para decidirmos sobre a orientação dos pontos precisamos olhar para a $\mathcal{G}_\mathcal{F}$-reta orientada ℓ que vai de p_1 a p_2. Se p_3 está localizado no $\mathcal{G}_\mathcal{F}$-meio-plano à direita de ℓ então dizemos que os pontos estão orientados no sentido horário (CW). Caso contrário, os pontos estão orientados no sentido anti-horário (CCW) e, se $p_3 \in \ell$, dizemos que os três pontos são $\mathcal{G}_\mathcal{F}$-colineares.

  Porém, a partir da equação que define uma $\mathcal{G}_\mathcal{F}$-reta não podemos concluir qual sua orientação, isto é, não sabemos diferenciar a $\mathcal{G}_\mathcal{F}$-reta que vai de p_1 a p_2 da $\mathcal{G}_\mathcal{F}$-reta que vai de p_2 a p_1.

  Seja ℓ a $\mathcal{G}_\mathcal{F}$-reta que contém p_1 e p_2. Considere uma $\mathcal{G}_\mathcal{F}$-circunferência $B_\mathcal{F}(p_1, \rho)$ centrada em p_1 e de raio $\rho > 0$ (figura 2.15). Agora, considere o percurso no sentido anti-horário feito sobre $B_\mathcal{F}(p_1, \rho)$, a partir do ponto q de interseção de $B_\mathcal{F}(p_1, \rho)$ com $\widetilde{p_1p_2}$. Se encontrarmos o segmento $\widetilde{p_1p_3}$ antes de cruzarmos ℓ, dizemos que os pontos

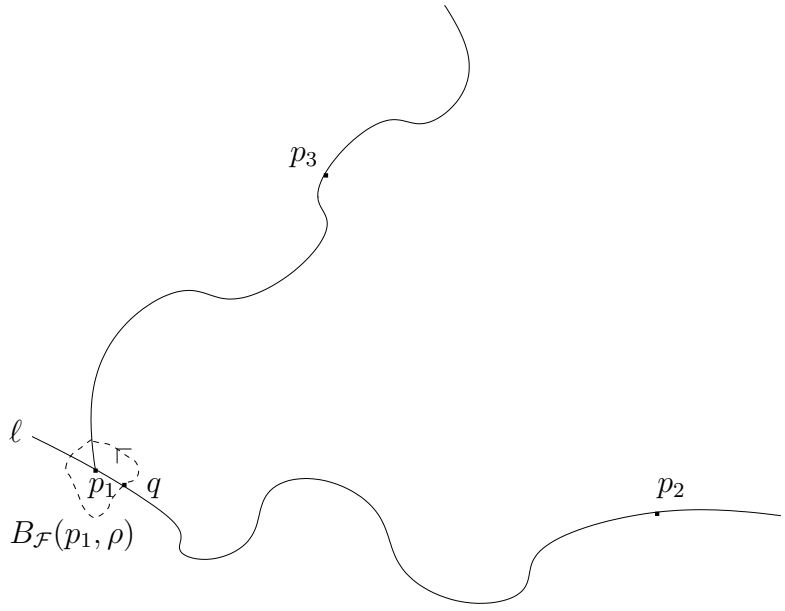


Figura 2.15: Pontos p_1, p_2 e p_3 com orientação positiva. O ponto q é $B_{\mathcal{F}}(q, \rho) \cap \widetilde{p_1 p_2}$. O percurso é feito a partir de q no sentido anti-horário.

p_1, p_2, p_3 têm orientação positiva (CCW), caso contrário, têm orientação negativa (CW).

Nesta dissertação, a orientação de um trio de pontos (p_1, p_2, p_3) será representada por $\Delta(p_1, p_2, p_3)$.

- Incidência de ponto em reta

  Dada a função paramétrica $f(t) = (x(t), y(t)), t \in [0, 1]$ que define uma $\mathcal{G}_{\mathcal{F}}$-reta ℓ, podemos decidir se um ponto p está sobre a mesma verificando se existe um valor de t no intervalo $[0, 1]$ que satisfaça a equação $(p.x, p.y) = (x(t), y(t))$.

- Interseção de segmentos

  O teste de interseção entre uma $\mathcal{G}_{\mathcal{F}}$-reta e um $\mathcal{G}_{\mathcal{F}}$-segmento é necessário em vários algoritmos. Encontrar o ponto de interseção pode não ser uma tarefa trivial, pois depende da complexidade da família $\mathcal{F}$ escolhida na formação da geometria. Por outro lado, podemos decidir se um $\mathcal{G}_{\mathcal{F}}$-segmento intercepta uma $\mathcal{G}_{\mathcal{F}}$-reta utilizando o teste de orientação dos extremos do $\mathcal{G}_{\mathcal{F}}$-segmento com relação à dois pontos pertencentes à $\mathcal{G}_{\mathcal{F}}$-reta. O $\mathcal{G}_{\mathcal{F}}$-segmento intercepta a $\mathcal{G}_{\mathcal{F}}$-reta se, e somente se, um de seus extremos está do lado direito da $\mathcal{G}_{\mathcal{F}}$-reta e o outro extremo está do lado esquerdo.

Caso um dos extremos do $\mathcal{G}_\mathcal{F}$-segmento pertença à $\mathcal{G}_\mathcal{F}$-reta, a interseção também ocorre.

Para decidirmos sobre a interseção entre dois $\mathcal{G}_\mathcal{F}$-segmentos $\widetilde{s}_1$ e $\widetilde{s}_2$, podemos utilizar o teste de interseção de $\mathcal{G}_\mathcal{F}$-segmento com $\mathcal{G}_\mathcal{F}$-reta. Porém, aqui precisamos testar cada $\mathcal{G}_\mathcal{F}$-segmento com a $\mathcal{G}_\mathcal{F}$-reta que contém o outro. Assim, $\widetilde{s}_1$ intercepta $\widetilde{s}_2$ se, e somente se, $\widetilde{s}_1$ intercepta a $\mathcal{G}_\mathcal{F}$-reta que contém $\widetilde{s}_2$ e $\widetilde{s}_2$ também intercepta a $\mathcal{G}_\mathcal{F}$-reta que contém $\widetilde{s}_1$.

- In Circle

  Outro predicado recorrente em algoritmos geométricos é a verificação da pertinência de um ponto q em um círculo formado por outros três pontos p_1, p_2 e p_3 não colineares. Porém, as propriedades que definem a geometria $\mathcal{G}_\mathcal{F}$ não são suficientes para garantir que temos um único ponto equidistante de p_1, p_2 e p_3, que caracteriza o centro do círculo passando por eles.

  Algumas famílias de curvas que atendem as propriedades exigidas para definir uma geometria $\mathcal{G}_\mathcal{F}$ garantem a unicidade do centro do círculo, como no caso das parábolas de mesma abertura junto com as retas verticais, ou a própria geometria euclidiana. Nestes casos, este predicado pode ser avaliado corretamente. Porém, sua avaliação depende da resolução de um sistema para encontrarmos o centro do círculo. Isso, por sua vez, depende da complexidade da família de curvas $\mathcal{F}$ escolhida. Através da solução desse sistema podemos encontrar facilmente o tamanho do raio e, assim, tomar uma decisão se q está ou não contido no círculo.

  Portanto, a verificação desse predicado está diretamente ligada à complexidade da família de curvas $\mathcal{F}$ escolhida.

- Ordenação circular de pontos

  A ordenação de pontos é uma ferramenta bastante recorrente em algoritmos de varredura.

  Na geometria $\mathcal{G}_\mathcal{F}$ é possível fazermos a ordenação de pontos de um conjunto P com relação a um ponto p_0 fora da envoltória convexa de P. Neste caso, utilizamos a precedência de pontos dentro da região de interesse definida por p_0 para definirmos uma ordenação circular dos pontos de P em torno de p_0.

Capítulo 3

Problemas

Neste capítulo são apresentados diversos problemas de natureza geométrica, com soluções bem conhecidas na geometria euclidiana. Em cada um deles é discutida uma ou mais soluções aplicadas na geometria euclidiana e, então, são propostos algoritmos para os problemas sobre a geometria $\mathcal{G}_{\mathcal{F}}$.

A seção 3.1 discute o problema da localização de pontos em relação a polígonos. São estudados os casos onde a entrada é um polígono simples, convexo ou estrelado. Na seção 3.2, são propostos dois algoritmos para o problema de interseção de polígonos convexos sobre a geometria $\mathcal{G}_{\mathcal{F}}$. Já a seção 3.3 trata do problema de simplicidade de polígonos. Os problemas de saparabilidade de conjuntos e separabilidade de polígonos são analisados nas seções 3.4.1 e 3.4.2, respectivamente. A seção 3.5 descreve um algoritmo incremental para construção do casco convexo de um conjunto de pontos, que foi implementado no visualizador `GFViewer`, mostrado no capítulo 4. Na seção 3.6, é apresentada uma solução eficiente para o problema da construção do casco convexo de um polígono simples. Por fim, a seção 3.7 analisa como encontrar o núcleo de um polígono simples.

Um algoritmo proposto para um problema geométrico sobre a geometria $\mathcal{G}_{\mathcal{F}}$ deve funcionar independentemente da família de curvas $\mathcal{F}$ escolhida na formação da mesma. Porém, na maioria dos casos, a escolha da família influencia na eficiência do algoritmo desenvolvido. Como discutido no apêndice A, as características da família de curvas $\mathcal{F}$ pode adicionar constantes multiplicativas à complexidade dos algoritmos aqui apresentados, porém, a complexidade assintótica não deve ser alterada.

Na análise dos problemas, a não ser quando especificado o contrário, consideramos que um polígono em $\mathbb{R}^2$ é formado por seu interior e sua fronteira. Além disso, seus vértices são dados no sentido anti-horário, isto é, dada uma aresta orientada do vértice i para o vértice $i+1$, o polígono sempre está contido no meio-plano fechado à esquerda da mesma. Em todas as operações de soma e subtração feitas nos índices dos vértices dos polígonos o resultado é tomado *mod* n, onde n é o número de vértices do polígono com vértices

$v_0, v_1, \ldots, v_{n-1}$.

3.1 Localização em um Polígono Simples

O problema da localização de ponto com relação a um polígono simples é definido como:

Problema 1 *Dado um polígono simples $\mathcal{P}$ com n vértices e um ponto p, determinar se p é interno a $\mathcal{P}$.*

A seção 3.1.1 apresenta um algoritmo para este problema sobre a geometria $\mathcal{G}_{\mathcal{F}}$. Nas seções 3.1.2 e 3.1.3 são discutidos os casos onde temos polígonos convexos ou estrelados na entrada, respectivamente. Nestes casos, podemos utilizar estas características para obtermos algoritmos mais eficientes.

3.1.1 Polígono Simples

Geometria Euclidiana No caso euclidiano, este problema pode ser resolvido em tempo $O(n)$, onde n é o número de vértices de $\mathcal{P}$. Um algoritmo que atinge essa taxa utiliza o fato de um polígono ser uma curva de Jordan fechada, e assim, dividir o plano em duas regiões disjuntas, o interior (região limitada) e o exterior (região ilimitada).

Tomando l como a reta horizontal que passa por p, se l não intercepta nenhuma aresta de $\mathcal{P}$, então p está no seu exterior. Portanto, podemos nos concentrar no caso em que l intercepta $\mathcal{P}$. Observando primeiramente o caso onde l não passa por nenhum vértice de $\mathcal{P}$, caminhando sobre l de $x = -\infty$ em direção a p, se atravessarmos uma aresta de $\mathcal{P}$, então estaremos entrando no mesmo; a partir daí, se atravessarmos outra aresta, voltamos para o exterior de $\mathcal{P}$. Portanto, se número de interseções entre l e $\mathcal{P}$ à esquerda de p é ímpar, p está contido no polígono, caso contrário p está no exterior de $\mathcal{P}$ (figura 3.1).

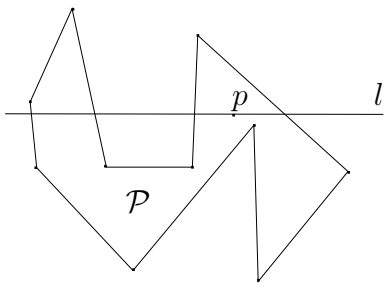


Figura 3.1: Localização de ponto em polígono simples. Como existem 3 interseções entre l e $\mathcal{P}$ de $x = -\infty$ a p, p é interno.

Considerando o caso onde l passa exatamente sobre algum vértice de $\mathcal{P}$, se imaginarmos uma rotação infinitesimal em l a contagem do número de interseções de l com a fronteira de $\mathcal{P}$ será corrigida; assim, uma interseção em um vértice será levada a nenhuma interseção, uma interseção ou duas interseções com arestas de $\mathcal{P}$ (figura 3.2) [PS85].

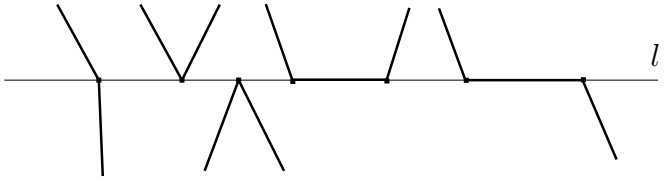


Figura 3.2: Casos onde l passa por vértices de $\mathcal{P}$. Uma rotação infinitesimal de l corrige a contagem das interseções de l com a fronteira de $\mathcal{P}$.

Geometria $\mathcal{G}_\mathcal{F}$ Considere o $\mathcal{G}_\mathcal{F}$-polígono $\mathcal{P}_\mathcal{F}$ e o ponto p dados como entrada do problema. Como estamos em uma geometria de curvas, não podemos escolher uma reta horizontal passando por p, uma vez que essa reta, por não fazer parte da geometria, pode ter mais de uma interseção com uma $\mathcal{G}_\mathcal{F}$-aresta. Porém, ao invés disso, podemos analisar a $\mathcal{G}_\mathcal{F}$-semi-reta $\widetilde{r}$ que parte de p e passa por algum dos vértices de $\mathcal{P}_\mathcal{F}$. Como no caso euclidiano, através de um percurso sobre $\widetilde{r}$ do infinito até p, podemos contar o número de interseções de $\widetilde{r}$ com a fronteira de $\mathcal{P}_\mathcal{F}$. Se esse número for par ou ímpar, podemos decidir se p é externo ou interno a $\mathcal{P}_\mathcal{F}$, exatamente como no caso euclidiano (figura 3.3). Porém, temos que saber se uma interseção deve ou não ser considerada caso ela ocorra em um vértice.

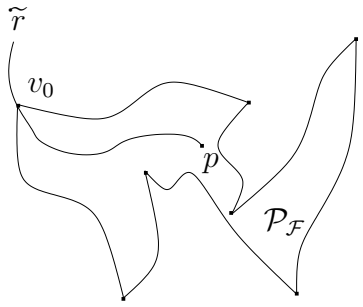


Figura 3.3: Localização de ponto em $\mathcal{G}_\mathcal{F}$-polígono simples. A $\mathcal{G}_\mathcal{F}$-semi-reta $\widetilde{r}$ parte de p e passa por v_0.

Se $\widetilde{r}$ passa por um vértice de $\mathcal{P}_\mathcal{F}$, mas não contém nenhuma das arestas incidentes

neste vértice, então esta interseção deve ser contada se os vértices v_{i-1} e v_{i+1} estiverem em $\mathcal{G}_\mathcal{F}$-semi-planos distintos com relação à reta que contém $\widetilde{r}$. Caso $\widetilde{r}$ contenha uma aresta de vértices v_i e v_{i+1}, a interseção deve ser contada se os vértices v_{i-1} e v_{i+2} estiverem em $\mathcal{G}_\mathcal{F}$-semi-planos distintos com relação à reta que contém $\widetilde{r}$. Vamos considerar que não temos três vértices $\mathcal{G}_\mathcal{F}$-colineares, já que podemos eliminar estes casos com um pré-processamento de tempo $O(n)$.

Como o algoritmo testa a interseção de cada aresta com $\widetilde{r}$, uma interseção que ocorra num vértice e deva ser contada será contada duas vezes, uma para cada aresta que nela incidem. Assim, sempre que a interseção ocorrer num vértice e este for contado, devemos marcá-lo como tal.

Para identificar se dois pontos estão em lados opostos de uma $\mathcal{G}_\mathcal{F}$-reta podemos utilizar a primitiva de orientação de pontos. Se os pontos tiverem a mesma orientação, com relação a dois pontos da $\mathcal{G}_\mathcal{F}$-reta, estão no mesmo $\mathcal{G}_\mathcal{F}$-meio-plano. Caso contrário, estão em $\mathcal{G}_\mathcal{F}$-meios-planos opostos. Assim, o algoritmo 1 determina se um ponto está no interior ou no exterior de um $\mathcal{G}_\mathcal{F}$-polígono.

A complexidade do algoritmo 1 é determinada pela complexidade do laço da linha 4. Como vemos, o laço percorre cada uma das arestas de $\mathcal{P}_\mathcal{F}$. Assim, o algoritmo é executado em tempo $O(n)$.

3.1.2 Polígono Convexo

Geometria Euclidiana Sendo $\mathcal{P}$ um polígono convexo, temos que qualquer segmento entre dois pontos internos a $\mathcal{P}$ está totalmente contido em $\mathcal{P}$. Usando essa propriedade podemos ver que, dado um ponto q interno a $\mathcal{P}$ (por exemplo, o baricentro de um triângulo formado por quaisquer três vértices de $\mathcal{P}$), os vértices de $\mathcal{P}$ estarão em ordem circular em sua volta. Assim, os segmentos que ligam q aos vértices de $\mathcal{P}$ dividem-no em fatias. Construído um vetor comos vértices de $\mathcal{P}$, podemos, através de uma busca binária, saber em que fatia da divisão se encontra p, e, observando a posição de p com relação a única aresta que corta essa fatia, decidir se p é interno ou externo (figura 3.4). Assim, a decisão sobre a localização de um ponto em um polígono pode ser feita em tempo $O(\log n)$, gastando-se tempo $O(n)$ de pré-processamento e $O(n)$ de armazenamento.

Geometria $\mathcal{G}_\mathcal{F}$ Para encontrarmos um algoritmo eficiente na geometria $\mathcal{G}_\mathcal{F}$, podemos tentar encontrar propriedades que nos permitam adaptar o algoritmo euclidiano. Os teoremas seguintes nos ajudam nesse sentido.

Teorema 3.1.1 *Dado um ponto q interno a um $\mathcal{G}_\mathcal{F}$-polígono convexo $\mathcal{P}_\mathcal{F}$, o $\mathcal{G}_\mathcal{F}$-segmento que liga q a qualquer um dos vértices de $\mathcal{P}_\mathcal{F}$ está contido em $\mathcal{P}_\mathcal{F}$.*

Algoritmo 1 Algoritmo de localização de ponto em polígono simples

```
1: procedimento G_F-LOCALIZAÇÃO(gf-polígono P_F[v_0, v_1, ..., v_{n-1}, ẽ_0, ẽ_1, ..., ẽ_{n-1}],
       ponto p)
2:     int i ← 0;
3:     G_F-semireta r̃ ← p⃗v_0
4:     para todo ẽ_i ∈ G_F-arestas de P_F faça
5:         se ẽ_i intercepta r̃ então
6:             q ← r̃ ∩ ẽ_i
7:             se (q ∉ vértice de P_F) então
8:                 i ← i + 1
9:             senão
10:                seja v_j o vértice sobre q
11:                se se v_j ainda não foi contado então
12:                    se r̃ contém a aresta ⌢v_{j-1}v_j então
13:                        se (v_{j-2} e v_{j+1} estão em lados opostos da reta que contém r̃)
                               então
14:                            i ← i + 1
15:                        fim se
16:                    senão se r̃ contém a aresta ⌢v_jv_{j+1} então
17:                        se (v_{j-1} e v_{j+2} estão em lados opostos da reta que contém r̃)
                               então
18:                            i ← i + 1
19:                        fim se
20:                    senão se (v_{j-1} e v_{j+1} estão em lados opostos da reta que contém r̃)
                           então
21:                        i ← i + 1
22:                    fim se
23:                    marque v_j como contado
24:                fim se
25:            fim se
26:        fim se
27:    fim para
28:    se i é impar então
29:        retorne p é interno
30:    senão
31:        retorne p é externo
32:    fim se
33: fim procedimento
```

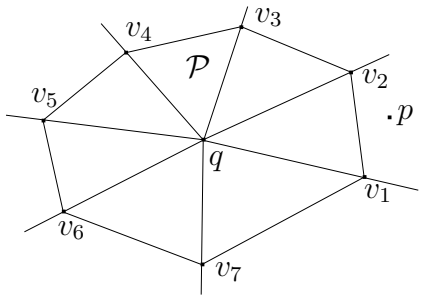


Figura 3.4: Localização de ponto em polígono convexo.

Prova: Segue-se da definição de conjunto $\mathcal{G}_\mathcal{F}$-convexo. ■

Proposição 3.1.1 *Dado um ponto q interno a um $\mathcal{G}_\mathcal{F}$-polígono convexo $\mathcal{P}_\mathcal{F}$, dois $\mathcal{G}_\mathcal{F}$-segmentos que partem de q e chegam a dois vértices distintos não se cruzam.*

Prova: Como os vértices de $\mathcal{P}_\mathcal{F}$ são pontos distintos, duas $\mathcal{G}_\mathcal{F}$-retas que passam por eles são distintas, não podendo se cruzar em outro ponto além de q. ■

Portanto, os $\mathcal{G}_\mathcal{F}$-segmentos que partem de q e passam pelos vértices de $\mathcal{P}_\mathcal{F}$ mantém uma relação de ordem circular em torno de q. Agora, precisamos achar um ponto q, interno ao polígono polígono $\mathcal{P}_\mathcal{F}$ para que este seja usado para gerar os $\mathcal{G}_\mathcal{F}$-segmentos partindo dele até os vértices de $\mathcal{P}_\mathcal{F}$, que irão dividir o polígonos em fatias. A próxima proposição nos ajuda a encontrar este ponto.

Proposição 3.1.2 *Dados um $\mathcal{G}_\mathcal{F}$-triângulo e os pontos médios de seus lados. Escolhidos dois vértices quaisquer e calculada a interseção q das $\mathcal{G}_\mathcal{F}$-retas que ligam cada um destes vértices ao ponto médio do lado oposto a eles, então q é interno ao triângulo (figura 3.5).*

Prova: Um $\mathcal{G}_\mathcal{F}$-segmento $\widetilde{s}$ que liga um vértice v_i de um triângulo ao ponto médio do lado oposto a v_i não possui interseção com as arestas que contêm v_i, pois como os $\mathcal{G}_\mathcal{F}$-segmentos se encontram em v_i não podem se encontrar em outro ponto. Como $\widetilde{s}$ também só pode ter uma interseção com a aresta oposta a v_i, então o segmento $\widetilde{s}$ está totalmente contido no interior do triângulo. O ponto de encontro de dois $\mathcal{G}_\mathcal{F}$-segmentos contidos em um triângulo tem que estar contido no interior do triângulo. ■

Como o polígono $\mathcal{P}_\mathcal{F}$ é convexo, pelo teorema 3.1.1, as arestas de um triângulo formado por quaisquer três de seus vértices tem que estar totalmente contidas em $\mathcal{P}_\mathcal{F}$, e portanto

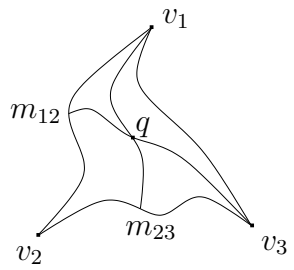


Figura 3.5: O ponto q, interno ao triângulo, é a interseção do $\mathcal{G}_\mathcal{F}$-segmento que parte de v_1 e chega a m_{23} com o $\mathcal{G}_\mathcal{F}$-segmento que parte de v_3 e chega a m_{12}.

o triângulo também estará contido em $\mathcal{P}_\mathcal{F}$. Então, pela proposição 3.1.2 podemos achar um ponto interno ao polígono através de um triângulo formado por quaisquer três de seus vértices.

Agora, precisamos de uma forma de fazermos a busca binária para sabermos em qual "fatia" do polígono está o ponto p. Para isso, podemos observar a fronteira $\partial B(q, \rho)$ da bola com centro em q raio $\rho > 0$ e sua interseção com as $\mathcal{G}_\mathcal{F}$-retas que partem de q. Essas interseções manterão a ordem circular encontrada, e, observando entre quais interseções se encontra a interseção do segmento que liga q a p, saberemos em qual fatia p está.

Por fim, precisamos comparar a posição de p com a aresta que corta esta fatia. Como a fatia formada por $v_i q v_{i+1}$ é convexa, a aresta $\widetilde{v_i v_{i+1}}$ a divide em duas regiões convexas. Assim, para sabermos se p é interno ou externo a $\mathcal{P}_\mathcal{F}$ basta testarmos a orientação do trio de pontos (v_i, v_{i+1}, p). Se a orientação for negativa, p é externo. Caso contrário, p é interno.

Assim, chegamos ao algoritmo 2 para o problema de localização de pontos em polígonos convexos.

Algoritmo 2 Algoritmo de localização de ponto em polígono convexo

1: **procedimento** $\mathcal{G}_\mathcal{F}$-LOCALIZAÇÃO($\mathcal{G}_\mathcal{F}$-polígono convexo $\mathcal{P}_\mathcal{F}[v_0, v_1, \ldots, v_{n-1}]$, ponto p)
2: Encontre um ponto q interno a qualquer $\mathcal{G}_\mathcal{F}$-triângulo formado por 3 vértices distintos de $\mathcal{P}_\mathcal{F}$.
3: Com uma busca binária descubra em qual fatia, definida por v_i e v_{i+1}, p está.
4: **se** $\Delta(v_i, v_{i+1}, p) < 0$ **então**
5: retorne p é externo
6: **senão**
7: retorne p é interno
8: **fim se**
9: **fim procedimento**

Como vemos, a complexidade do algoritmo 2 é dominada pela linha 3. Construída a estrutura conveniente, uma busca binária pode ser feita em tempo $O(\log n)$. Essa construção toma tempo e espaço $O(n)$ de pré-processamento. Feito isso, o algoritmo é executado em tempo $O(\log n)$.

3.1.3 Polígono Estrelado

Definição 3.1.1 *Dado um polígono simples $\mathcal{P}$, dizemos que $\mathcal{P}$ é* estrelado *se existe pelo menos um ponto p no seu interior tal que, para qualquer ponto q na borda de $\mathcal{P}$, o segmento $\overline{pq}$ está inteiramente contido em $\mathcal{P}$. O lugar geométrico dos pontos com essa propriedade é chamado* núcleo *de $\mathcal{P}$.*

Se tivermos um ponto q no núcleo do polígono, poderemos decidir se um dado ponto p é interno ou externo ao polígono de modo análogo ao caso de polígonos convexos, utilizando q como o ponto que, ligado aos vértices do polígono, gera as fatias para a busca binária. Como o núcleo de um polígono pode ser encontrado em tempo $O(n)$ no caso euclidiano, o problema da localização pode ser resolvido em tempo $O(\log n)$ com um tempo $O(n)$ de pré-processamento.

Na seção 3.7 é proposto um algoritmo linear para encontrar o núcleo de um $\mathcal{G}_{\mathcal{F}}$-polígono. Com isso, o problema de localização de pontos em $\mathcal{G}_{\mathcal{F}}$-polígonos estrelados pode ser resolvido de modo análogo ao caso de $\mathcal{G}_{\mathcal{F}}$-polígonos convexos, com tempo e espaço $O(n)$ de pré-processamento e tempo $O(\log n)$ de consulta.

3.2 Interseção de Polígonos Convexos

O problema da interseção de polígonos convexos pode ser definido como:

Problema 2 *Dados dois polígonos convexos $\mathcal{P}$ e $\mathcal{Q}$ com n e m vértices, respectivamente, encontrar a interseção entre $\mathcal{P}$ e $\mathcal{Q}$.*

Na seção 3.2.1 serão apresentados dois algoritmos existentes para o problema no caso euclidiano. A seguir, na seção 3.2.2, são propostos dois algoritmos para a geometria $\mathcal{G}_{\mathcal{F}}$.

3.2.1 Geometria Euclidiana

Teorema 3.2.1 *A interseção de dois polígonos convexos $\mathcal{P}$ e $\mathcal{Q}$, com m e n lados, respectivamente, possui no máximo $m + n$ lados.*

Prova: A interseção dos polígonos é a interseção dos $m+n$ meios-planos determinados por suas arestas. ■

É facil observar que a fronteira de $\mathcal{P} \cap \mathcal{Q}$ é formada por vértices dos dois polígonos e por vértices que são interseções de arestas dos dois polígonos. Assim, um algoritmo direto para encontrarmos a interseção seria percorrer a borda de $\mathcal{P}$ e, para cada aresta, calcular sua interseção com as arestas de $\mathcal{Q}$, mantendo a relação dos vértices e interseções visitados que formam a interseção. Esse algoritmo tem complexidade $O(mn)$ pois cada aresta de $\mathcal{P}$ precisa ser comparada com cada aresta de $\mathcal{Q}$ para determinar se elas se interceptam. Porém, este algoritmo é pouco eficiente. A seguir são apresentados dois algoritmos de complexidade $O(m+n)$.

Algoritmo de Shamos e Hoey

Uma possível abordagem para achar a interseção de polígonos convexos é dividir o plano em regiões, tal que seja fácil calcular a interseção dos polígonos em cada uma delas.

Shamos e Hoey [SH76] se basearam nesse fato e propuseram a partição do plano por retas verticais passando por cada vértice de $\mathcal{P}$ e $\mathcal{Q}$. Em cada "fatia" do plano criada, a interseção da mesma com um polígono convexo é um trapézio ou um triângulo (como ilustrado na figura 3.6). A interseção de dois quadriláteros dentro de uma mesma fatia pode ser calculada em tempo constante. Assim, podemos percorrer todas as fatias unindo as interseções calculadas em cada uma delas. Como os polígonos têm juntos $m+n$ vértices, criando no máximo $O(m+n)$ fatias, o algoritmo faz a interseção dos polígonos em tempo $O(m+n)$.

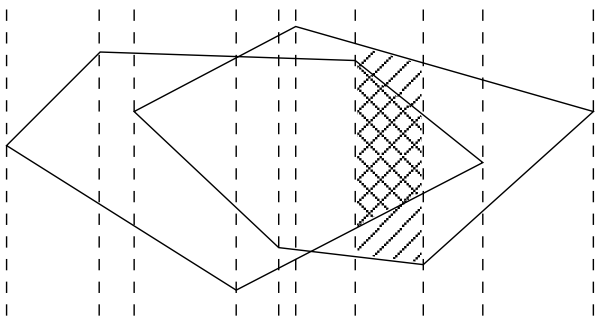


Figura 3.6: Algoritmo de Shamos e Hoey. O plano é fatiado por retas verticais.

Algoritmo de O'Rourke, Chien, Olson e Naddor

Uma segunda abordagem, proposta por O'Rourke, Chien, Olson e Naddor [OCON82], faz uso diretamente do fato de que a fronteira da interseção é formada por uma seqüência de vértices dos polígonos e pontos de interseção de suas arestas. Se uma parte de $\mathcal{P} \cap \mathcal{Q}$ é

formada por uma seqüência de vértices de, por exemplo, $\mathcal{P}$, então $\mathcal{Q}$ passa por fora dessa seqüência de vértices, formando uma *orelha* (figura 3.7). Essas orelhas podem ser bem caracterizadas através de um ponto inicial p_i e um ponto final p_f, que são pontos onde as bordas dos polígonos se interceptam. Denominamos *cadeia interna* da orelha a cadeia de arestas que pertence a $\mathcal{P} \cap \mathcal{Q}$ entre p_i e p_f. Da mesma forma, a cadeia de arestas que não pertence a $\mathcal{P} \cap \mathcal{Q}$ entre p_i e p_f é denominada *cadeia externa* da orelha.

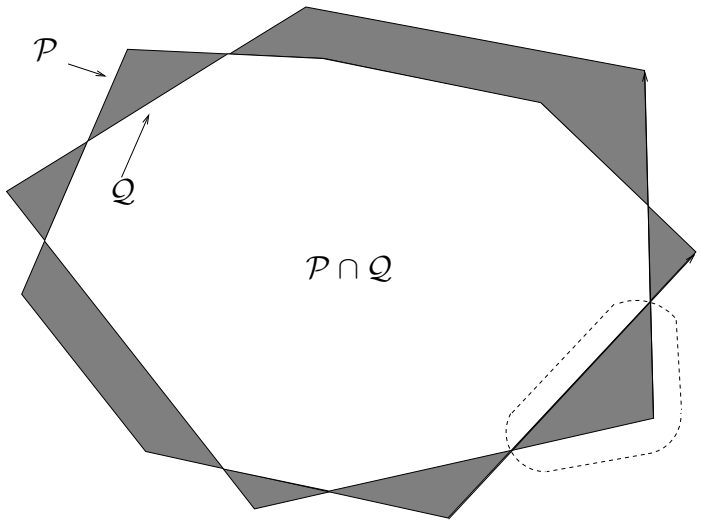


Figura 3.7: Algoritmo de O'Rourke, Chien, Olson e Naddor. As áreas sombreadas são as áreas que queremos eliminar para ficarmos com $\mathcal{P} \cap \mathcal{Q}$. Em pontilhado é mostrada uma orelha.

Através de um mecanismo para percorrer os vértices dos polígonos, podemos achar os pontos iniciais e finais das orelhas, decidindo qual polígono contém a cadeia de vértices interna e qual contém a externa. Com essa informação, podemos excluir as cadeias indesejadas, aparando as orelhas e chegando a $\mathcal{P} \cap \mathcal{Q}$.

O mecanismo buscado se baseia no posicionamento relativo de uma aresta em $\mathcal{P}$ e uma aresta em $\mathcal{Q}$. Suponha que estamos percorrendo os polígonos no sentido anti-horário e sejam $\overline{e}_{\mathcal{P}}$ e $\overline{e}_{\mathcal{Q}}$ as duas arestas em $\mathcal{P}$ e $\mathcal{Q}$ respectivamente, as quais analisaremos quanto a posição (note que agora estamos direcionando as arestas). O algoritmo avança $\overline{e}_{\mathcal{P}}$ ou $\overline{e}_{\mathcal{Q}}$ garantindo que sempre encontremos uma interseção entre as arestas de $\mathcal{P}$ e $\mathcal{Q}$.

Sejam p o ponto final da aresta $\overline{e}_{\mathcal{P}}$ e q o ponto final da aresta $\overline{e}_{\mathcal{Q}}$. Se $\overline{e}_{\mathcal{Q}}$ vai em direção à reta que contém $\overline{e}_{\mathcal{P}}$, mas não a cruza, então queremos avançar $\overline{e}_{\mathcal{Q}}$ de forma a nos aproximar de uma possível interseção com $\overline{e}_{\mathcal{P}}$. Essa é a idéia principal do mecanismo de percurso dos vértices.

Considerando $H(\overline{e}_{\mathcal{P}})$ o meio-plano fechado à esquerda da aresta $\overline{e}_{\mathcal{P}}$ podemos tomar a decisão de qual aresta avançar observando o valor de $\overline{e}_{\mathcal{P}} \times \overline{e}_{\mathcal{Q}}$ e se $H(\overline{e}_{\mathcal{P}})$ contém q ou não. A tabela 3.1 nos mostra todas as regras de avanço das arestas (veja [OCON82]).

$\overline{e}_{\mathcal{P}} \times \overline{e}_{\mathcal{Q}}$	Condição de meio-plano	Quem avança
> 0	$q \in H(\overline{e}_{\mathcal{P}})$	$\overline{e}_{\mathcal{P}}$
> 0	$q \notin H(\overline{e}_{\mathcal{P}})$	$\overline{e}_{\mathcal{Q}}$
< 0	$p \in H(\overline{e}_{\mathcal{Q}})$	$\overline{e}_{\mathcal{Q}}$
< 0	$p \notin H(\overline{e}_{\mathcal{Q}})$	$\overline{e}_{\mathcal{P}}$

Tabela 3.1: Regras para decidirmos qual aresta avançar.

Se tanto $\overline{e}_{\mathcal{P}}$ quanto $\overline{e}_{\mathcal{Q}}$ forem em direção uma à outra, podemos avançar qualquer uma delas. Se nenhuma delas for em direção à outra, devemos avançar aquela que não está no meio-plano fechado à esquerda da outra, ou as duas se nenhuma satisfizer essa condição. Como o algoritmo percorre cada fronteira dos polígonos passando de aresta em aresta em tempo constante, o algoritmo tem complexidade $O(m + n)$. Uma implementação desse algoritmo está descrita em [O'R94].

3.2.2 Geometria $\mathcal{G}_{\mathcal{F}}$

Passemos agora ao problema sobre a geometria $\mathcal{G}_{\mathcal{F}}$. Aqui, serão apresentados dois algoritmos para o problema de interseção de $\mathcal{G}_{\mathcal{F}}$-polígonos convexos, ambos com complexidade $O(m + n)$, onde m e n são o número de arestas dos $\mathcal{G}_{\mathcal{F}}$-polígonos de entrada.

Algoritmo de Shamos e Hoey

Quando tentamos aplicar o algoritmo proposto por Shamos e Hoey sobre a geometria $\mathcal{G}_{\mathcal{F}}$ vemos que a divisão do plano por retas verticais não é mais factível, uma vez que, dependendo da escolha da família $\mathcal{F}$, essa divisão pode gerar regiões não $\mathcal{G}_{\mathcal{F}}$-convexas. A proposição a seguir nos ajuda a encontrar uma partição do plano em regiões $\mathcal{G}_{\mathcal{F}}$-convexas.

Proposição 3.2.1 *$\mathcal{G}_{\mathcal{F}}$-raios partindo de um ponto p dividem o plano em regiões $\mathcal{G}_{\mathcal{F}}$-convexas.*

Prova: Se alguma das regiões formadas não fosse $\mathcal{G}_{\mathcal{F}}$-convexa, deveria existir uma $\mathcal{G}_{\mathcal{F}}$-reta cuja interseção com essa região fosse não conexa. Como a fronteira da região são $\mathcal{G}_{\mathcal{F}}$-raios, se a interseção não fosse conexa, teríamos duas interseções entre duas $\mathcal{G}_{\mathcal{F}}$-retas, o que é impossível pela definição da geometria. ■

Consideremos, então, o particionamento do plano através de $\mathcal{G}_\mathcal{F}$-raios que partem de um ponto fora da envoltória $\mathcal{G}_\mathcal{F}$-convexa de um $\mathcal{G}_\mathcal{F}$-polígono $\mathcal{P}_\mathcal{F}$ e passam pelos vértices de $\mathcal{P}_\mathcal{F}$ (figura 3.8). Harada [Har00] propos um algoritmo linear para encontrar um ponto fora da envoltória $\mathcal{G}_\mathcal{F}$-convexa de um $\mathcal{G}_\mathcal{F}$-polígono.

Proposição 3.2.2 *Sejam um ponto p_0 externo a um $\mathcal{G}_\mathcal{F}$-polígono convexo $\mathcal{P}_\mathcal{F}$, e dois vértices v_1 e v_2 de $\mathcal{P}_\mathcal{F}$, com p_0, v_1 e v_2 não $\mathcal{G}_\mathcal{F}$-colineares. Então, os dois $\mathcal{G}_\mathcal{F}$-segmentos que partem de p e chegam v_1 e v_2, mantêm uma relação de ordem circular em torno de p_0.*

Prova: Uma vez que v_1 e v_2 são pontos distintos e não simultaneamente $\mathcal{G}_\mathcal{F}$-colineares com p_0, duas $\mathcal{G}_\mathcal{F}$-retas, uma que passa por p_0 e v_1 e outra passando por p_0 e v_2 são distintas. Além disso, por cada dois pontos passa apenas uma $\mathcal{G}_\mathcal{F}$-reta, então, dadas duas $\mathcal{G}_\mathcal{F}$-retas, elas se cruzam em apenas um ponto. Já que as $\mathcal{G}_\mathcal{F}$-retas que contêm $\widetilde{p_0v_1}$ e $\widetilde{p_0v_2}$ se cruzam em p_0, elas não podem se cruzar em outro ponto, mantendo uma relação de ordem circular em torno de p_0 (figura 3.8). ■

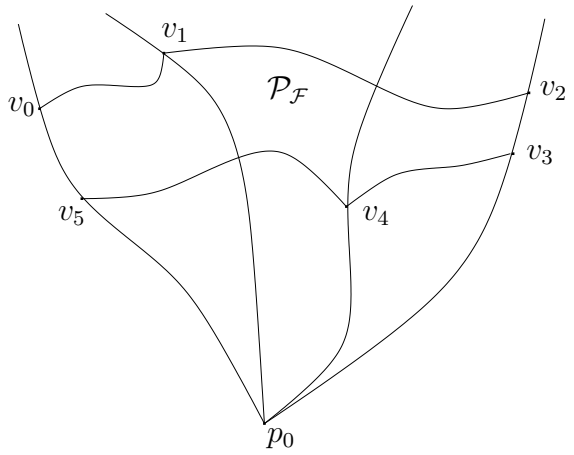


Figura 3.8: $\mathcal{G}_\mathcal{F}$-raios partindo de p_0 e passando pelos vértices de $\mathcal{P}_\mathcal{F}$ criando uma partição de $\mathbb{R}^2$ em regiões $\mathcal{G}_\mathcal{F}$-convexas.

Sejam $\mathcal{P}_\mathcal{F}$ e $\mathcal{Q}_\mathcal{F}$ os $\mathcal{G}_\mathcal{F}$-polígonos convexos dados na entrada do problema de interseção. Baseado na proposição 3.2.2 será descrito um algoritmo linear no número de vértices de cada polígono para encontrar os dois vértices extremos de $\mathcal{P}_\mathcal{F}$ e $\mathcal{Q}_\mathcal{F}$. Sendo p_0 um ponto fora de $\mathcal{P}_\mathcal{F}$, um *vértice extremo* é tal que, dada a $\mathcal{G}_\mathcal{F}$-reta ℓ passando por ele e por p_0, $\mathcal{P}_\mathcal{F}$ está totalmente contido em um dos $\mathcal{G}_\mathcal{F}$-meios-planos definidos por ℓ. Esse algoritmo foi primeiramente proposto por Harada [Har00].

Sejam P o conjunto formado pelos vértices de $\mathcal{P}_\mathcal{F}$ e $\mathcal{Q}_\mathcal{F}$, e ℓ_{p_0} uma $\mathcal{G}_\mathcal{F}$-reta que separa P de p_0 (figura 3.9). Queremos, com auxílio de p_0 e ℓ_{p_0} "fatiar" $\mathcal{P}_\mathcal{F}$ e $\mathcal{Q}_\mathcal{F}$ com $\mathcal{G}_\mathcal{F}$-raios partindo de p_0 e passando por cada um de seus vértices. A $\mathcal{G}_\mathcal{F}$-reta ℓ_{p_0} será importante para conhecermos quais $\mathcal{G}_\mathcal{F}$-raios limitam tais fatias, e em qual ordem essas fatias ocorrem. Primeiramente vamos descrever como criar as fatias somente sobre $\mathcal{P}_\mathcal{F}$. As fatias sobre $\mathcal{Q}_\mathcal{F}$ podem ser criadas de modo análogo.

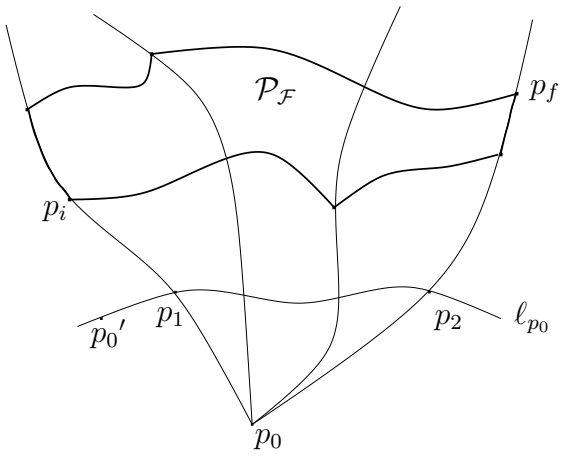


Figura 3.9: Os pontos p_i e p_f são os pontos extremos de $\mathcal{P}_\mathcal{F}$.

Seja P_1 o conjunto de todas as interseções de ℓ_{p_0} com todas as $\mathcal{G}_\mathcal{F}$-retas passando por p_0 e por vértices de $\mathcal{P}_\mathcal{F}$. Seja $p_0{}'$ um ponto em ℓ_{p_0} fora da área de interesse. Defina p_1 e p_2 da seguinte maneira:

$$p_1 = p \in P_1 : p \prec^{\ell_{p_0}}_{p_0{}'} q, \text{ para todo } q \in P_1$$
$$p_2 = p \in P_1 : q \prec^{\ell_{p_0}}_{p_0{}'} p, \text{ para todo } q \in P_1$$

Ou seja, p_1 e p_2 são, respectivamente, o ponto de P_1 mais próximo e mais distante de $p_0{}'$ (figura 3.9). Seja p_i (p_f) um ponto de P na $\mathcal{G}_\mathcal{F}$-reta ℓ_{p_0,p_1} (ℓ_{p_0,p_2}) tal que:

$$p_i = p \in \ell_{p_0,p_1} \text{ e } p \in P : p \prec^{\ell_{p_0,p_1}}_{p_0{}'} q, \text{ para todo } q \in P \text{ e } q \in \ell_{p_0,p_1}$$
$$p_f = p \in \ell_{p_0,p_2} \text{ e } p \in P : q \prec^{\ell_{p_0,p_2}}_{p_0{}'} p, \text{ para todo } q \in P \text{ e } q \in \ell_{p_0,p_2}$$

Isto é, o ponto p_i é o vértice de $\mathcal{P}_\mathcal{F}$ mais próximo a p_1 sobre ℓ_{p_0,p_1}. Já p_f é o ponto mais distante de p_2 sobre ℓ_{p_0,p_2} (figura 3.9). Como demonstrado em [Har00], as retas ℓ_{p_0,p_1} e ℓ_{p_0,p_2} são retas de suporte de $\mathcal{P}_\mathcal{F}$, e, portanto, os pontos p_i e p_f são os pontos extremos de $\mathcal{P}_\mathcal{F}$. Qualquer interseção entre $\mathcal{G}_\mathcal{F}$-raios partindo de p_0 e passando por vértice de $\mathcal{P}_\mathcal{F}$ com a reta ℓ_{p_0}, pela proposição 3.2.2, só pode estar entre p_1 e p_2. Assim, p_1 e p_2 delimitam

o início e o fim das fatias de $\mathcal{P}_\mathcal{F}$. Agora, passaremos a analisar as características das interseções de uma fatia com $\mathcal{P}_\mathcal{F}$.

Proposição 3.2.3 *[Har00] A interseção de conjuntos $\mathcal{G}_\mathcal{F}$-convexos é um conjunto $\mathcal{G}_\mathcal{F}$-convexo.*

Decorre desta proposição que a interseção de uma das fatias do plano com $\mathcal{P}_\mathcal{F}$ é ainda um $\mathcal{G}_\mathcal{F}$-polígono convexo. Além disso, é fácil observar (como mostrado na figura 3.8) que essa região é formada por dois segmentos de $\mathcal{G}_\mathcal{F}$-arestas de $\mathcal{P}_\mathcal{F}$ limitados entre os $\mathcal{G}_\mathcal{F}$-raios que formam a partição, e por um ou dois segmentos destes $\mathcal{G}_\mathcal{F}$-raios, sendo um $\mathcal{G}_\mathcal{F}$-polígono convexo com 3 ou 4 lados. Porém, precisamos de um modo de conhecermos cada vértice que forma cada um desses polígonos, para que possamos obter as suas interseções posteriormente.

Como assumido no início, os vértices dos $\mathcal{G}_\mathcal{F}$-polígonos de entrada são dados no sentido anti-horário. Dados os pontos extremos de $\mathcal{P}_\mathcal{F}$ encontrados, vemos que estes dividem $\mathcal{P}_\mathcal{F}$ em duas cadeias ordenadas de vértices, uma que vai de p_i a p_f e outra que vai de p_f a p_i. Olhando para os pontos de P_1 sobre a $\mathcal{G}_\mathcal{F}$-reta ℓ_{p_0} descritos anteriormente, cada uma das cadeias gera uma seqüência de interseções ordenadas em um sentido, uma de p_1 a p_2 e outra de p_2 a p_1. Através da intercalação da seqüência de interseções da primeira cadeia, com o inverso da segunda, temos uma seqüência ordenada de interseções sobre ℓ_{p_0} de p_1 a p_2. De forma análoga, podemos achar as interseções geradas por $\mathcal{G}_\mathcal{F}$-raios partindo de p_0 e passando pelos vértices de $\mathcal{Q}_\mathcal{F}$ sobre ℓ_{p_0}. Como as duas seqüências estão ordenadas, é possível fazer a intercalação em tempo linear.

Através de um percurso de p_1 a p_2 podemos identificar as interseções dos $\mathcal{G}_\mathcal{F}$-raios partindo de p_0 com as $\mathcal{G}_\mathcal{F}$-arestas dos dois $\mathcal{G}_\mathcal{F}$-polígonos. Como argumentado anteriormente, a interseção da região entre dois desses $\mathcal{G}_\mathcal{F}$-raios e qualquer um dos dois $\mathcal{G}_\mathcal{F}$-polígonos forma um $\mathcal{G}_\mathcal{F}$-polígono convexo de 3 ou 4 lados.

Como em cada fatia temos no máximo dois desses $\mathcal{G}_\mathcal{F}$-polígonos, sendo um gerado por $\mathcal{P}_\mathcal{F}$ e outro por $\mathcal{Q}_\mathcal{F}$, podemos, comparando a distância de seus vértices ao ponto p_0, achar a interseção entre os dois $\mathcal{G}_\mathcal{F}$-polígonos em uma fatia em tempo constante.

Por fim, com um novo percurso de p_1 a p_2 sobre ℓ_{p_0}, analisando cada fatia gerada pelos $\mathcal{G}_\mathcal{F}$-raios que passam pelos vértices de $\mathcal{P}_\mathcal{F}$ e $\mathcal{Q}_\mathcal{F}$, e fazendo a união das interseções dos polígonos em cada fatia, chegamos a $\mathcal{P}_\mathcal{F} \cap \mathcal{Q}_\mathcal{F}$.

Prova de Corretude

De acordo com a proposição 3.2.2 é possível dividir qualquer $\mathcal{G}_\mathcal{F}$-polígono convexo em fatias formadas por $\mathcal{G}_\mathcal{F}$-raios que partem de um ponto fora do $\mathcal{G}_\mathcal{F}$-polígono e passam pelos vértices do mesmo. Essas fatias mantêm uma relação de ordem circular em torno do ponto usado para gerá-las, o que possibilita um percurso em ordem. Como mostrado na

proposição 3.2.3 e argumentado a seguir, a interseção do $\mathcal{G}_\mathcal{F}$-polígono com cada uma das fatias é um $\mathcal{G}_\mathcal{F}$-polígono convexo formado por 3 ou 4 lados.

Portanto, para qualquer instância de entrada, sempre é possível gerar as fatias, definir a interseção de cada uma delas com os $\mathcal{G}_\mathcal{F}$-polígonos dados, calcular a interseção dentro de cada uma das fatias e fazer a união das interseções resultantes. Como a partição inicial não elimina nenhuma parte dos polígonos de entrada, a união das interseções finais é $\mathcal{P}_\mathcal{F} \cap \mathcal{Q}_\mathcal{F}$.

Análise de Complexidade

A primeira parte do algoritmo para encontrar a interseção de $\mathcal{G}_\mathcal{F}$-polígonos convexos exige que se encontre um ponto p_0 fora da envoltória convexa de ambos $\mathcal{G}_\mathcal{F}$-polígonos. O teorema 3.2.2 nos garante que isto pode ser feito em tempo $O(m+n)$.

Teorema 3.2.2 *[Har00] Um ponto fora da envoltória $\mathcal{G}_\mathcal{F}$-convexa de um conjunto de n pontos pode ser encontrado em tempo linear.*

Podemos encontrar a $\mathcal{G}_\mathcal{F}$-reta ℓ_{p_0} que separa p_0 dos $\mathcal{G}_\mathcal{F}$-polígonos e encontrar cada interseção de ℓ_{p_0} com as $\mathcal{G}_\mathcal{F}$-retas que ligam p_0 aos vértices dos $\mathcal{G}_\mathcal{F}$-polígonos em tempo $O(m+n)$. Nessa mesma ordem de tempo, podemos encontrar os pontos p_1, p_2, p_i e p_f.

A próxima etapa exige que encontremos sobre ℓ_{p_0} a seqüência ordenada de interseções entre p_1 e p_2. Como os vértices dos dois $\mathcal{G}_\mathcal{F}$-polígonos são dados em ordem circular, só precisamos fazer a união de duas cadeias de vértices ordenadas. Isso também pode ser feito em tempo $O(m+n)$.

A parte menos intuitiva do algoritmo é encontrar as interseções das $\mathcal{G}_\mathcal{F}$-retas que ligam p_0 aos vértices v_i dos $\mathcal{G}_\mathcal{F}$-polígonos com as $\mathcal{G}_\mathcal{F}$-arestas dos mesmos. Para evitar testar cada $\mathcal{G}_\mathcal{F}$-aresta com cada $\mathcal{G}_\mathcal{F}$-reta, pois isso tomaria tempo $\Omega((m+n)^2)$, considere a $\mathcal{G}_\mathcal{F}$-reta que contém $\widetilde{p_0v_i}$ e sua interseção p_{v_i} com $\widetilde{p_1p_2}$ (figura 3.10). Como os $\mathcal{G}_\mathcal{F}$-polígonos são convexos, a interseção $p_{v_{i+1}}$ da $\mathcal{G}_\mathcal{F}$-reta que contém $\widetilde{p_0v_{i+1}}$ com $\widetilde{p_1p_2}$ não pode estar contida em $\widetilde{p_1p_{v_i}}$. Por isso, dada uma seqüência do tipo $\{p_{v_i}, p_{v_{k1}}, p_{v_{k2}}, \ldots, p_{v_{kn}}, p_{v_{i+1}}\}$ sobre $\widetilde{p_1p_2}$, sabemos que as $\mathcal{G}_\mathcal{F}$-retas que contêm os $\mathcal{G}_\mathcal{F}$-segmentos $\widetilde{p_0p_{v_{kj}}}$, com $j \in [1,n]$ interceptam a $\mathcal{G}_\mathcal{F}$-aresta $\widetilde{v_iv_{i+1}}$. Assim, através de apenas um percurso sobre $\widetilde{p_1p_2}$ podemos encontrar todas as fatias do plano em tempo $O(m+n)$.

Encontradas as fatias e os pontos que limitam cada parte dos $\mathcal{G}_\mathcal{F}$-polígonos no seu interior, podemos achar as interseções nas fatias, levando tempo constante em cada uma. Por fim, percorrendo cada fatia a partir de p_1 até p_2 podemos juntar todas as interseções encontradas, formando a interseção final em tempo $O(m+n)$.

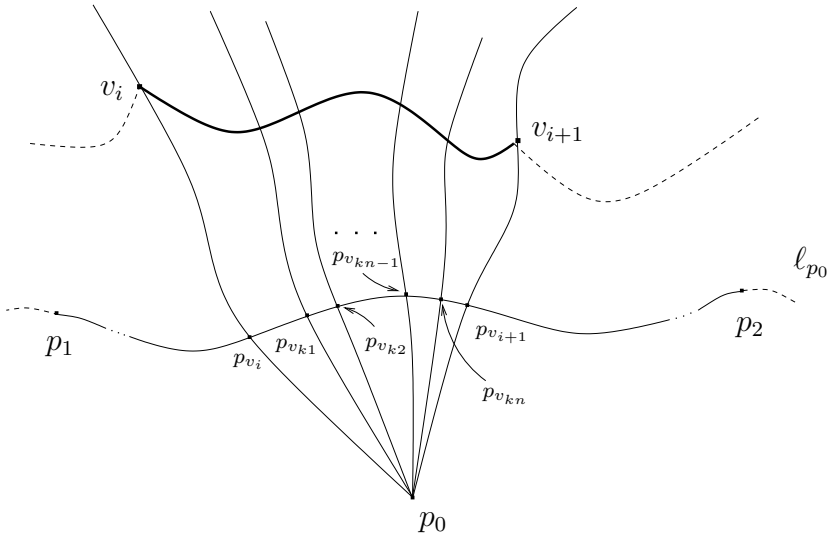


Figura 3.10: Os $\mathcal{G}_\mathcal{F}$-segmentos $\widetilde{p_0 p_{v_{kj}}}$, com $j \in [1, n]$ interceptam a $\mathcal{G}_\mathcal{F}$-aresta $\widetilde{v_i v_{i+1}}$.

Algoritmo de O'Rourke, Chien, Olson e Naddor

A estrutura geral do algoritmo proposta em [OCON82], mostrada no algoritmo 3, pode ser mantida na geometria $\mathcal{G}_\mathcal{F}$. Entretanto, a decisão de qual aresta avançar não pode ser feita da mesma maneira, como discutiremos a seguir.

Quando tratamos problemas na geometria euclidiana, o produto vetorial é uma ferramenta importante. Com ele, podemos calcular facilmente áreas de figuras como triângulos ou trapézios, ou mesmo o volume de um paralelepípedo. Mas, muitas vezes, saber apenas a magnitude do produto vetorial já é suficiente para podermos tomar decisões em nossos algoritmos. Um exemplo surge no algoritmo apresentado em [OCON82] para encontrar a interseção de polígonos convexos. Nele, uma decisão precisa ser tomada a partir da informação do posicionamento relativo de dois vetores. Isto é, dados dois vetores $\vec{v}_1$ e $\vec{v}_2$, precisamos saber se $\vec{v}_1$ vai em direção à reta que contém $\vec{v}_2$, $\vec{v}_2$ vai em direção à reta que contém $\vec{v}_1$ ou nenhuma das duas situações (nesse caso, os vetores são paralelos).

Essa informação, que é obtida através do produto vetorial dos dois vetores, junto com a posição do ponto final do vetor, nos possibilita decidir qual vetor avançar no algoritmo de O'Rourke et al.

Quando passamos para a geometria $\mathcal{G}_\mathcal{F}$, por esta ser uma geometria de curvas, não faz sentido pensarmos no produto vetorial entre dois segmentos de $\mathcal{G}_\mathcal{F}$-retas. Assim, precisamos de um outro método para sabermos se um $\mathcal{G}_\mathcal{F}$-segmento vai em direção a outro, para saber sobre qual $\mathcal{G}_\mathcal{F}$-aresta devemos avançar.

Algoritmo 3 Algoritmo para calcular a interseção de $\mathcal{G}_{\mathcal{F}}$-polígonos convexos.

procedimento INTERSEÇÃO-DE-POLÍGONOS-CONVEXOS($\mathcal{P}_{\mathcal{F}}$, $\mathcal{Q}_{\mathcal{F}}$)
    $k \leftarrow 1, i \leftarrow 1, j \leftarrow 1$
    repita
        se $\widetilde{p_{i-1}p_i}$ e $\widetilde{q_{j-1}q_j}$ se interceptam **então**
            imprima a interseção
            atualize a informação de qual polígono contém a cadeia interna
        fim se
        AVANCE (* i ou j é incrementado *)
        $k \leftarrow k + 1$
    até $k = 2(m + n)$
    se (nenhuma interseção foi encontrada) **então**
        se ($p_i \in \mathcal{Q}_{\mathcal{F}}$) **então**
            retorne $\mathcal{P}_{\mathcal{F}}$ (* pois $\mathcal{P}_{\mathcal{F}} \subseteq \mathcal{Q}_{\mathcal{F}}$ *)
        senão
            se ($q_j \in \mathcal{P}_{\mathcal{F}}$) **então**
                retorne $\mathcal{Q}_{\mathcal{F}}$ (* pois $\mathcal{Q}_{\mathcal{F}} \subseteq \mathcal{P}_{\mathcal{F}}$ *)
            senão
                retorne $\mathcal{P}_{\mathcal{F}} \cap \mathcal{Q}_{\mathcal{F}} = \emptyset$
            fim se
        fim se
    fim se
fim procedimento

Dados os pontos inicial p_1 e final p_2 de um $\mathcal{G}_\mathcal{F}$-segmento orientado $\widetilde{s}$ e uma $\mathcal{G}_\mathcal{F}$-reta ℓ_1, queremos saber se $\widetilde{s}$ vai em direção a ℓ_1. Isso pode ser verificado através da relação de precedência entre os pontos p_1 e p_2 e o ponto de interseção p de ℓ_1 com a $\mathcal{G}_\mathcal{F}$-reta ℓ_2 que contém $\widetilde{s}$. O $\mathcal{G}_\mathcal{F}$-segmento $\widetilde{s}$ vai em direção a ℓ se, e somente se, $p_2 \prec_{p_1}^{\ell_2} p$.

O algoritmo 4 nos mostra qual aresta avançar na linha 8 do algoritmo 3.

Algoritmo 4 Algoritmo de O'Rourke et al. para encontrar a interseção de polígonos convexos.

1: **procedimento** AVANCE($\mathcal{G}_\mathcal{F}$-aresta de $\mathcal{P}_\mathcal{F}[\widetilde{p_{i-1}p_i}]$, $\mathcal{G}_\mathcal{F}$-aresta de $\mathcal{Q}_\mathcal{F}[\widetilde{q_{j-1}q_j}]$)
2: **se** $\widetilde{p_{i-1}p_i}$ vai em direção a $\widetilde{q_{j-1}q_j}$ e $\widetilde{q_{j-1}q_j}$ vai em direção a $\widetilde{p_{i-1}p_i}$ **então**
3: $i \leftarrow i + 1$ (* alternativamente, poderiamos avançar j *)
4: **senão se** $\widetilde{p_{i-1}p_i}$ vai em direção a $\widetilde{q_{j-1}q_j}$ **então**
5: se p_i contém a cadeia interna, imprima p_i
6: $i \leftarrow i + 1$
7: **senão se** $\widetilde{q_{j-1}q_j}$ vai em direção a $\widetilde{p_{i-1}p_i}$ **então**
8: se q_j contém a cadeia interna, imprima q_j
9: $j \leftarrow j + 1$
10: **senão**
11: **se** p_i está à direita de $\widetilde{q_{j-1}q_j}$ **então**
12: se p_i contém a cadeia interna, imprima p_i
13: $i \leftarrow i + 1$
14: **senão**
15: se q_j contém a cadeia interna, imprima q_j
16: $j \leftarrow j + 1$
17: **fim se**
18: **fim se**
19: **fim procedimento**

O algoritmo 3 imprime na saída apenas as interseções encontradas entre os $\mathcal{G}_\mathcal{F}$-polígonos. Essas interseções são exatamente os pontos iniciais e finais das orelhas procuradas. A definição de orelha é análoga ao caso euclidiano, como descrito na seção 3.2.1. Porém, quando identificada uma orelha é necessário saber também qual dos $\mathcal{G}_\mathcal{F}$-polígonos possui a cadeia interna da mesma, pois a cadeia de $\mathcal{G}_\mathcal{F}$-arestas deste polígono, na orelha, faz parte da interseção procurada, e deve ser mostrada na saída também.

Supondo que uma interseção entre as $\mathcal{G}_\mathcal{F}$-arestas $\widetilde{p_{i-1}p_i}$ e $\widetilde{q_{j-1}q_j}$ foi encontrada e sendo $H(\widetilde{e})$ o $\mathcal{G}_\mathcal{F}$-meio-plano fechado à esquerda de $\widetilde{e}$, p_i pertence à cadeia interna se, e somente se, $p_i \in H(\widetilde{q_{j-1}q_j})$. Do mesmo modo, q_j pertence à cadeia interna se, e somente se, $q_j \in H(\widetilde{p_{i-1}p_i})$. Uma vez encontrado o polígono que possui a cadeia interna, a cada nova interseção entre duas $\mathcal{G}_\mathcal{F}$-arestas, este passará a conter a cadeia externa da próxima orelha. Note que o caso onde a interseção ocorre sobre um vértice de um dos $\mathcal{G}_\mathcal{F}$-polígonos pode fazer com que essa inversão não ocorra, nos obrigando a fazer uma nova verificação para

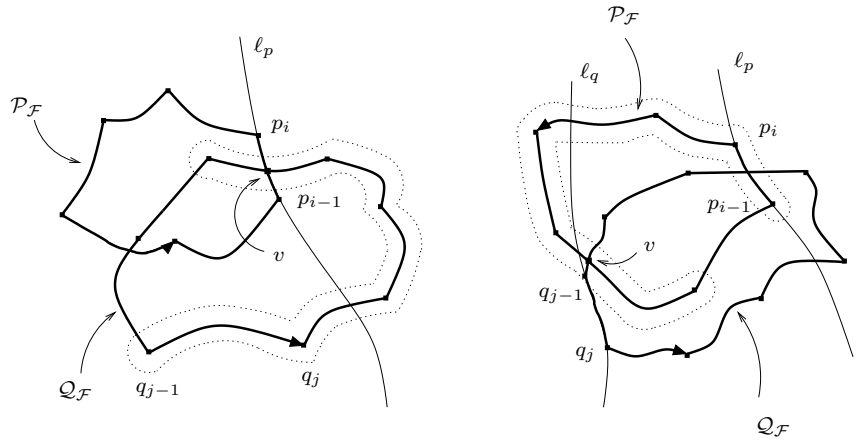
saber qual dos polígonos possui a cadeia interna da próxima orelha. A cadeia interna da orelha é impressa pelo algoritmo 4.

Prova de Corretude

Dados dois vértices pertencentes a uma mesma orelha, através da escolha de qual vértice avançar no algoritmo AVANCE, temos a garantia de que a interseção final da orelha será encontrada, uma vez que nunca é feito o avanço de uma aresta que pode conter tal interseção. Isso garante que, achados dois vértices p_i e q_j pertencentes a uma mesma orelha, todas as orelhas são encontradas sucessivamente.

Para completar a prova, é preciso mostrar que, se $\mathcal{P}_\mathcal{F}$ e $\mathcal{Q}_\mathcal{F}$ se interceptam, então o algoritmo encontra uma interseção, mesmo começando com p_i e q_j em orelhas diferentes.

Seja $\widetilde{p_{i-1}p_i}$, a aresta corrente de $\mathcal{P}_\mathcal{F}$, uma aresta que contém uma interseção com alguma aresta de $\mathcal{Q}_\mathcal{F}$ (figura 3.11). Seja ℓ_p a $\mathcal{G}_\mathcal{F}$-reta que contém $\widetilde{p_{i-1}p_i}$. É fácil perceber que, se $\widetilde{q_{j-1}q_j}$ vai em direção a ℓ_p, então $\widetilde{q_{j-1}q_j}$ será avançada até que uma interseção seja encontrada. Isso ocorre pois mesmo que a aresta cruze a reta, ela continuará sendo avançada por q_j estar à direita de $\widetilde{p_{i-1}p_i}$ [figura 3.11(a)]. Porém, a situação onde $\widetilde{q_{j-1}q_j}$ não vai em direção a ℓ_p deve ser analisada com mais cuidado. Antes da análise deste caso, é importante lembrar que as interseções ocorrem sempre aos pares (no caso de apenas uma interseção, esta pode ser considerada ponto inicial e final de uma orelha).



(a) (b)

Figura 3.11: Ilustração da prova de corretude do algoritmo para encontrar a interseção de dois $\mathcal{G}_\mathcal{F}$-polígonos. (a) $\widetilde{q_{j-1}q_j}$ é avançada até que v seja encontrado. (b) $\widetilde{p_{i-1}p_i}$ é avançada até que v seja encontrado.

Seja ℓ_q a $\mathcal{G}_\mathcal{F}$-reta que contém $\widetilde{q_{j-1}q_j}$. Se $\widetilde{p_{i-1}p_i}$ e $\widetilde{q_{j-1}q_j}$ são paralelas, então $\widetilde{q_{j-1}q_j}$ será avançada até ir em direção a $\widetilde{p_{i-1}p_i}$, e então caimos no caso anterior. Senão, $\widetilde{p_{i-1}p_i}$ será avançada até cruzar a reta ℓ_q duas vezes ou mudar de direção e não ir mais em direção a ℓ_q. Em ambos os casos, temos $p_i \in H(\widetilde{q_{j-1}q_j})$, e assim, como agora nem $\widetilde{p_{i-1}p_i}$ vai em direção a $\widetilde{q_{j-1}q_j}$ e nem o contrário, p_i é avançada até achar o começo da orelha que contém q_j [figura 3.11(b)]. Neste ponto, caimos no caso inicial, e AVANCE alcança a interseção final desta orelha.

Portanto, uma interseção é encontrada depois de $m + n$ passos. Após estes passos, a fronteira de pelo menos um dos polígonos foi inteiramente visitada. Mais $m + n$ passos são suficientes para encontrarmos a fronteira de $\mathcal{P}_\mathcal{F} \cap \mathcal{Q}_\mathcal{F}$.

Análise de Complexidade

O complexidade do algoritmo proposto é dada pelo laço entre as linhas 3 e 10. No pseudo-código proposto, este laço é percorrido exatamente $2(m + n)$ vezes. Porém, numa implementação mais cuidadosa, é possível terminá-lo assim que a primeira interseção encontrada entre os dois polígonos seja novamente visitada. Desta forma, para determinadas instâncias, é possível encontrar a interseção realizando $m + n$ passagens no laço.

Como o procedimento AVANCE é realizado em tempo constante, o algoritmo realiza $2(m + n)$ iterações no pior caso, tendo complexidade $O(m + n)$.

3.3 Teste de Simplicidade

Nesta seção, analisaremos o seguinte problema:

Problema 3 *Dada uma seqüência de n pontos que definem um polígono no plano, determinar se este polígono é simples, isto é, determinar se algum par dos n segmentos de reta que ligam pontos consecutivos da seqüencia se intercepta em pontos que não sejam os vértices.*

Como é fácil observar, esse problema pode ser reduzido ao problema de interseção de segmentos. Na seção 3.3.1 será discutida uma solução para o caso euclidiano, e, na seção 3.3.2 será proposto um algoritmo para o problema na geometria $\mathcal{G}_\mathcal{F}$.

3.3.1 Geometria Euclidiana

O problema da interseção de segmentos pode ser resolvido trivalmente em tempo $O(n^2)$, fazendo-se a verificação de interseção para cada par de segmentos. Entretanto, esse algoritmo não atinge a otimalidade, já que uma quota inferior do mesmo é $\Omega(n \log n)$ ([PS85]).

Bentley e Ottmann [BO79] propuseram um algoritmo de complexidade $O((n+k)\log n)$ para a solução do problema, onde k é o número de interseções encontradas. É importante observar que o valor de k pode ser da ordem de n^2, o que levaria a termos um algoritmo $O(n^2 \log n)$. Porém, como o polígono tem n vértices e estamos interessados apenas no problema de decisão, isto é, saber se existem dois segmentos que se interceptam, sem ser nos próprios vértices, o algoritmo pode parar ao encontrar a primeira interseção, terminando em tempo ótimo. Abaixo, faremos uma breve descrição do algoritmo original de Bentley e Ottmann, isto é, o algoritmo que encontra todas as interseções. Posteriormente, no algoritmo para a geometria $\mathcal{G}_{\mathcal{F}}$ iremos nos ater à versão de decisão do mesmo.

Algoritmo de Bentley e Ottmann

O algoritmo faz uma varredura planar por uma reta vertical, da esquerda para direita. A propriedade fundamental para a corretude do algoritmo é que, se dois segmentos se interceptam, então em algum momento suas interseções com a reta de varredura serão adjacentes, como mostrado na figura 3.12.

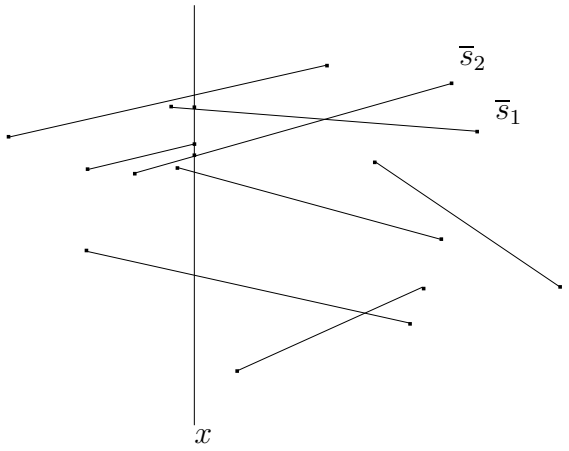


Figura 3.12: Reta de varredura no ponto x. Os segmentos $\overline{s}_1$ e $\overline{s}_2$ se interceptam e suas interseções com a reta de varredura ficam adjacentes.

Para compararmos o posicionamento das interseções e dizermos que estas são adjacentes, é preciso definir a seguinte relação $>_x$. Dizemos que dois segmentos $\overline{s}_1$ e $\overline{s}_2$ são comparáveis em x se a reta vertical na abcissa x intercepta $\overline{s}_1$ e $\overline{s}_2$. Neste caso, dizemos que $\overline{s}_1 >_x \overline{s}_2$ se a interseção de $\overline{s}_1$ com a vertical possui uma ordenada maior que a interseção da vertical com $\overline{s}_2$. Na figura 3.12, $\overline{s}_1 >_x \overline{s}_2$.

É importante observar que, com a movimentação da reta vertical, a ordem dos segmentos com relação a $>_x$ pode mudar em três situações:

- O extremo esquerdo de um novo segmento é encontrado. Neste caso, o segmento tem que ser inserido na ordenação.
- O extremo direito de um segmento é encontrado, portanto, ele não é mais comparável aos outros e deve ser retirado da ordenação.
- A interseção de dois segmentos é encontrada, e sua ordem é invertida.

Assim, esses são os três eventos a serem tratados pela varredura.

A relação de ordem $>_x$ dos segmentos interceptados pela reta de varredura descreve o status da mesma num dado momento. Para mantermos esse status consistente, precisamos que as seguintes operações possam ser realizadas eficientemente (em tempo $O(\log n)$) na estrutura $\mathcal{L}$ que armazena o status da reta.

- INSERE($\overline{s}$, $\mathcal{L}$) Insere o segmento $\overline{s}$ na relação de ordem em $\mathcal{L}$.
- EXCLUI($\overline{s}$, $\mathcal{L}$) Exclui o segmento $\overline{s}$ de $\mathcal{L}$.
- ACIMA($\overline{s}$, $\mathcal{L}$) Retorna o segmento imediatamente superior a $\overline{s}$ na ordem $\mathcal{L}$.
- ABAIXO($\overline{s}$, $\mathcal{L}$) Retorna o segmento imediatamente inferior a $\overline{s}$ em $\mathcal{L}$.

Uma estrutura que admite estas operações é conhecida como dicionário, e as operações acima podem ser realizadas em tempo logarítmico no seu tamanho.

Além do status da reta de varredura, também é necessário manter uma a fila de eventos a serem tratados durante a varredura. Esses eventos são formados pelos extremos dos segmentos, que são inseridos na fila no início do processo, e pelas interseções entre os segmentos, que são detectadas durante a varredura e também alteram o status da reta de varredura.

Assim, precisamos que a fila de eventos seja armazenada em uma estrutura que possibilite as seguintes operações em tempo $O(\log n)$:

- MIN($\mathcal{E}$) Determina o menor elemento em $\mathcal{E}$ e o exclui.
- INSERE(x, $\mathcal{E}$) Insere a abcissa x na ordem mantida por $\mathcal{E}$.
- MEMBRO(x, $\mathcal{E}$) Determina se a abcissa x está em $\mathcal{E}$.

A estrutura $\mathcal{E}$ pode ser uma árvore de busca balanceada, que suporta as três operações em tempo logarítmico no seu tamanho.

O algoritmo começa com a ordenação dos extremos dos segmentos em relação a x e a y e a inserção dos mesmos na fila de eventos. Enquanto a varredura é feita, o status da reta

de varredura é atualizado a cada evento, com a inserção, remoção ou a troca de elementos na estrutura que armazena o status. Além disso, cada vez que o status é atualizado, é feita a verificação se há interseção entre dois segmentos que se tornem adjacentes na reta de varredura. Se uma nova interseção for detectada, esta é reportada e inserida na fila de eventos. Ao término do tratamento de todos os eventos da fila, todas as interseções foram reportadas.

Passemos agora a versão de decisão deste problema sobre a geometria $\mathcal{G}_{\mathcal{F}}$.

3.3.2 Geometria $\mathcal{G}_{\mathcal{F}}$

Ao passarmos para a geometria $\mathcal{G}_{\mathcal{F}}$, temos que fazer mudanças no tratamento da varredura, pois uma reta vertical pode não fazer parte da geometria, possuindo mais de uma interseção com um $\mathcal{G}_{\mathcal{F}}$-segmento, o que faria com que a varredura não funcionasse.

Além disso, para usarmos o mesmo algoritmo, conseguindo a mesma eficiência, temos que ter estruturas de dados capazes de fazer as mesmas operações do caso euclidiano na mesma ordem de tempo.

Primeiramente, seja p_0 um ponto fora da envoltória convexa dos vértices do polígono de entrada. Sejam $B_{\mathcal{F}}(p_0, \rho)$ a $\mathcal{G}_{\mathcal{F}}$-bola de raio $\rho > 0$ centrada em p_0 e q um ponto sobre $B_{\mathcal{F}}(p_0, \rho)$ fora da região de interesse com relação a p_0, conforme mostrado na figura 3.13.

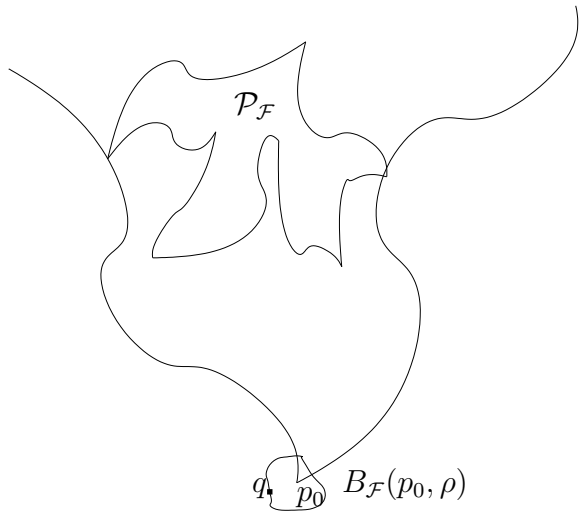


Figura 3.13: Estruturas para varredura circular em torno de p_0.

Caminhando sobre $B_{\mathcal{F}}(p_0, \rho)$ a partir de q no sentido horário podemos fazer uma varredura circular, onde a reta de varredura é o $\mathcal{G}_{\mathcal{F}}$-raio que parte de p_0 e passa pelo ponto que gerou o evento, seja ele um extremo de $\mathcal{G}_{\mathcal{F}}$-segmento ou uma interseção entre

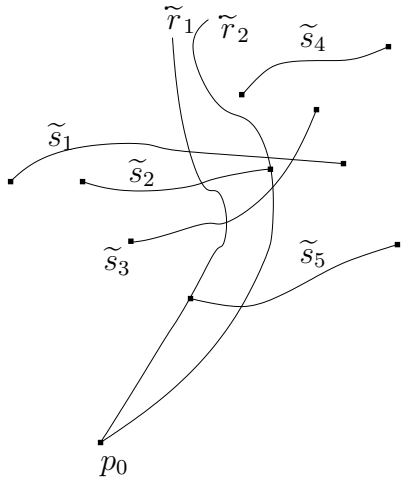


Figura 3.14: Dois $\mathcal{G}_\mathcal{F}$-segmentos que se interceptem, como $\widetilde{s}_1$ e $\widetilde{s}_3$ devem ter suas interseções com o $\mathcal{G}_\mathcal{F}$-raio de varredura adjacentes em algum momento.

dois $\mathcal{G}_\mathcal{F}$-segmentos. Como já provado em 3.2.2, esses $\mathcal{G}_\mathcal{F}$-raios partindo de p_0 e passando pelos pontos mantêm uma relação de ordem circular em torno de p_0, o que garante que os eventos serão tratados na ordem correta.

Agora, precisamos redefinir uma relação de ordem semelhante a $>_x$, de forma que esta seja efetiva na geometria $\mathcal{G}_\mathcal{F}$.

Podemos considerar uma varredura por uma reta vertical, no caso euclidiano, como uma varredura circular em torno de um ponto de coordenada $y = -\infty$. Assim, a comparação entre as ordenadas de dois pontos de mesma abcissa pode ser interpretada como a comparação relativa da distância dos pontos ao ponto em torno do qual estamos fazendo a varredura. Assim, podemos definir a relação $>_\ell$ como:

Dado o ponto p_0 descrito acima e um $\mathcal{G}_\mathcal{F}$-raio $\widetilde{r}$ partindo de p_0 e interceptando dois $\mathcal{G}_\mathcal{F}$-segmentos $\widetilde{s}_1$ e $\widetilde{s}_2$, dizemos que $\widetilde{s}_1 >_\ell \widetilde{s}_2$ se a distância entre $\widetilde{s}_1$ e p_0 for maior que a distância de $\widetilde{s}_2$ a p_0.

Vemos aqui que o operador relacional $>_\ell$, como no caso euclidiano, mantém a ordem total dos segmentos ao longo da varredura em torno de p_0. Os eventos de extremos de segmentos inserem e retiram elementos na ordem criada, mas não mudam a ordem de dois elementos já comparados. A ordem dos segmentos com relação ao raio partindo de p_0 só é alterada nos mesmos três eventos do caso euclidiano.

Vemos na figura 3.14 que, como $\widetilde{s}_1$ e $\widetilde{s}_3$ se interceptam, suas interceções com o $\mathcal{G}_\mathcal{F}$-raio de varredura no momento $\widetilde{r}_2$ são adjacentes. Se isso não ocorresse, deveríamos ter um outro segmento (como $\widetilde{s}_2$) interceptando os dois primeiros no mesmo ponto de $\widetilde{s}_1 \cap \widetilde{s}_3$. Mas, neste caso, num ponto antes de $\widetilde{s}_1 \cap \widetilde{s}_2 \cap \widetilde{s}_3$, $\widetilde{s}_1$ seria adjacente a $\widetilde{s}_2$, e $\widetilde{s}_2$ seria adjacente

a $\widetilde{s}_3$. Portanto, com o operador $>_\ell$, dois segmentos que possuam uma interseção terão suas interseções com $\widetilde{r}$ adjacentes em algum momento, como no caso euclidiano.

Através das mesmas estruturas de dados $\mathcal{L}$ e $\mathcal{E}$ utilizadas no caso euclidiano e do operador $>_\ell$ temos as operações necessárias, na mesma ordem de tempo, para o algoritmo. funcione também na $\mathcal{G}_\mathcal{F}$. Assim, o algoritmo 5 resolve o problema da simplicidade de polígonos no caso da geometria $\mathcal{G}_\mathcal{F}$ em tempo $O(n \log n)$.

3.4 Separabilidade

Nesta seção, serão analisados os problemas de separabilidade de conjuntos e de polígonos.

3.4.1 Separabilidade de Conjuntos

O problema da separabilidade de conjuntos tem grande utilidade em áreas como estatística, reconhecimento de padrões e na área de otimização combinatória, para a resolução de problemas de programação inteira.

Definição 3.4.1 *Dados dois conjuntos de pontos P_1 e P_2, com m e n pontos respectivamente, eles são* separáveis *se existe uma reta l tal que todos os pontos de P_1 estão em um lado de l e todos os pontos de P_2 estão do outro lado.*

Assim, podemos definir o problema da separabilidade como:

Problema 4 *Dados dois conjuntos de pontos, determinar se eles são separáveis.*

Na seção 3.4.1 serão discutidas soluções existentes para o problemas no caso euclidiano e na seção 3.4.1 será proposto um algoritmo na geometria $\mathcal{G}_\mathcal{F}$.

Geometria Euclidiana

O principal critério para a decisão sobre a separabilidade de dois conjuntos de pontos é dado pelo teorema 3.4.1.

Teorema 3.4.1 *[SW70] Dois conjuntos de pontos são linearmente separáveis se, e somente se, seus cascos convexos não se interceptam.*

Por este teorema, vemos que construindo o casco convexo dos conjuntos e calculando suas interseções, podemos decidir sobre a separabilidade dos conjuntos. Como os cascos convexos podem ser construídos em tempo $O((m+n)\log(m+n))$ ([PS85]) e a interseção dos mesmos pode ser calculada em tempo $O(m+n)$, podemos decidir se dois conjuntos são separáveis em tempo $O((m+n)\log(m+n))$.

Algoritmo 5 Algoritmo para verificação da simplicidade de polígonos.

procedimento TESTE-DE-SIMPLICIDADE-DE-POLÍGONO($\mathcal{P}_\mathcal{F}$)
    Ache um ponto p_0 fora da envoltória convexa de $\mathcal{P}_\mathcal{F}$ e a distância de cada vértice de $\mathcal{P}_\mathcal{F}$ a p_0;
    Faça a ordenação circular dos vértices de $\mathcal{P}_\mathcal{F}$ em torno de p_0 e em relação às distâncias dos pontos a p_0 e insira-os numa árvore de busca balanceada $\mathcal{E}$;
    Fila $\mathcal{Q} \leftarrow \emptyset$
    enquanto $\mathcal{E} \neq \emptyset$ **faça**
        $p \leftarrow$ MIN($\mathcal{E}$)
        se p é um extremo esquerdo de uma $\mathcal{G}_\mathcal{F}$-aresta **então**
            $\widetilde{s} \leftarrow$ $\mathcal{G}_\mathcal{F}$-aresta do qual p é extremo esquerdo;
            INSERE($\widetilde{s}$, $\mathcal{L}$);
            $\widetilde{s}_1 \leftarrow$ ACIMA($\widetilde{s}$, $\mathcal{L}$);
            $\widetilde{s}_2 \leftarrow$ ABAIXO($\widetilde{s}$, $\mathcal{L}$);
            se ($\widetilde{s}_1$ intercepta $\widetilde{s}$) **então** $\mathcal{Q} \leftarrow (\widetilde{s}_1, \widetilde{s})$;
            se ($\widetilde{s}_2$ intercepta $\widetilde{s}$) **então** $\mathcal{Q} \leftarrow (\widetilde{s}, \widetilde{s}_2)$;
        senão se p é um extremo direito de uma $\mathcal{G}_\mathcal{F}$-aresta **então**
            $\widetilde{s} \leftarrow$ $\mathcal{G}_\mathcal{F}$-aresta do qual p é extremo direito;
            $\widetilde{s}_1 \leftarrow$ ACIMA($\widetilde{s}$, $\mathcal{L}$);
            $\widetilde{s}_2 \leftarrow$ ABAIXO($\widetilde{s}$, $\mathcal{L}$);
            se $\widetilde{s}_1$ intercepta $\widetilde{s}_2$ à direita de p **então**
                $\mathcal{Q} \leftarrow (\widetilde{s}_1, \widetilde{s}_2)$;
            fim se
            EXCLUI($\widetilde{s}$, $\mathcal{L}$);
        fim se
        enquanto $\mathcal{Q} \neq \emptyset$ **faça**
            $(\widetilde{s}, \widetilde{s}') \leftarrow \mathcal{Q}$
            se $(\widetilde{s}, \widetilde{s}')$ não é um vértice **então**
                retorne "Não é um polígono simples"
            fim se
        fim enquanto
    fim enquanto
    retorne "É um polígono simples"
fim procedimento

Porém, para o caso euclidiano, como mostrado em [PS85], o problema da separabilidade de conjuntos pode ainda ser encarado como um problema de programação linear, e pode ser resolvido em tempo ótimo $\Theta(m+n)$.

Geometria $\mathcal{G}_\mathcal{F}$

Analisando o problema na geometria $\mathcal{G}_\mathcal{F}$ vemos que o teorema 3.4.1 ainda se aplica, como provado abaixo.

Teorema 3.4.2 *Dois conjuntos de pontos P_1 e P_2 são linearmente separáveis numa geometria $\mathcal{G}_\mathcal{F}$ se, e somente se, seus $\mathcal{G}_\mathcal{F}$-cascos convexos não se interceptam.*

Prova: $\Rightarrow$ Se os $\mathcal{G}_\mathcal{F}$-cascos convexos se interceptassem, então pelo menos um ponto p do $\mathcal{G}_\mathcal{F}$-casco convexo de P_1 estaria no $\mathcal{G}_\mathcal{F}$-casco convexo de P_2. Como p pertence ao $\mathcal{G}_\mathcal{F}$-casco convexo de P_2 é impossível separá-lo de P_2 com uma $\mathcal{G}_\mathcal{F}$-reta.

$\Leftarrow$ Suponha que os conjuntos não sejam separáveis, isto é, não existe nenhuma $\mathcal{G}_\mathcal{F}$-reta que deixe os pontos de P_1 e P_2 em lados opostos. Portanto, qualquer $\mathcal{G}_\mathcal{F}$-reta que deixe todos os pontos de P_1 em um de seus lados possui pontos de P_2 neste mesmo lado (caso contrário, ela seria uma separadora). Isso vale também para qualquer $\mathcal{G}_\mathcal{F}$-reta de suporte de P_1. Como dada qualquer $\mathcal{G}_\mathcal{F}$-reta de suporte de P_1 temos sempre um ponto de P_2 no mesmo $\mathcal{G}_\mathcal{F}$-meio-plano que P_1, então temos, obrigatoriamente, um ponto de P_2 interno ao casco convexo de P_1. Logo, os $\mathcal{G}_\mathcal{F}$-cascos convexos se interceptam.

■

Logo, construindo os cascos convexos e determinando sua interseção, podemos decidir se dois conjuntos de pontos são separáveis por uma $\mathcal{G}_\mathcal{F}$-reta. Neste caso, o algoritmo tem a mesma eficiência do algoritmo equivalente para o caso euclidiano, passando a ser afetado apenas por uma constante k dependente dos métodos numéricos para a obtenção de interseções entre $\mathcal{G}_\mathcal{F}$-retas.

Entretanto, este algoritmo não atinge a otimalidade. Permanece aberta a questão de se tratar o problema de forma semelhante à formulação do caso euclidiano como um problema de programação linear.

3.4.2 Separabilidade de Polígonos

Como afirmado no teorema 3.4.1, dois conjuntos são separáveis se, e somente se, seus cascos convexos são disjuntos. Esse teorema é utilizado também para a decisão da separabilidade de polígonos. O problema da separabilidade de polígonos é definido como:

Problema 5 *Dados dois polígonos simples $\mathcal{P}$ e $\mathcal{Q}$ com n e m vértices respectivamente, determinar se eles são separáveis (definição 3.4.1).*

Na seção 3.4.2 o problema será discutido na geometria euclidiana. A seção 3.4.2 apresenta um algoritmo para o problema na geometria $\mathcal{G}_{\mathcal{F}}$.

Geometria Euclidiana

Através da construção do casco convexo de dois conjuntos de pontos e a verificação da interseção dos mesmos, temos um algoritmo $O((m+n)\log(m+n))$ para a resolução da separabilidade de conjuntos. O fator logarítmico na complexidade do algoritmo surge no passo de construção do casco convexo, o que não pode ser feito em tempo inferior a $\Omega(n \log n)$.

Porém, se os pontos dados formam um polígono simples, podemos encontrar algoritmos melhores. Para a construção do casco convexo de polígonos simples, existem algoritmos de complexidade $O(n)$, o que nos leva diretamente a um algoritmo $O(m+n)$ para o problema da separabilidade de polígonos simples.

Temos que lembrar também que, no caso euclidiano, o problema da separabilidade de polígonos também pode ser resolvido em tempo $O(m+n)$ por programação linear.

Geometria $\mathcal{G}_{\mathcal{F}}$

O problema de separabilidade de conjuntos pode ser resolvido em tempo $O((m+n)\log(m+n))$ na geometria $\mathcal{G}_{\mathcal{F}}$ (seção 3.4.1).

No caso da entrada ser formada por polígonos simples, podemos fazer uso desse conhecimento para chegarmos a um algoritmo mais eficiente. Será mostrado na seção 3.6.2 que o casco convexo de um $\mathcal{G}_{\mathcal{F}}$-polígono simples pode ser construído em tempo $O(n)$. Além disso, como mostrado na seção 3.2.2 a interseção entre dois $\mathcal{G}_{\mathcal{F}}$-polígonos convexos pode ser feita em tempo $O(m+n)$. Portanto, na geometria $\mathcal{G}_{\mathcal{F}}$ também podemos resolver a separabilidade de polígonos simples em tempo $O(m+n)$.

3.5 Casco Convexo Incremental

A envoltória $\mathcal{G}_{\mathcal{F}}$-convexa ou casco $\mathcal{G}_{\mathcal{F}}$-convexo de um conjunto de pontos foi definida por Harada:

Definição 3.5.1 *(Harada [Har00]) A envoltória $\mathcal{G}_{\mathcal{F}}$-convexa ou casco $\mathcal{G}_{\mathcal{F}}$-convexo de um conjunto de pontos em $\mathbb{R}^2$ é o menor conjunto $\mathcal{G}_{\mathcal{F}}$-convexo que o contém. Onde "menor" é determinado pela relação de ordem parcial induzida pela relação "$\subset$".*

Com base nessa definição, Harada propôs dois algoritmos para a construção do casco $\mathcal{G}_{\mathcal{F}}$-convexo de um conjunto de n pontos [Har00]. Ambos visam obter uma boa eficiência na construção do casco $\mathcal{G}_{\mathcal{F}}$-convexo, sendo um com complexidade $O(n \log n)$ e outro $O(n \log h)$, onde h é o número de pontos no casco convexo construído. Porém, essa ênfase na busca por uma melhor eficiêcia faz com que estes algoritmos tenham detalhes que dificultam a implementação dos mesmos. Assim, descrevemos aqui um algoritmo incremental para a construção do casco $\mathcal{G}_{\mathcal{F}}$-convexo de um conjunto de pontos, o qual implementamos no visualizador da geometria $\mathcal{G}_{\mathcal{F}}$ (veja capítulo 4). Primeiro, será descrito o funcionamento do algoritmo na geometria euclidiana e, depois, será apresentada uma versão para a geometria $\mathcal{G}_{\mathcal{F}}$.

3.5.1 Geometria Euclidiana

Para facilitar a descrição do algoritmo, vamos supor que nenhum trio dos n pontos de entrada sejam colineares. O algoritmo incremental para a construção do casco convexo tem como base a construção do casco convexo de três pontos, que é o triângulo formado por eles. A partir deste, incluímos os demais pontos, um a um, sempre montando o casco convexo do novo conjunto. O pseudo-código apresentado no algoritmo 6 é adaptado de [dRS94]. Nele, $\Delta(p_1, p_2, p_3)$ representa a orientação dos pontos p_1, p_2 e p_3.

Algoritmo 6 Algoritmo incremental para construção do casco convexo de n pontos.

```
1:  procedimento CASCO-CONVEXO-INCREMENTAL(conjunto P de n pontos no plano)
2:      se |P| = 3 então
3:          se Δ(p1, p2, p3) > 0 então
4:              Retorne (p1, p2, p3)
5:          senão
6:              Retorne (p2, p1, p3)
7:          fim se
8:      fim se
9:      Escolha um ponto p ∈ P e faça P' ← P − {p}
10:     Conv(P') = CASCO-CONVEXO-INCREMENTAL(P')
11:     se p ∈ Conv(P') então
12:         Retorne Conv(P')
13:     senão
14:         Forme o Conv(P) estendendo Conv(P') de forma que p seja um de seus vértices
15:         Retorne Conv(P)
16:     fim se
17: fim procedimento
```

O passo da linha 14 é feito através da construção de duas pontes, que são as duas retas de suporte do casco convexo de $Conv(P')$ passando por p. Encontradas as pontes e

os vértices p_l e p_h de $Conv(P')$ que as formam, a cadeia de arestas $(p_l, p_{l+1}, \ldots, p_{h-1}, p_h)$ deve ser eliminada e substituída por (p_l, p, p_h) (figura 3.15).

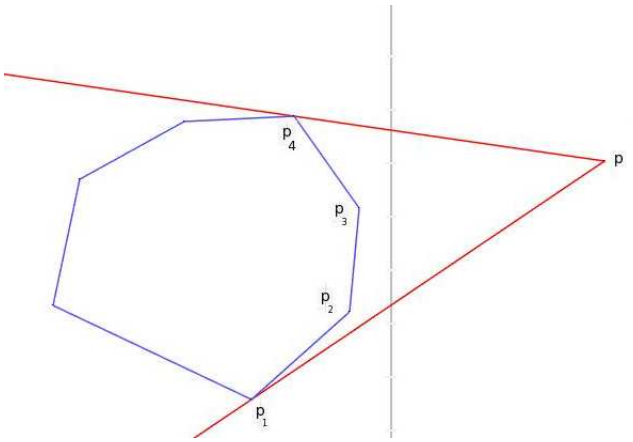


Figura 3.15: Para inserir o ponto p no $Conv(P')$, p_1 é o ponto p_l e p_4 é o ponto p_h. A cadeia (p_1, p_2, p_3, p_4) é substituída por (p_1, p, p_4).

Para acharmos p_h fazemos uma busca pelos n pontos de forma a encontrar os dois pontos onde os trios (p, p_{i-1}, p_i) e (p, p_i, p_{i+1}) têm orientações opostas. Esses são os dois pontos que pertencem às retas de suporte de $Conv(P')$ passando por p. Agora, se observarmos as orientações de (p, p_i, p_{i+1}) para esses dois pontos, podemos ver que, se essa orientação for positiva, então p_i é p_h, caso contrário, p_i é p_l. Como mostrado em [dRS94], esse algoritmo tem complexidade $O(n^2)$.

3.5.2 Geometria $\mathcal{G}_\mathcal{F}$

Observando o algoritmo anterior, vemos que todos os passos do mesmo podem ser feitos através da verificação da orientação de pontos, que pode ser resolvida também na geometria $\mathcal{G}_\mathcal{F}$, como mostrado na seção 2.4. Além disso, temos a utilização do caso base de três pontos que na geometria $\mathcal{G}_\mathcal{F}$ é o $\mathcal{G}_\mathcal{F}$-triângulo formado por eles.

Vemos que quando o ponto p escolhido na linha 9 é interno a $Conv(P')$, o algoritmo retorna o próprio $Conv(P')$, o que é correto também na $\mathcal{G}_\mathcal{F}$. Já para o caso em que $p \notin Conv(P')$, p é incluído no $\mathcal{G}_\mathcal{F}$-casco convexo anterior e, com isso, todos os pontos internos ao novo $\mathcal{G}_\mathcal{F}$-casco convexo são eliminados. Isto pode ser verificado pois os pontos $(p_{l+1}, \ldots, p_{h-1})$ estão sempre à esquerda da $\mathcal{G}_\mathcal{F}$-reta que passa por p e p_h com essa orientação, e à direita da $\mathcal{G}_\mathcal{F}$-reta que passa por p e p_l, com essa orientação.

Por fim, vemos que o passo da linha 14 exige apenas um percurso na fronteira de $Conv(P')$, tomando tempo $O(n)$. Portanto, como no caso euclidiano, o algoritmo continua

tendo complexidade $O(n^2)$. A figura 3.16 ilustra um caso similar ao mostrado na figura 3.15, porém, numa geometria $\mathcal{G}_\mathcal{F}$ ao invés da euclidiana.

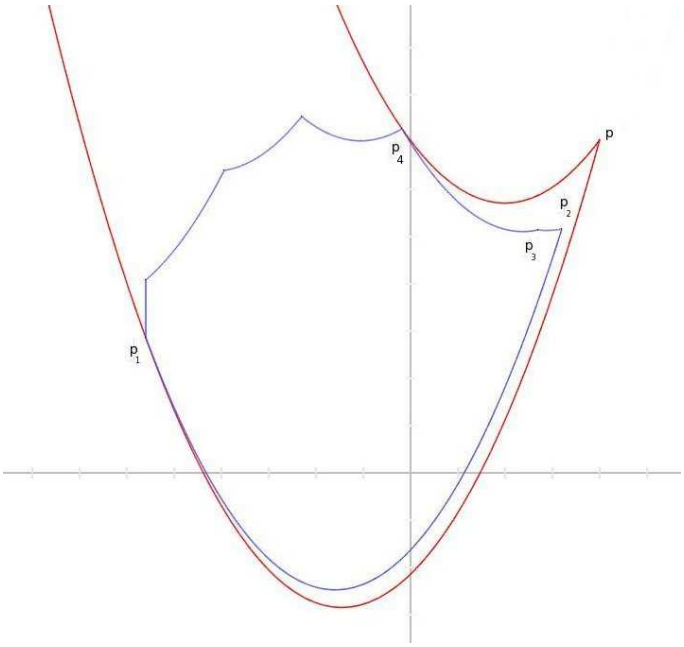


Figura 3.16: Caso análogo ao mostrado na figura 3.15, mas para uma geometria $\mathcal{G}_\mathcal{F}$. Para inserir o ponto p no $Conv(P')$, p_1 é o ponto p_l e p_4 é o ponto p_h. A cadeia (p_1, p_2, p_3, p_4) é substituída por (p_1, p, p_4).

3.6 Casco Convexo de um Polígono Simples

Sempre que uma restrição é feita na formulação de um problema, vale a pena questionar se isso possibilita uma melhora nos algoritmos existentes para aquele problema.

A construção do casco convexo de um polígono simples é um dos problemas em que se vê claramente uma restrição na entrada. Dado um conjunto de pontos, sempre é possível interpretá-lo como um polígono, tomando uma seqüência qualquer dos pontos e adicionando arestas entre dois pontos consecutivos. Entretanto, isso não garante que o polígono gerado seja simples, podendo ter interseção entre as arestas. O fato de sabermos que o polígono de entrada é simples possibilitou, no caso euclidiano, o desenvolvimento de algoritmos cuja complexidade é menor que a quota inferior de $\Omega(n \log n)$ conhecida para o problema sem restrições.

A seguir, apresentaremos um algoritmo $O(n)$ para a construção do casco convexo de

um polígono simples $\mathcal{P}$ no caso euclidiano e, posteriormente, será proposto um algoritmo de mesma complexidade para um $\mathcal{G}_\mathcal{F}$-polígono simples $\mathcal{P}_\mathcal{F}$ na geometria $\mathcal{G}_\mathcal{F}$.

3.6.1 Geometria Euclidiana

O algoritmo apresentado a seguir é descrito por Preparata e Shamos [PS85], sendo uma variação do algoritmo proposto por D. T. Lee [Lee83].

Seja v_1 o vértice mais à esquerda de um polígono simples $\mathcal{P}$ e seja $(v_1, v_2, \ldots, v_n)$ os seus vértices ordenados (com v_1 seguindo v_n). Assuma que seus vértices são dados no sentido horário (figura 3.17) e seja o v_m o vértice mais a direita. Claramente v_1 e v_m fazem parte do casco convexo de $\mathcal{P}$. Além disso, esses vértices dividem a cadeia total em duas partes, a superior, que vai de v_1 a v_m e outra, inferior, que vai de v_m a v_1. Vamos explicitar aqui como construir o casco da cadeia que vai de v_1 a v_m, também chamado de casco superior. O casco inferior, pertencente à cadeia que vai de v_m a v_1 pode ser construído de maneira análoga. Queremos, então, encontrar a subseqüência $(q_1, q_2, \ldots, q_r)$, com $v_1 = q_1$ e $q_r = v_m$, de $(v_1, v_2, \ldots, v_m)$ que forme o casco superior. Cada aresta $\overline{q_i q_{i+1}}$ $(i = 1, \ldots, r-1)$ pode ser vista como uma "tampa" do "bolso" K_i, onde o bolso K_i é uma subcadeia de $(v_1, v_2, \ldots, v_m)$, cujo primeiro vértice é q_i e o último é q_{i+1}.

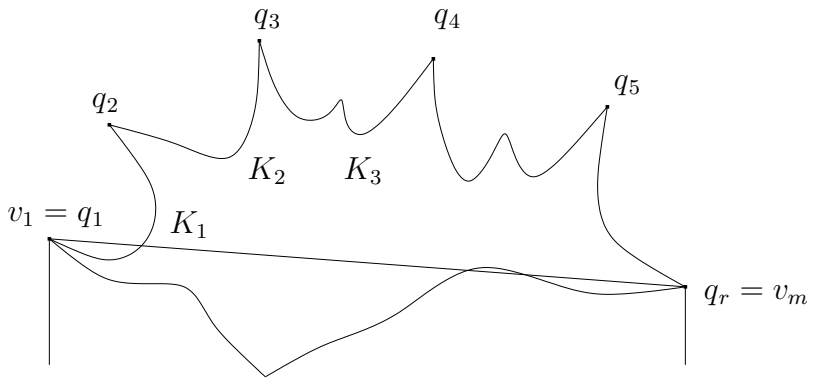


Figura 3.17: Construção do casco convexo superior de um polígono simples.

O algoritmo faz um percurso por $(v_1, \ldots, v_m)$ achando a tampa de cada bolso. O principal evento na execução do algoritmo é a identificação de um vértice q_i que forma uma tampa com o último vértice q_{i-1} encontrado (de agora em diante, q_i será chamado de *vértice de evento*). Convém destacar que cada vértice de evento não pertence necessariamente ao casco convexo, mas é um candidato a tal. Vamos observar agora, como

o algoritmo passa de um vértice de evento para o próximo, o que define o passo para avançar a construção do casco convexo.

Suponha que a fronteira já tenha sido visitada de v_1 a v_s ($s < m$) e que $v_s = q_i$ é um vértice de evento. Essa situação é mostrada na figura 3.18, onde o bolso K_{i-1} foi fechado pela tampa $\overline{q_{i-1}q_i}$. A caminhada continua pelos vértices posteriores a v_s. Chamamos de u o vértice que precede q_i na fronteira de $\mathcal{P}$. Podemos identificar duas situações de acordo com o posicionamento de u com relação ao segmento orientado $\overrightarrow{q_r q_i}$.

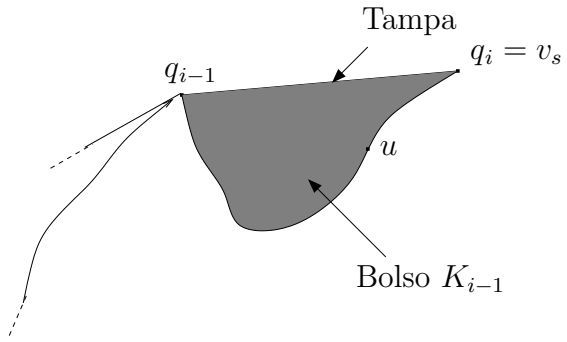


Figura 3.18: Um bolso e sua tampa.

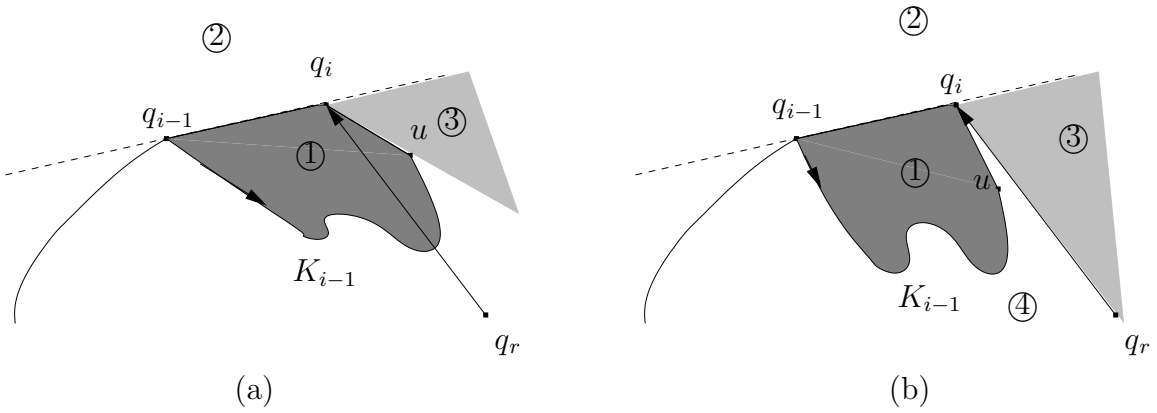


Figura 3.19: Regiões onde o vértice v, que segue o vértice q_i pode estar (dependendo da posição de u com relação a $\overrightarrow{q_r q_i}$).

1. *u está à direita de $\overrightarrow{q_r q_i}$.* Neste caso, podemos identificar três regiões bem definidas, rotuladas ①, ② e ③ na figura 3.19 (a). As regiões são formadas pela reta passando por q_{i-1} e q_i, o raio obtido pela extensão do segmento $\overline{q_i u}$ na direção de u e o pedaço da fronteira de $\mathcal{P}$ correspondente ao bolso K_{i-1}.

2. *u está à esquerda de* $\overline{q_r q_i}$. Neste caso, temos uma região a mais, rotulada ④ na figura 3.19 (b).

Sendo v o vértice seguinte a q_i na fronteira de $\mathcal{P}$, vemos que v deve estar em uma das regiões identificadas anteriormente. Se v estiver localizado em ② ou ③ ele é um vértice de evento, o que não ocorre se ele estiver em ① ou ④. Veremos cada um dos casos detalhadamente:

1. $v \in$ *região* ① Neste caso, a fronteira de $\mathcal{P}$ entra no bolso K_{i-1}. Então, deve-se percorrer os vértices até encontrarmos o primeiro vértice w que esteja fora do bolso, isto é, na região ② [figura 3.20 (a)]. Como $\mathcal{P}$ é um polígono simples, para que não haja interseção entre as arestas de $\mathcal{P}$ e para que $\mathcal{P}$ possa ser fechado, um caminho de arestas que entra pela tampa do bolso K_{i-1} deve sair pela mesma, logo w existe e deve ser tratado como pertencente à região ② (item 2).

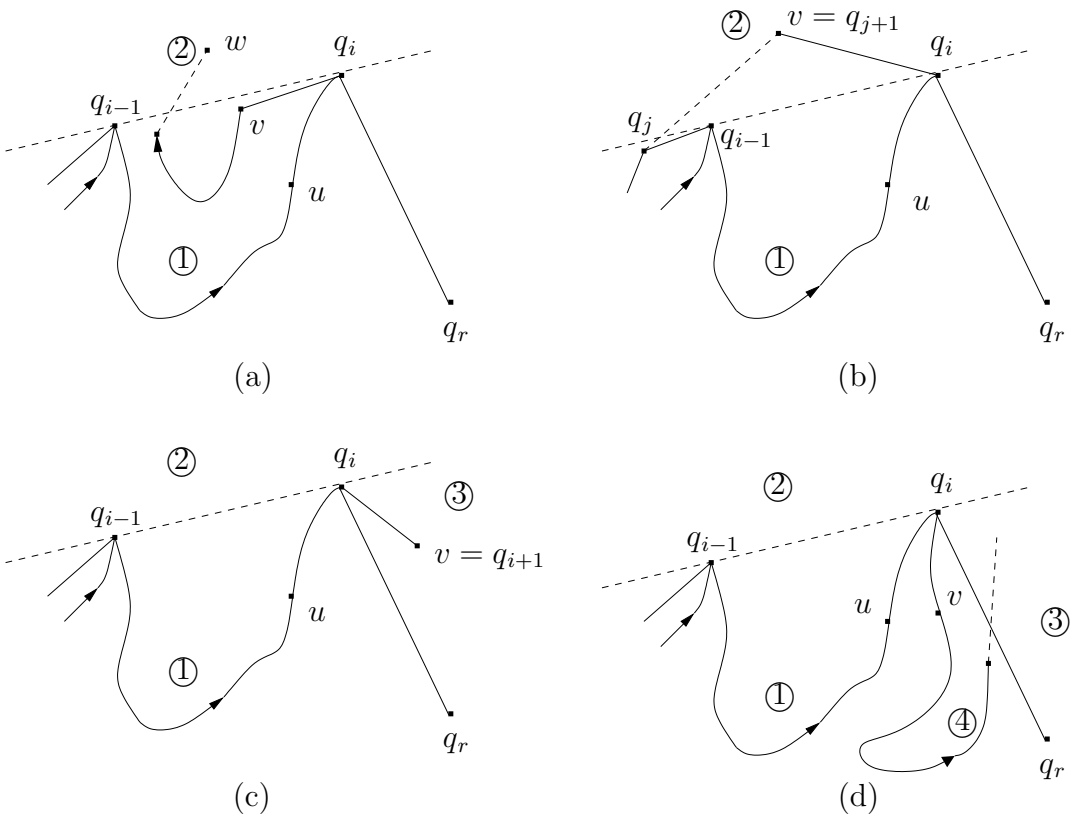


Figura 3.20: Quatro situações possíveis quando se percorre a fronteira de $\mathcal{P}$ após o evento q_i.

2. $v \in$ *região* ② O vértice v é um vértice de evento. A reta de suporte de v para a cadeia $(q_1, q_2, \ldots, q_{i-1})$ deve ser encontrada observando os vértices a partir de q_{i-1} até q_1. Supondo que q_j seja o vértice que define a reta de suporte procurada, com $j < i$, então os vértices $q_{j+1}, \ldots, q_i$ devem ser descartados e v se torna q_{j+1} [figura 3.20 (b)]. Podemos afirmar que o vértice v faz parte do casco convexo construído até o momento (sendo assim um ponto de evento), já que é externo ao casco convexo construído pelo último vértice de evento encontrado $(q_1, \ldots, q_r)$.

3. $v \in$ *região* ③ O vértice v é um vértice de evento e se torna q_{i+1}. De fato, v tem que ser externo ao casco convexo atual $(q_1, \ldots, q_i, q_r)$ [figura 3.20 (c)]. Além disso, pela construção da região ③, v tem que estar à direita da reta que vai de q_{i-1} a q_i.

4. $v \in$ *região* ④ A fronteira de $\mathcal{P}$ entra no casco convexo do polígono. Como no caso 1, temos que seguir o caminho dos vértices até encontrar um que seja externo à região ④ ou coincida com q_r. No segundo caso, o procedimento é encerrado. No primeiro caso, o caminho dos vértices deve cruzar $\overline{q_i q_r}$ obrigatoriamente, já que q_r é o vértice mais à direita de $\mathcal{P}$. O vértice que estamos procurando pode estar na região ③ (e tratado como no item 3) ou na ② (e tratado como no item 2) [figura 3.20 (d)].

O algoritmo 7 implementa os casos mostrados acima. Nele, $\mathcal{Q}$ é uma fila contendo os vértices $(v_1, \ldots, v_m)$ e $\mathcal{S}$ é uma pilha para a seqüência $(q_0, q_1, q_2, \ldots)$, onde q_0 é um vértice sentinela com mesma abcissa que $v_1 = q_1$ e menor ordenada. A escolha do ponto q_0 é feita de forma que sua inserção no casco convexo superior de $\mathcal{P}$ não altere o mesmo. De fato, vemos que o casco convexo superior de $\mathcal{P}$ é formado por um segmento de reta que vai do ponto $y = -\infty$ até q_1, seguido dos segmentos $(q_1, q_2, \ldots, q_r)$, e, por fim, do segmento que vai de q_r ao ponto $y = -\infty$. Portanto, qualquer ponto tomado no segmento que vai de $y = -\infty$ até q_1 não altera o casco convexo superior. Como mencionado anteriormente, u é o vértice que precede q_i na fronteira de $\mathcal{P}$ e v é vértice que está sendo visitado. Dada a tripla ordenada de vértices (v, u, w), dizemos que ela tem *orientação negativa* se w está à direita da reta direcionada que vai de v a u, caso contrário, dizemos que tem *orientação positiva*. Além disso, $\mathcal{S} \leftarrow v$ denota a operação de inserção de v em $\mathcal{S}$, $v \leftarrow$ desenfilera($\mathcal{Q}$) significa que v recebe o primeiro elemento de $\mathcal{Q}$ e este é removido da mesma, $v \leftarrow$ desempilha($\mathcal{S}$) significa v recebe o elemento que está no topo da pilha $\mathcal{S}$ e este é desempilhado, e topo($\mathcal{S}$) é último elemento que foi inserido em $\mathcal{S}$.

Como mostrado por Preparata e Shamos [PS85], esse algoritmo constrói o casco convexo de um polígono simples em tempo ótimo $O(n)$.

Algoritmo 7 Algoritmo para construção do casco convexo de um polígono simples.

```
1: procedimento CASCO-CONVEXO-SUPERIOR-DE-POLÍGONO-SIMPLES(P)
2:     Q ← (v_2, ..., v_m);
3:     S ← q_0;
4:     S ← v_1;
5:     q_1 ← topo(S)
6:     q_r ← v_m
7:     enquanto Q ≠ ∅ faça
8:         v ← desenfilera(Q)
9:         se (q_{i-1} q_i v) tem orientação negativa /* regiões ①, ③, ④ */ então
10:            se (u q_i v) tem orientação negativa /* regiões ③, ④ */ então
11:                se (q_r q_i v) tem orientação negativa /* região ③ */ então
12:                    S ← v;
13:                senão
14:                    /* região ④ */
15:                    q ← desenfilera(Q)
16:                    enquanto (q está a esquerda de q_r q_i) faça
17:                        q ← desenfilera(Q)
18:                    fim enquanto
19:                fim se
20:            senão
21:                /* região ① */
22:                q ← desenfilera(Q)
23:                enquanto (q está a esquerda de q_i q_{i-1}) faça
24:                    q ← desenfilera(Q)
25:                fim enquanto
26:            fim se
27:        senão
28:            /* região ② */
29:            enquanto ((q_{i-1} q_i v) tem orientação positiva) faça
30:                q_i ← desempilha(S)
31:            fim enquanto
32:            S ← v;
33:        fim se
34:    fim enquanto
35: fim procedimento
```

3.6.2 Geometria $\mathcal{G}_\mathcal{F}$

Um $\mathcal{G}_\mathcal{F}$-polígono simples na geometria $\mathcal{G}_\mathcal{F}$, que é formado por $\mathcal{G}_\mathcal{F}$-segmentos de curvas que não se interceptam exceto nos vértices, é também uma curva de Jordan. Com, e isso, análogo ao caso euclidiano, podemos procurar dois pontos extremos do $\mathcal{G}_\mathcal{F}$-polígono $\mathcal{P}_\mathcal{F}$ de entrada e construir os $\mathcal{G}_\mathcal{F}$-cascos convexos superior e inferior de $\mathcal{P}_\mathcal{F}$ encontrando as tampas para os bolsos do $\mathcal{G}_\mathcal{F}$-polígono. Veremos cada passo mais detalhadamente.

Primeiramente, podemos encontrar um ponto p_0 fora de $\mathcal{P}_\mathcal{F}$, e os pontos p_i e p_f de $\mathcal{P}_\mathcal{F}$ da mesma forma que foi feito na seção 3.2.2, em tempo $O(n)$. Esses dois pontos dividem o polígono de entrada em duas cadeias de vértices, uma que vai de p_i a p_f, e outra de p_f a p_i.

Teorema 3.6.1 *Os pontos p_i e p_f fazem parte do $\mathcal{G}_\mathcal{F}$-casco convexo de $\mathcal{P}_\mathcal{F}$.*

Prova: Será apresentada a prova de que p_i pertence ao $\mathcal{G}_\mathcal{F}$-casco convexo de $\mathcal{P}_\mathcal{F}$, sendo análoga a prova para p_f. Por construção, p_i é o vértice mais à esquerda de $\mathcal{P}_\mathcal{F}$ com relação a p_0 (onde o termo esquerda é utilizado como mostrado na definição 2.3.4). Por absurdo, suponha que p_i não faz parte do $\mathcal{G}_\mathcal{F}$-casco convexo de $\mathcal{P}_\mathcal{F}$. Assim, deve existir uma $\mathcal{G}_\mathcal{F}$-aresta do $\mathcal{G}_\mathcal{F}$-casco convexo que passa pelo $\mathcal{G}_\mathcal{F}$-meio-plano à esquerda da $\mathcal{G}_\mathcal{F}$-reta ℓ que passa por p_0 e p_i e é orientada neste sentido, deixando p_i no interior do $\mathcal{G}_\mathcal{F}$-casco. Como não existe vértice à esquerda de p_i com relação a p_0, se existir tal aresta, ela terá duas interseções com ℓ, o que é um absurdo pela construção da geometria. Portanto, p_i está no casco convexo de $\mathcal{P}_\mathcal{F}$. ■

Assim, os pontos p_i e p_f fazem o papel dos pontos v_1 e v_m, respectivamente, do caso euclidiano.

Observando os possíveis casos de posicionamento dos vértices de evento no caso euclidiano, vemos que o tratamento de cada caso faz uso apenas da primitiva de orientação de pontos, decidindo de que lado um vértice está com relação a uma $\mathcal{G}_\mathcal{F}$-reta orientada formada por outros dois pontos. Como mostrado na seção 2.4, a orientação de pontos também pode ser verificada na geometria $\mathcal{G}_\mathcal{F}$. Além disso, a escolha dos casos onde o ponto de evento pode estar posicionado faz uso do fato do polígono de entrada ser uma curva de Jordan, o que também é verdadeiro para um $\mathcal{G}_\mathcal{F}$-polígono. Portanto, o algoritmo 7 continua válido para um $\mathcal{G}_\mathcal{F}$-polígono na entrada, exceto pelo ponto q_0 utilizado como sentinela no início do algoritmo.

No caso da geometria $\mathcal{G}_\mathcal{F}$, assim como no euclidiano, q_0 deve ser um ponto que preceda v_1 no $\mathcal{G}_\mathcal{F}$-casco convexo superior e, quando inserido na sequência de vértices do $\mathcal{G}_\mathcal{F}$-casco convexo, não altere sua forma. O $\mathcal{G}_\mathcal{F}$-casco convexo superior procurado é formado pela seqüência de pontos $(p_0, q_1, q_2, \ldots, q_r)$. Assim, qualquer ponto no $\mathcal{G}_\mathcal{F}$-segmento $\widetilde{p_0 v_1}$ pode ser inserido na seqüência de vértices do $\mathcal{G}_\mathcal{F}$-casco convexo sem alterá-lo, servindo de

sentinela. Portanto, o ponto q_0 pode ser qualquer ponto sobre o segmento $\widetilde{p_0v_1}$, exceto os extremos.

Como o processamento da fila $\mathcal{Q}$ é feito de forma que cada vértice seja aceito (linhas 12 e 32) ou descartado (linha 17 ou 24), ele toma tempo $O(m)$. Além disso, cada vértice excluído de $\mathcal{S}$ no loop da linha 29 não volta a ser empilhado. Portanto, o algoritmo é executado em tempo linear.

Assim, na geometria $\mathcal{G}_\mathcal{F}$ também podemos construir o casco convexo de um polígono simples em tempo $O(n)$.

3.7 Núcleo de Polígono Simples

Como visto anteriormente, saber se um polígono é estrelado e conhecer seu núcleo (definição 3.1.1) pode ser um pré-processamento vantajoso para que possamos responder sobre a inclusão de pontos nesse polígono.

Neste capítulo será tratado o problema da construção do núcleo de um polígono simples.

Problema 6 *Dado um polígono simples $\mathcal{P}$ com n vértices, determinar se $\mathcal{P}$ é estrelado e, se o for, identificar seu núcleo.*

O polígono de entrada $\mathcal{P}$ será descrito pela seqüência de vértices $v_0, v_1, \ldots, v_{n-1}$ de sua fronteira dados no sentido anti-horário, com $n \geq 4$. Com isso, é fácil ver que o núcleo de $\mathcal{P}$ é formado pela interseção de todos os meios-planos à esquerda de todas arestas de $\mathcal{P}$. Podemos representar $\mathcal{P}$ por uma lista circular de vértices e arestas como $v_0\overline{e}_1v_1\overline{e}_2 \ldots \overline{e}_{n-1}v_{n-1}\overline{e}_0v_0$, onde $\overline{e}_i = \overline{v_{i-1}v_i}$. A mesma representação também será utilizada para o problema na geometria $\mathcal{G}_\mathcal{F}$, onde teremos um $\mathcal{G}_\mathcal{F}$-polígono $\mathcal{P}_\mathcal{F}$ na entrada e as $\mathcal{G}_\mathcal{F}$-arestas serão representadas como $\widetilde{e}_i = \widetilde{v_{i-1}v_i}$.

Além disso, um vértice v_i é chamado *reflexo* se v_{i+1} está à direita da reta que contém a aresta $\overline{e}_i$ e direcionada como $\overline{e}_i$. Um vértice é chamado *convexo* no caso contrário. Esta mesma nomenclatura será utilizada para a geometria $\mathcal{G}_\mathcal{F}$. Se o núcleo de $\mathcal{P}$, denotado $K(\mathcal{P})$, não for vazio, será representado na saída como a seqüência de vértices e arestas do polígono $K(\mathcal{P})$.

Primeiro será apresentado um algoritmo proposto por Lee e Preparata [LP79] para o caso euclidiano e, posteriormente, será mostrado um algoritmo para a geometria $\mathcal{G}_\mathcal{F}$.

3.7.1 Geometria Euclidiana

O algoritmo percorre os vértices de $\mathcal{P}$ e constrói uma seqüência de polígonos convexos $\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_{n-1}$, que podem ser limitados ou ilimitados. Cada polígono $\mathcal{K}_i$ é a interseção

dos meios-planos à esquerda das arestas direcionadas $\overline{e}_0$, $\overline{e}_1$, ..., $\overline{e}_i$. Isso implica que $\mathcal{K}_{n-1} = K(\mathcal{P})$ e que $\mathcal{K}_1 \supseteq \mathcal{K}_2 \supseteq \cdots \supseteq \mathcal{K}_i$. Esta última implicação faz com que exista um $r > 1$ tal que $\mathcal{K}_i$ seja limitado ou ilimitado se $i < r$ ou $i \geq r$, respectivamente.

Para facilitar a notação, se p_i e p_{i+1} são pontos pertencentes à reta que contém a aresta $\overline{e}$ de $\mathcal{P}$, então $p_i\overline{e}p_{i+1}$ denota o segmento entre p_i e p_{i+1} com a mesma direção de $\overline{e}$. Quando o polígono $\mathcal{K}_i$ for ilimitado, duas de suas arestas são raios, e $\Lambda\overline{e}p$ denota o raio terminando no ponto p e direcionado como $\overline{e}$, enquanto $p\overline{e}\Lambda$ denota o raio que parte de p e vai para o infinito orientado como $\overline{e}$.

Durante a execução do algoritmo, a fronteira de $\mathcal{K}_i$ é mantida como uma lista duplamente ligada de vértices e arestas intercaladas. Esta lista será linear ou circular, dependendo se $\mathcal{K}_i$ é ilimitado ou limitado, respectivamente. No primeiro caso, o primeiro e o último elementos da lista serão chamados de *cabeça* e *cauda* respectivamente.

Entre os vértices de $\mathcal{K}_i$ distinguiremos dois deles, $q_i{}'$ e $q_i{}''$, definidos a seguir. Considere as duas retas de suporte de $\mathcal{K}_i$ passando pelo vértice v_i de $\mathcal{P}$. Sejam $\overline{r}'_i$ e $\overline{r}''_i$ os dois raios sobre essas linhas, partindo de v_i e contendo os pontos de suporte, de forma que o raio $\overline{r}'_i$ deixa $\mathcal{K}_i$ no meio-plano à sua direita e $\overline{r}''_i$ deixa $\mathcal{K}_i$ no meio-plano à sua esquerda (figura 3.21). O vértice $q_i{}'$ é o ponto comum a $\overline{r}'_i$ e $\mathcal{K}_i$ mais afastado de v_i; $q_i{}''$ é definido de maneira análoga. Estes dois vértices são essenciais para a construção de $\mathcal{K}_{i+1}$ a partir de $\mathcal{K}_i$.

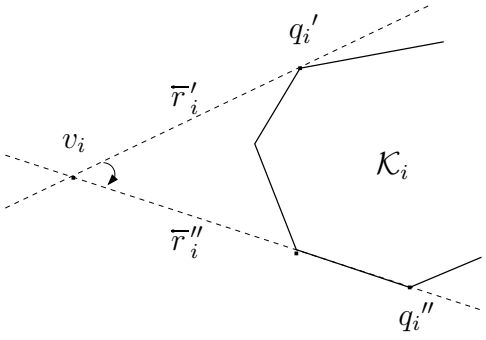


Figura 3.21: Ilustração da definição de $q_i{}'$ e $q_i{}''$.

Se $\mathcal{P}$ não tem ângulos reflexos, então $\mathcal{P}$ é convexo e trivialmente $K(\mathcal{P}) = \mathcal{P}$. Considere os vértices de $\mathcal{P}$ renomeados de forma que v_0 seja um vértice reflexo, e os demais sejam nomeados $v_1, v_2, \ldots, v_{n-1}$ a partir dele. Passemos à descrição do algoritmo.

Passo inicial $\mathcal{K}_1$ é a interseção dos meios-planos à esquerda das arestas $\overline{e}_0$ e $\overline{e}_1$, isto é, $\mathcal{K}_1 \leftarrow \Lambda\overline{e}_1 v_0 \overline{e}_0 \Lambda$ (figura 3.22). $q_i{}' \leftarrow$ ponto no infinito de $\Lambda\overline{e}_1 v_0$; $q_i{}'' \leftarrow$ ponto no infinito de $v_0\overline{e}_0\Lambda$.

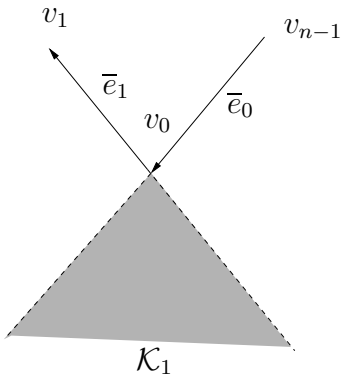


Figura 3.22: Polígono $\mathcal{K}_1$.

Passo geral Aqui precisamos separar alguns casos. Sejam v_i o vértice de $\mathcal{P}$ que está sendo visitado, e $w_1, w_2, \ldots$ os vértices de $\mathcal{K}_i$, dados no sentido anti-horário.

1. *Vértice v_i é reflexo.* [figura 3.23(a, b)]

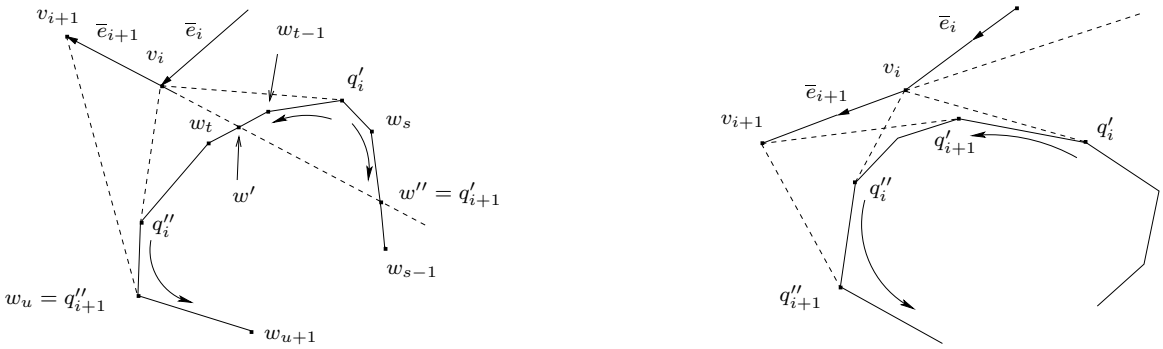


Figura 3.23: Passo geral quando v_i é reflexo.

(a) *q'_i está sobre ou à direita de $\Lambda\overline{e}_{i+1}v_{i+1}$.* [figura 3.23(a)] - Percorremos a fronteira de $\mathcal{K}_i$ no sentido anti-horário a partir q'_i até encontrarmos a aresta única $\overline{w_{t-1}w_t}$ de $\mathcal{K}_i$ que intercepta $\Lambda\overline{e}_{i+1}v_{i+1}$, ou alcançamos q''_i sem encontrar tal aresta. No segundo caso, o algoritmo termina ($K(\mathcal{P}) = \emptyset$). No primeiro caso, são realizadas as seguintes ações:

i. Encontramos a interseção w' de $\overline{w_{t-1}w_t}$ e $\Lambda\overline{e}_{i+1}v_{i+1}$.

ii. Percorremos a fronteira de $\mathcal{K}_i$ no sentido horário a partir de q_i' até atingirmos uma aresta $\overline{w_{s-1}w_s}$ interceptando $\Lambda\overline{e}_{i+1}v_{i+1}$ num ponto w'' (é garantido que isto ocorra se $\mathcal{K}_i$ for limitado) ou (quando $\mathcal{K}_i$ é ilimitado) alcançamos a cabeça da lista sem encontrar tal aresta. No primeiro caso, sendo $\mathcal{K}_i = \alpha w_s \dots w_{t-1}\beta$ (onde α e β são seqüências de vértices e arestas alternados), fazemos $\mathcal{K}_{i+1} \leftarrow \alpha w''\overline{e}_{i+1}w'\beta$; no segundo caso ($\mathcal{K}_i$ é ilimitado) precisamos testar se $\mathcal{K}_{i+1}$ é limitado ou ilimitado. Para isso, percorremos a fronteira de $\mathcal{K}_i$ no sentido anti-horário a partir de w_t a procura de uma aresta $\overline{w_{r-1}w_r}$ que intercepte $\Lambda\overline{e}_{i+1}v_{i+1}$ em um ponto w''. Se encontrarmos tal ponto, então sendo $\mathcal{K}_i = \gamma w_{t-1}\delta w_r\eta$ fazemos $\mathcal{K}_{i+1} \leftarrow \delta w''\overline{e}_{i+1}w'$ e a lista se torna circular. Caso cheguemos a cauda sem encontrar o ponto w'', então $\mathcal{K}_{i+1} \leftarrow \Lambda\overline{e}_{i+1}w'\beta$ também é ilimitado.

A escolha de q_{i+1}' é feita da seguinte maneira: se $\Lambda\overline{e}_{i+1}v_{i+1}$ tem apenas uma interseção com $\mathcal{K}_i$, então $q_{i+1}' \leftarrow$ (ponto no infinito de $\Lambda\overline{e}_{i+1}v_{i+1}$); senão $q_{i+1}' \leftarrow w''$. Para determinarmos q_{i+1}'', percorremos $\mathcal{K}_i$ no sentido anti-horário a partir de q_i'' até que um vértice w_u de $\mathcal{K}_i$ tal que w_{u+1} esteja à esquerda de $v_{i+1}\overline{v_{i+1}w_u}\Lambda$ seja encontrado, ou a lista de $\mathcal{K}_i$ termine sem que tal vértice seja encontrado. No primeiro caso, $q_{i+1}'' \leftarrow w_u$; no segundo (que só pode ocorrer quando $\mathcal{K}_i$ for ilimitado) $q_{i+1}'' \leftarrow q_i''$.

(b) *q_i' está à esquerda de $\Lambda\overline{e}_{i+1}v_{i+1}$.* [figura 3.23(b)] - Neste caso, $\mathcal{K}_{i+1} \leftarrow \mathcal{K}_i$, mas q_i' e q_i'' precisam ser atualizados. Para determinar q_{i+1}', percorremos $\mathcal{K}_i$ no sentido anti-horário a partir de q_i' até que encontremos um vértice w_t de $\mathcal{K}_i$ tal que w_{t+1} esteja à direita de $v_{i+1}\overline{v_{i+1}w_t}\Lambda$; então, fazemos $q_{i+1}' \leftarrow w_t$. A determinação de q_{i+1}'' é feita como no caso 1(a).

2. *Vértice v_i é convexo.* [figura 3.24(a, b)]

(a) *q_i'' está sobre ou à direita de $v_i\overline{e}_{i+1}\Lambda$.* [figura 3.24(a)] - Percorremos a fronteira de $\mathcal{K}_i$ no sentido horário a partir q_i'' até encontrarmos a aresta única $\overline{w_{t-1}w_t}$ de $\mathcal{K}_i$ que intercepta $v_i\overline{e}_{i+1}\Lambda$, ou alcançamos q_i' sem encontrar tal aresta. No segundo caso, o algoritmo termina ($K(\mathcal{P}) = \emptyset$). No primeiro caso, são realizadas as seguintes ações:

i. Encontramos a interseção w' de $\overline{w_{t-1}w_t}$ e $v_i\overline{e}_{i+1}\Lambda$.

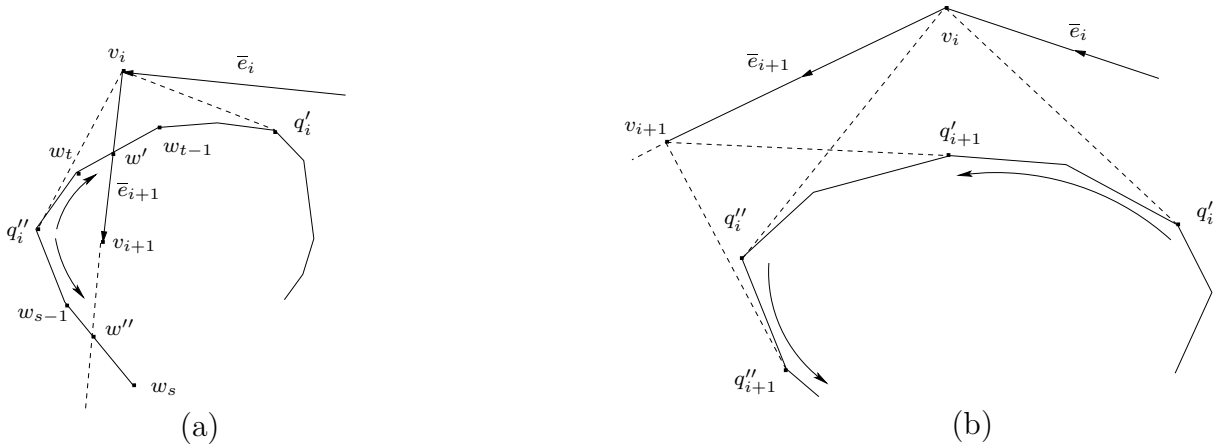


Figura 3.24: Passo geral quando v_i é convexo.

ii. Percorremos a fronteira de $\mathcal{K}_i$ no sentido anti-horário a partir de q''_i até atingirmos uma aresta $\overline{w_{s-1}w_s}$ interceptando $v_i\overline{e}_{i+1}\Lambda$ num ponto w'' (é garantido que isto ocorra se $\mathcal{K}_i$ for limitado) ou (quando $\mathcal{K}_i$ é ilimitado) alcançamos a cauda da lista sem encontrar tal aresta. No primeiro caso, sendo $\mathcal{K}_i = \alpha w_t \dots w_{s-1}\beta$, fazemos $\mathcal{K}_{i+1} \leftarrow \alpha w'\overline{e}_{i+1}w''\beta$; no segundo caso ($\mathcal{K}_i$ é ilimitado) precisamos testar se $\mathcal{K}_{i+1}$ é limitado ou ilimitado. Para isso, percorremos a fronteira de $\mathcal{K}_i$ no sentido horário a partir de w_{t-1} a procura de uma aresta $(w_{r-1}w_r)$ que intercepte $v_i\overline{e}_{i+1}\Lambda$ em um ponto w''. Se encontrarmos tal ponto, então sendo $\mathcal{K}_i = \gamma w_{r-1}\delta w_t\eta$ fazemos $\mathcal{K}_{i+1} \leftarrow \delta w'\overline{e}_{i+1}w''$ e a lista se torna circular. Caso cheguemos a cabeça sem encontrar o ponto w'', então $\mathcal{K}_{i+1} \leftarrow \alpha w'\overline{e}_{i+1}\Lambda$ também é ilimitado.

A escolha de q'_{i+1} e q''_{i+1} depende da posição de v_{i+1} com relação ao raio $v_i\overline{e}_{i+1}\Lambda$ e se $v_i\overline{e}_{i+1}\Lambda$ tem uma ou duas interseções com $\mathcal{K}_i$. Os dois casos são distinguidos a seguir:

i. $v_i\overline{e}_{i+1}\Lambda$ intercepta $\mathcal{K}_i$ em w' e w''. Se $v_{i+1} \in [v_i\overline{e}_{i+1}w']$ então q'_{i+1} é selecionado como no caso 1(b). Senão $q'_{i+1} \leftarrow w'$. Se $v_{i+1} \in [v_i\overline{e}_{i+1}w'']$ então $q''_{i+1} \leftarrow w''$. Senão, q''_{i+1} é selecionado como no caso 1(a), exceto que percorremos $\mathcal{K}_{i+1}$ no sentido anti-horário a partir de w''.

ii. $v_i\overline{e}_{i+1}\Lambda$ intercepta $\mathcal{K}_i$ em w'. Se $v_{i+1} \in [v_i\overline{e}_{i+1}w']$ então q'_{i+1} é selecionado como no caso 1(b). Senão, $q'_{i+1} \leftarrow w'$. $q''_{i+1} \leftarrow$ ponto no infinito de $v_i\overline{e}_{i+1}\Lambda$.

(b) q''_i *está à esquerda de* $v_i\overline{e}_{i+1}\Lambda$. [figura 3.24(b)] - Neste caso, $\mathcal{K}_{i+1} \leftarrow \mathcal{K}_i$. q'_{i+1} é determinado como em 1(b). Se $\mathcal{K}_i$ é limitado, então q''_{i+1} é determinado como em 1(a); senão, $q''_{i+1} \leftarrow q''_i$.

Como provado por Lee e Preparata em [LP79], este algoritmo determina corretamente o núcleo de um polígono simples. Porém, para alguns polígonos, o algoritmo executa em tempo $O(n^2)$. Uma instância onde isto ocorre pode ser vista na figura 3.25. Para evitarmos estes casos e garantir a complexidade $O(n)$ é preciso fazer uma nova verificação no polígono de entrada.

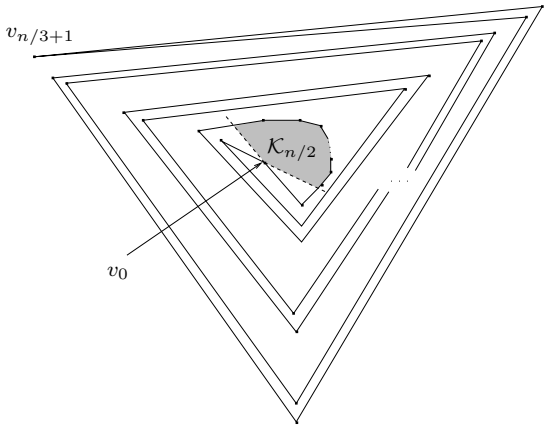


Figura 3.25: Instância onde o algoritmo termina em tempo $\Theta(n^2)$. O polígono $\mathcal{K}_{n/2}$, com n lados, é mostrado pela área hachurada. O algoritmo dá $n/12$ voltas ao redor de $\mathcal{K}_{n/2}$ antes de alcançar o vértice $v_{n/3+1}$, onde é descoberto que $K(\mathcal{P}) = \emptyset$.

Considere as duas retas de suporte de $\mathcal{K}_i$ passando pelo vértice v_0 de $\mathcal{P}$. Sejam $\overline{r}'$ e $\overline{r}''$ os dois raios destas retas que partem de v_0 e contêm os pontos de suporte, e de forma que o raio $\overline{r}'$ deixe $\mathcal{K}_i$ no meio-plano à sua direita e $\overline{r}''$ deixe $\mathcal{K}_i$ à sua esquerda (figura 3.26). Sejam também $\overline{s}'$ o segmento entre v_0 e o ponto de suporte mais afastado de v_0 sobre $\overline{r}'$, e $(\overline{r}')^*$ o raio complementar de $\overline{r}'$. $\overline{s}''$ e $(\overline{r}'')^*$ são definidos analogamente.

Assim, como provado por Lee e Preparata, temos o seguinte teorema:

Teorema 3.7.1 *[LP79] Se* $\mathcal{K}_{i+1} \neq \emptyset$ *e* $\overline{e}_{i+1}$ *intercepta* $\overline{s}''$ *ou* $(\overline{r}')^*$*, com* v_{i+1} *na região convexa delimitada por* $(\overline{r}')^*$ *e* $\overline{r}''$ *(área hachurada na figura 3.26); então* $K(\mathcal{P}) = \emptyset$.

A prova completa do teorema será omitida por sua complexidade, mas ela se baseia no fato de que, se $\overline{e}_{i+1}$ intercepta $\overline{r}''$ ou $(\overline{r}')^*$, com v_{i+1} na região convexa delimitada por $(\overline{r}')^*$ e $\overline{r}''$ então ou $\overline{e}_{i+1}$ separa $\mathcal{K}_{i+1}$ de v_0 [figura 3.26(a)] ou v_0 e suas arestas separam $\mathcal{K}_{i+1}$ de um ponto u sobre $\overline{e}_{i+1}$ [figura 3.26(b)], o que, em ambos os casos, implica $K(\mathcal{P}) = \emptyset$.

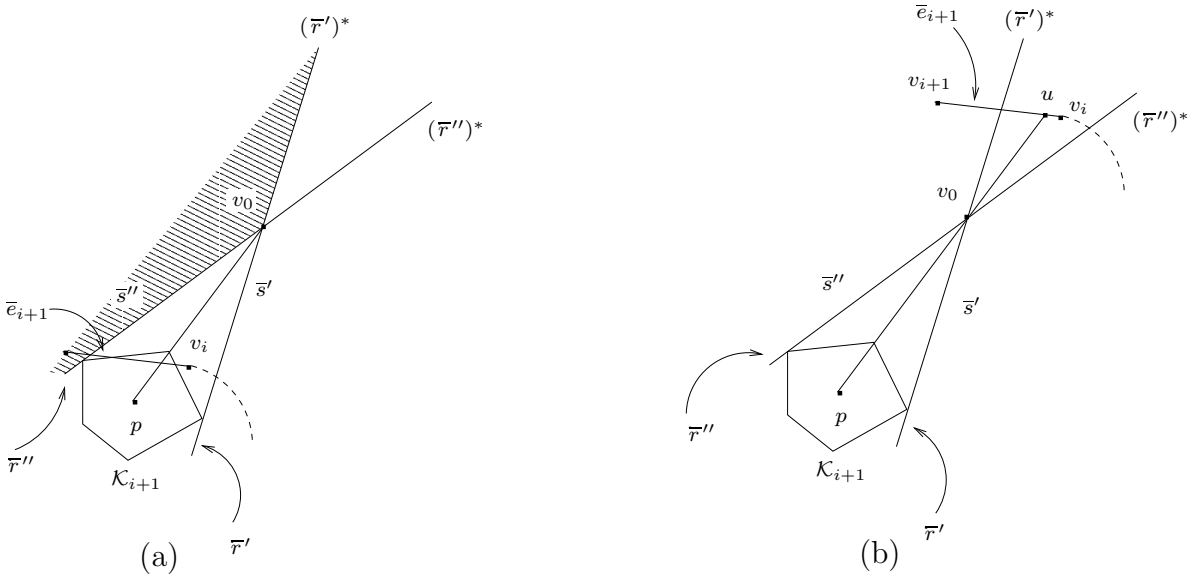


Figura 3.26: Ilustração da prova do teorema 3.7.1.

Através da verificação indicada pelo teorema 3.7.1 o algoritmo pára assim que a fronteira de $\mathcal{P}$ der uma volta e meia em torno de seu núcleo, como no caso da figura 3.25, evitando o tempo $O(n^2)$. Os exemplos gerais que são evitados com a verificação das condições do teorema são mostrados na figura 3.27.

Assim, o passo geral do algoritmo deve ser modificado, adicionando-se as seguintes operações: (1) antes de determinar $\mathcal{K}_{i+1}, q'_{i+1}$ e q''_{i+1}: se $\overline{e}_{i+1}$ intercepta $\overline{s}''$ ou $(\overline{r}')^*$, com v_{i+1} na região convexa delimitada por $(\overline{r}')^*$ e $\overline{r}''$, então termine o algoritmo com $K(\mathcal{P}) = \emptyset$. (2) nos casos 1(a) e 2(a), depois de determinados $\mathcal{K}_{i+1}, q'_{i+1}$ e q''_{i+1} denote por p' e p'' os pontos de suporte em $\overline{r}'$ e $\overline{r}''$, respectivamente. Inicialmente, como $\mathcal{K}_1$ é $\Lambda\overline{e}_1 v_0 \overline{e}_0 \Lambda$, p' e p'' são os pontos no infinito em $\Lambda\overline{e}_1 v_0$ e $v_0\overline{e}_0\Lambda$, respectivamente. Se na obtenção de $\mathcal{K}_{i+1}$ a partir e $\mathcal{K}_i$, os vértices p' e/ou p'' forem apagados, o vértice apagado deve ser atualizados da seguinte maneira: (i) v_i *é reflexo:* $p' \leftarrow w', p'' \leftarrow w''$ se v_0 está a esquerda de $\Lambda\overline{e}_{i+1}v_{i+1}$ e $p' \leftarrow w'', p'' \leftarrow w'$ caso contrário. (ii) v_i *é convexo:* $p' \leftarrow w'', p'' \leftarrow w'$ se v_0 está a esquerda de $v_i\overline{e}_{i+1}\Lambda$ e $p' \leftarrow w', p'' \leftarrow w''$ caso contrário. Note que os pontos w' e w'' são construídos como em 1(a) e 2(a), e, se w'' não existir, seu lugar é tomado pelo ponto no infinito no raio sendo considerado.

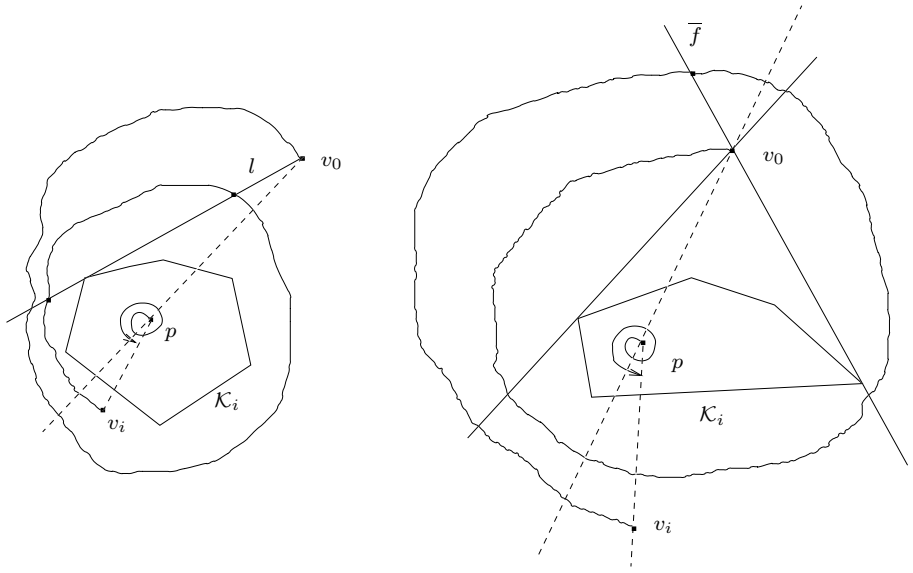


Figura 3.27: Exemplos onde a construção do núcleo pára após detectada a não satisfação das condições do teorema 3.7.1

Para analisarmos a complexidade deste algoritmo, precisamos analisar dois ações separadas. A primeira é o cálculo da interseção de H_{i+1} com $\mathcal{K}_i$ e a segunda são as atualizações de q_i', q_i'', p' e p''.

Na primeira ação, para o caso 1(a), quando o algoritmo não termina, visitamos um número a_i de vértices na fronteira de $\mathcal{K}_i$ antes de encontrarmos w' e w''. O processo então remove $(a_i - 2)$ arestas de $\mathcal{K}_i$ (aquelas entre w' e w''). Como cada aresta de $\mathcal{K}_i$ é colinear a uma aresta de $\mathcal{P}$, temos que $\sum(a_i - 2) \leq n$. portanto, o número total de vértices visitados $\sum a_i$ é $O(n)$. O mesmo argumento é utilizado para o caso 2(a).

Na segunda ação, podemos observar que q_i' e q_i'' sempre avançam sobre a fronteira de $\mathcal{K}_i$ para que possamos achar q_{i+1}' e q_{i+1}''. Considere o número de arestas visitadas durante o avanço de um dos vértices. Como cada aresta de $\mathcal{K}_i$ corresponde a uma aresta de $\mathcal{P}$, se cada aresta de $\mathcal{K}_i$ for visitada mais de duas vezes, teremos dado mais de duas voltas sobre a fronteira de $\mathcal{P}$. Por outro lado, a verificação adicional colocada a partir do teorema 3.7.1 nos garante que esse avanço pára sempre que a fronteira de $\mathcal{P}$ der mais de uma volta e meia em torno de si mesmo. Assim, a visita dos vértices da fronteira de $\mathcal{K}_i$ para a atualização de q_i' e q_i'' também é $O(n)$. Por fim, a atualização de p' e p'' é feita diretamente quando encontramos w' e w'', o que faz com que o algoritmo seja linear no número de vértices do polígono de entrada.

Passemos agora a análise desse algoritmo sobre a geometria $\mathcal{G}_{\mathcal{F}}$.

3.7.2 Geometria $\mathcal{G}_{\mathcal{F}}$

Como os vértices do $\mathcal{G}_{\mathcal{F}}$-polígono de entrada $\mathcal{P}_{\mathcal{F}}$ são dados no sentido anti-horário, isto é, seu interior está sempre à esquerda de qualquer $\mathcal{G}_{\mathcal{F}}$-aresta, o núcleo de $\mathcal{P}_{\mathcal{F}}$ é dado pela interseção dos $\mathcal{G}_{\mathcal{F}}$-meios-planos à esquerda de suas $\mathcal{G}_{\mathcal{F}}$-arestas.

Além disso, através da definição de vértice convexo e reflexo dadas no começo desta seção, podemos caracterizar um $\mathcal{G}_{\mathcal{F}}$-polígono convexo:

Proposição 3.7.1 *Um $\mathcal{G}_{\mathcal{F}}$-polígono é $\mathcal{G}_{\mathcal{F}}$-convexo se, e somente se, não possui vértices reflexos.*

Prova: $\Leftarrow$ Por absurdo, suponha que o $\mathcal{G}_{\mathcal{F}}$-polígono não possua vértices reflexos e seja não $\mathcal{G}_{\mathcal{F}}$-convexo. Assim, deve existir uma $\mathcal{G}_{\mathcal{F}}$-reta ℓ cuja interseção com o $\mathcal{G}_{\mathcal{F}}$-polígono é não-conexa. Então, ℓ deve atravessar o $\mathcal{G}_{\mathcal{F}}$-polígono, sair por uma $\mathcal{G}_{\mathcal{F}}$-aresta $\widetilde{e}$ qualquer, e depois passar pelo mesmo novamente [figura 3.28(a)]. Porém, como o $\mathcal{G}_{\mathcal{F}}$-polígono não possui vértices reflexos, todo seu interior está no $\mathcal{G}_{\mathcal{F}}$-meio-plano à esquerda da $\mathcal{G}_{\mathcal{F}}$-reta que contém $\widetilde{e}$. Assim, para que ℓ possa passar pelo $\mathcal{G}_{\mathcal{F}}$-polígono novamente, esta tem que atravessar a $\mathcal{G}_{\mathcal{F}}$-reta que contém $\widetilde{e}$. Como ℓ já intercepta $\widetilde{e}$, isso não pode acontecer pela definição da geometria. Portanto, o $\mathcal{G}_{\mathcal{F}}$-polígono tem que ser $\mathcal{G}_{\mathcal{F}}$-convexo.

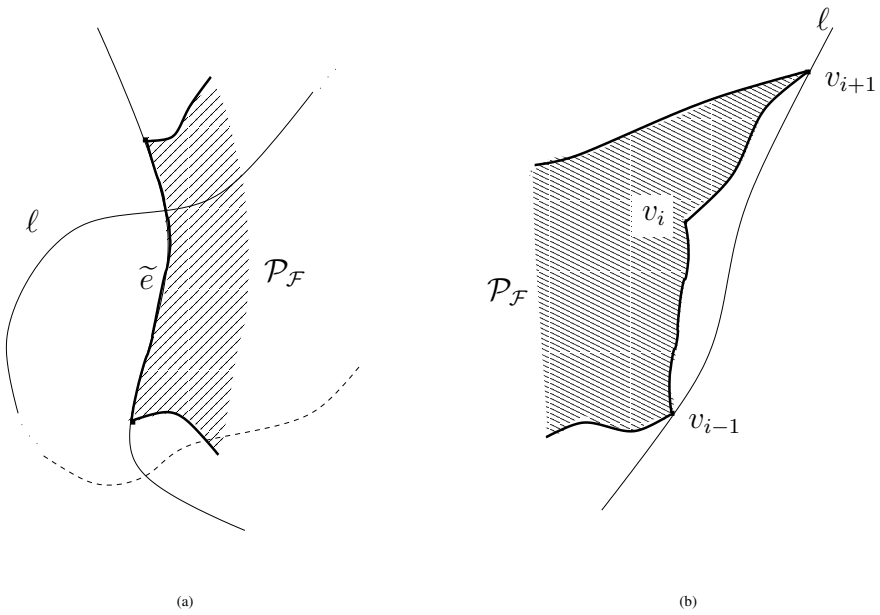


Figura 3.28: $\mathcal{P}_{\mathcal{F}}$ é $\mathcal{G}_{\mathcal{F}}$-convexo se, e somente se, não possui vértices reflexos.

⇒ Suponha que o $\mathcal{G}_\mathcal{F}$-polígono possua um vértice reflexo, com v_{i+1} à direita da $\mathcal{G}_\mathcal{F}$-reta que contém e $\widetilde{v_{i-1}v_i}$ [figura 3.28(b)]. Seja ℓ a $\mathcal{G}_\mathcal{F}$-reta que passa por v_{i-1} e por v_{i+1}. Como v_i está à esquerda de ℓ (orientada de v_{i-1} para v_{i+1}), o triângulo $v_{i-1}v_{i+1}v_i$ não intercepta o interior do $\mathcal{G}_\mathcal{F}$-polígono. Logo, o $\mathcal{G}_\mathcal{F}$-segmento $\widetilde{v_{i-1}v_{i+1}}$ liga dois pontos do $\mathcal{G}_\mathcal{F}$-polígono mas não está contido nele. Portanto, o polígono não pode ser $\mathcal{G}_\mathcal{F}$-convexo. ■

Com a proposição 3.7.1, vemos que, como no caso euclidiano, se $\mathcal{P}_\mathcal{F}$ não tiver vértices reflexos, então $K(\mathcal{P}_\mathcal{F}) = \mathcal{P}_\mathcal{F}$.

O algoritmo proposto por Lee e Preparata para o caso euclidiano apresentado anteriormente faz, no passo geral, a interseção dos meios-planos à esquerda de cada aresta $\overline{e}_i$ com o polígono $\mathcal{K}_{i-1}$. Para isso, elimina de $\mathcal{K}_{i-1}$ a sua interseção com o meio plano à direita da reta que passa por $\overline{e}_i$ através do uso da orientação de trios de pontos e do cálculo de interseções. Como na geometria $\mathcal{G}_\mathcal{F}$ o núcleo de um $\mathcal{G}_\mathcal{F}$-polígono também é dado pela interseção dos $\mathcal{G}_\mathcal{F}$-meios-planos à esquerda das $\mathcal{G}_\mathcal{F}$-retas que contêm as $\mathcal{G}_\mathcal{F}$-arestas, e conseguimos resolver as duas primitivas básicas, podemos fazer uso do mesmo algoritmo também nesta geometria.

Assim, o passo inicial também pode ser tomado como a interseção dos $\mathcal{G}_\mathcal{F}$-meios planos definidos pelas $\mathcal{G}_\mathcal{F}$-retas que contêm $\widetilde{e}_0$ e $\widetilde{e}_1$ e orientadas como tais. Já no passo geral, podemos calcular $\mathcal{K}_{i+1}$ e os pontos q_i', q_i'', p' e p'' de forma análoga ao caso euclidiano, mas utilizando as primitivas da $\mathcal{G}_\mathcal{F}$.

Seja $\mathcal{H}_i$ o $\mathcal{G}_\mathcal{F}$-meio-plano à esquerda da $\mathcal{G}_\mathcal{F}$-reta que contém $\widetilde{e}_i$ e orientada como tal.

Teorema 3.7.2 *O $\mathcal{G}_\mathcal{F}$-polígono $\mathcal{K}_{i+1}$, construído conforme o algoritmo mostrado na seção anterior, sobre uma geometria $\mathcal{G}_\mathcal{F}$, é a interseção de $\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_{i+1}$, para $i = 0, 1, \ldots, n-2$.*

Prova: Por indução finita fraca.

Caso base: $\mathcal{K}_1$ é, por definição, a interseção de $\mathcal{H}_0$ e $\mathcal{H}_1$, dado no passo inicial do algoritmo.

Hipótese de indução: $\mathcal{K}_i = \mathcal{H}_0 \cap \mathcal{H}_1 \cap \cdots \cap \mathcal{H}_i$.

Passo da indução: Em todos os casos contemplados no passo geral do algoritmo, fazemos a interseção de $\mathcal{K}_i$ com $\mathcal{H}_{i+1}$, logo, pela hipótese de indução, $\mathcal{K}_{i+1} = \mathcal{H}_0 \cap \mathcal{H}_1 \cap \cdots \cap \mathcal{H}_{i+1}$.

■

Assim, o teorema 3.7.2 garante que o algoritmo dado na seção anterior contrói o núcleo $K(\mathcal{P}_\mathcal{F})$ na geometria $\mathcal{G}_\mathcal{F}$. Como as únicas modificações feitas no algoritmo são na

resolução das primitivas de orientação de trios de pontos e do cálculo de interseções, sua complexidade fica alterada apenas pelas constantes multiplicativas discutidas no apêndice A, sendo sua complexidade assintótica a mesma do caso euclidiano.

Capítulo 4

Visualizador da Geometria $\mathcal{G}_{\mathcal{F}}$

Neste capítulo será descrito o software `GFViewer`, que é um visualizador de figuras geométricas para a geometria $\mathcal{G}_{\mathcal{F}}$, desenvolvido como parte deste trabalho.

Por lidarmos todos os dias com noções geométricas euclidianas, é normal surgirem dificuldades para se passar a pensar na $\mathcal{G}_{\mathcal{F}}$. A fim de aumentarmos a nossa intuição, entendermos melhor o comportamento das figuras geométricas e algoritmos aplicados às mesmas, na geometria $\mathcal{G}_{\mathcal{F}}$, desenvolvemos um visualizador para essa geometria, o qual permite a construção de figuras, interação com as mesmas e a execução de algoritmos.

A seção 4.1 apresenta o abiente de desenvolvimento e execução do aplicativo. Na seção 4.2 são mostrados os objetos geométricos presentes no `GFViewer`. As funcionalidades do software e sua interface são discutidas na seção 4.3. Por último, nas seções 4.4 e 4.5 são mostradas, respectivamente, as famílias de $\mathcal{G}_{\mathcal{F}}$-retas e os algoritmos implementados no programa, bem como alguns detalhes desta implementação.

4.1 Ambiente

O `GFViewer` é uma aplicação desenvolvida em C++, podendo ser compilado tanto para Linux quanto para Microsoft Windows. O software faz uso da biblioteca gráfica OpenGL para a visualização das figuras geométricas na tela. Para a criação de janelas e a interface com o usuário, foi utilizado o framework para desenvolvimento de aplicações Qt. Sua característica de desenvolvimento de aplicações multi-plataforma tornou possível fazer um visualizador que independe do sistema operacional, desde que haja a a disponibilidade do framework Qt e de OpenGL. A figura 4.1 mostra a janela do visualizador `GFViewer`.

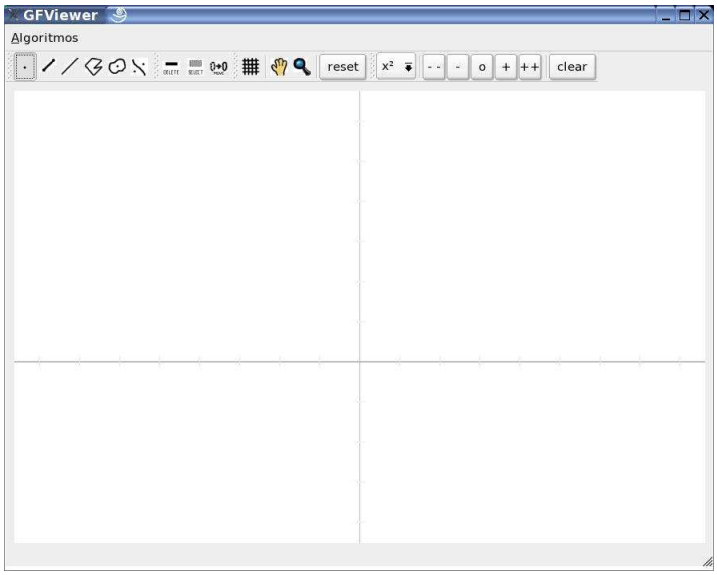


Figura 4.1: Interface do visualizador `GFViewer`.

4.2 Objetos Geométricos

O `GFViewer` provê a criação e manipulação de uma série de objetos geométricos básicos, que por sua vez servem de entrada para a execução de algoritmos. A natureza de cada um dos objetos está bem definida pela especificação da geometria $\mathcal{G}_\mathcal{F}$, porém, sua forma geométrica depende da $\mathcal{G}_\mathcal{F}$-reta escolhida. Os objetos presentes no visualizador são:

- Ponto - um ponto em $\mathbb{R}^2$, definido por um par de coordenadas (x, y).
- $\mathcal{G}_\mathcal{F}$-reta - uma $\mathcal{G}_\mathcal{F}$-reta definida por dois pontos.
- $\mathcal{G}_\mathcal{F}$-segmento - um $\mathcal{G}_\mathcal{F}$-segmento ligando dois pontos.
- $\mathcal{G}_\mathcal{F}$-polígono - uma sequência de vértices, interligados por $\mathcal{G}_\mathcal{F}$-segmentos.
- $\mathcal{G}_\mathcal{F}$-círculo - um $\mathcal{G}_\mathcal{F}$-círculo é definido por um centro c e um ponto p em sua borda, consistindo-se do lugar geométrico dos pontos eqüidistantes de c à distância $d_\mathcal{F}(c, p)$[1].
- $\mathcal{G}_\mathcal{F}$-bissetor - um $\mathcal{G}_\mathcal{F}$-bissetor é definido por dois pontos p_1 e p_2, consistindo-se do lugar geométrico dos pontos eqüidistantes de p_1 e p_2, segundo a distância $d_\mathcal{F}$.

Como vemos, apenas os pontos não dependem do tipo de $\mathcal{G}_\mathcal{F}$-reta escolhida, ficando o formato de todos os outros objetos dependentes da escolha.

[1] Veja seção 2.1

4.3 Funcionalidades

Nesta seção serão mostradas as principais funcionalidades do `GFViewer`. A interface do `GFViewer` permite a criação e manipulação dos objetos básicos, assim como a interação entre eles. A interação com o visualizador é feita em diferentes modos, como os de criação, deleção, movimentação e seleção. Cada um desses modos será detalhado nas subseções seguintes. Para entrar no modo de criação, é preciso selecionar um botão na barra mostrada na figura 4.2. Já os demais modos são acionados pelos botões mostrados na figura 4.3.

Figura 4.2: Barra de ferramentas para criação de pontos, $\mathcal{G}_{\mathcal{F}}$-segmentos, $\mathcal{G}_{\mathcal{F}}$-retas, $\mathcal{G}_{\mathcal{F}}$-polígonos, $\mathcal{G}_{\mathcal{F}}$-círculos e $\mathcal{G}_{\mathcal{F}}$-bissetores.

Figura 4.3: Barra de ferramentas para seleção dos modos de deleção, seleção e movimentação.

Além desses, ainda pode-se interagir com a posição de visualização do plano através da barra de ferramenta de visualização, mostrada na figura 4.4. O primeiro botão desta barra permite a habilitação e desabilitação das linhas de grade no plano, para facilitar a localização dos pontos e objetos. O segundo botão permite a movimentação lateral (*pan*) do ponto de visão do plano, que é feita através do clique e arraste do mouse. Quando selecionado o modo de *zoom*, indicado pelo terceiro botão, ao clicarmos na tela e arrastarmos o mouse para cima ou para baixo a proximidade do ponto de visualização com o plano é alterada. Porém, o *zoom* pode ser alterado em qualquer outro modo através do botão de rolagem (*scroll*) do mouse. Por fim, o botão *reset* retorna o ponto de visão do plano para o valor inicial, cancelando qualquer alteração de *zoom* e *pan* feitas até aquele momento.

Figura 4.4: Barra de ferramentas para ajustes de visualização.

4.3.1 Modo de Criação de Objetos

Através da seleção de um dos botões da barra mostrada na figura 4.2, entra-se no modo de criação de novos objetos. Quando na criação de uma $\mathcal{G}_{\mathcal{F}}$-reta, $\mathcal{G}_{\mathcal{F}}$-segmento, $\mathcal{G}_{\mathcal{F}}$-polígono, $\mathcal{G}_{\mathcal{F}}$-círculo ou $\mathcal{G}_{\mathcal{F}}$-bissetor, se clicarmos em cima de um ponto já existente, aquele ponto é utilizado para a criação do novo objeto. Desta forma, vários objetos podem ficar dependentes de um mesmo ponto. Isso faz com que a movimentação ou deleção desse ponto cause mudanças em todos objetos que dependam dele.

4.3.2 Modo de Deleção de Objetos

Selecionando o botão de deleção na barra de ferramentas é possível destruir qualquer objecto criado anteriormente. Uma vez neste modo, ao clicarmos sobre um ponto, o mesmo é apagado, junto com todos outros objetos que foram construídos a partir dele. No caso de um polígono, a exclusão de um ponto leva à exclusão do vértice correspondente àquele ponto, com a destruição de duas arestas e a construção de uma nova.

Se o clique dado não acertar nenhum ponto, então é feito o teste para saber se algum objeto foi clicado. Neste caso, todos os objetos que foram clicados são apagados. É importante salientar que a remoção de um objeto não implica na remoção dos pontos que o criaram, uma vez que estes pontos podem pertencer a outros objetos.

4.3.3 Modo de Seleção de Objetos

O modo de seleção de objetos nos permite escolher quais objetos serão utilizados como entrada para a execução de algoritmos. Toda vez que um algoritmo é escolhido para ser executado, o programa testa para saber se os objetos selecionados satisfazem a entrada daquele algoritmo. Caso isso não seja satisfeito, o algoritmo não tem como ser executado. Um ponto selecionado tem o seu tamanho aumentado. Os demais objetos têm sua linha tornada mais espessa quando selecionados.

A seleção de objetos faz com que seu estado seja negado, assim, se selecionarmos um objeto que não estava selecionado, este passa a estar, e vice-versa, tornando fácil a deseleção de um objeto.

Pode-se selecionar objetos de duas maneiras. A primeira delas é através de um clique do mouse. Se clicarmos sobre um ponto, o mesmo é selecionado. Se não acertarmos nenhum ponto, então é testado para ver se o clique acertou algum outro objeto, que passa a ser selecionado.

O segundo método é através de um retângulo de seleção, criado quando o mouse é clicado e arrastado neste modo. Todos os pontos dentro do retângulo de seleção são selecionados. São selecionados também todos os objetos cujos pontos utilizados em sua criação também estão dentro do retângulo.

4.3.4 Modo de Movimentação de Objetos

O modo de movimentação permite tanto o deslocamento como também o redimensionamento de objetos. Isto é possível, pois ao movermos um ponto, todas os objetos que foram criados a partir dele também são alterados. Neste modo é possível movimentar o ponto, modificando os objetos que o contêm, ou simplesmente transladar um objeto. É importante notar que a translação de um objeto causa a translação dos pontos que o formaram, podendo levar ao redimensionamento ou translação de outros objetos.

4.4 $\mathcal{G}_\mathcal{F}$-retas

Como descrito do capítulo 2, a geometria $\mathcal{G}_\mathcal{F}$ é baseada em uma família de curvas escolhida de forma que esta satisfaça as três condições básicas apresentadas. Além disso, também no capítulo 2, é feita uma introdução sobre a escolha de curvas baseada em curvas geradoras.

No `GFViewer` foram implementadas quatro diferentes famílias de curvas, podendo estas serem selecionadar a partir do menu mostrado na figura 4.5.

Figura 4.5: $\mathcal{G}_\mathcal{F}$-retas existentes no visualizador.

A primeira opção, mostrada como "x", faz com que as retas criadas sejam lineares, tornando o visualizador um ambiente que trata a geometria euclidiana. A segunda opção, a "x^2", escolhe as $\mathcal{G}_\mathcal{F}$-retas como parábolas com derivada segunda constante e positiva, junto com as retas verticais. Já a opção "$-x^2$" é a família de parábolas com derivada

segunda também constante, mas negativa, junto com as retas verticais. A última opção, "$1/x$", é a família de $\mathcal{G}_\mathcal{F}$-retas formadas pelas seguintes geradoras:

- Curva formada pela função $f(x) = a/x$ para $x > 0$ e com a constante;
- Retas $f(x) = bx$ para $b \in [0, \infty)$;
- Reta $x = 0$.

Figura 4.6: Barra de ferramentas utilizada para alterar os valores das constantes multiplicativas da família de curvas, gerando novas geometrias.

É importante notar que, na verdade, numa mesma opção dentre x^2, $-x^2$ ou $1/x$ temos infinitas opções de famílias de curvas, porque o programa permite variar o valor da segunda derivada nos dois primeiros casos e o valor da constante a no terceiro caso. Cada mudança nesses valores gera uma outra geometria $\mathcal{G}_\mathcal{F}$. Os valores são alterados a partir da barra de ferramentas mostrada na figura 4.6. Nesta barra, os botões $++$ e $+$ aumentam os valores das constantes em 50% e 10% respectivamente. Já os botões $--$ e $-$ diminuem as constantes pelos mesmos fatores dos casos anteriores. O botão "o" retorna o valor da constante múltiplicativa daquela família de curvas para o valor inicial. A figura 4.7 mostra um conjunto de objetos geométricos gerados a partir dos mesmos pontos em diferentes geometrias.

4.4.1 Implementação de Novas $\mathcal{G}_\mathcal{F}$-retas

O desenvolvimendo do visualizador foi projetado de forma a facilitar a adição de novas $\mathcal{G}_\mathcal{F}$-retas. Cada tipo de $\mathcal{G}_\mathcal{F}$-reta é representada por uma classe que estende a classe abstrata `GFLine`. Esta classe possui os métodos virtuais necessários para a manipulação correta daquela $\mathcal{G}_\mathcal{F}$-reta pelo programa. Na figura 4.8 é mostrado um pequeno diagrama de classes para as $\mathcal{G}_\mathcal{F}$-retas já implementadas.

Considerando `r` um objeto de uma classe que estenda `GFLine`, e que, portanto, contenha dois pontos `p1` e `p2`, podemos destacar os seguintes métodos herdados da classe `GFLine`:

`bool hasPoint(CPoint p)` - este método deve retornar *true* se o ponto `p` pertence a `r`.

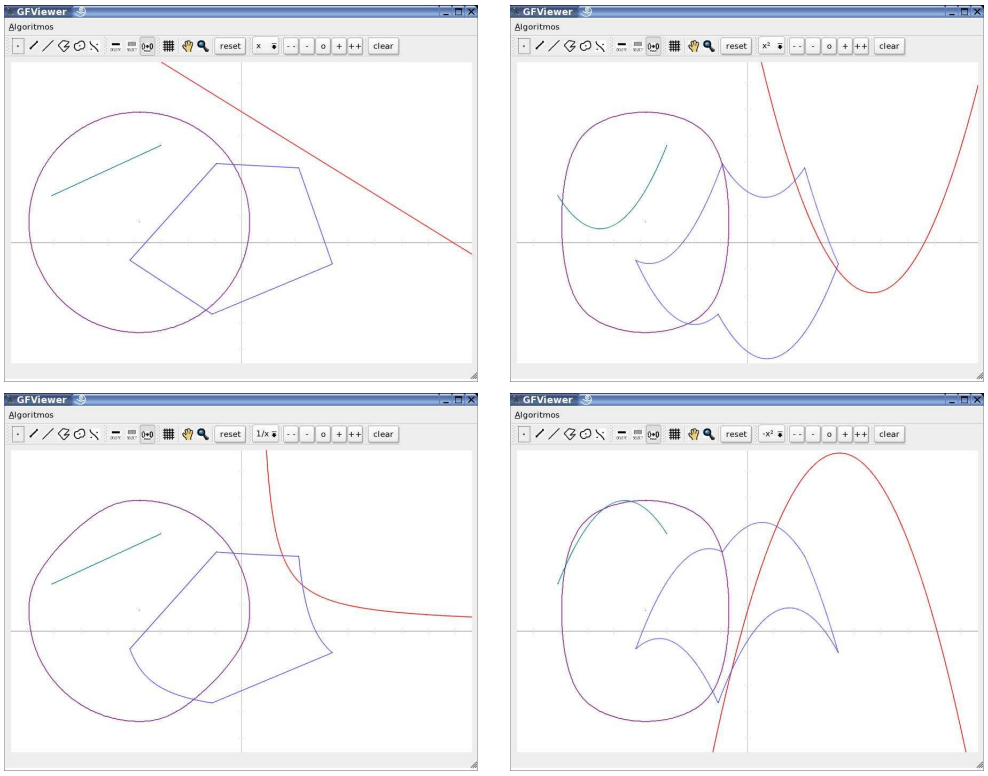


Figura 4.7: Objetos geométricos criados a partir do mesmo conjunto de pontos em diferentes geometrias. No sentido horário: $x, x^2, -x^2$ e $1/x$.

`virtual bool hasPointInSegment(CPoint p)` - este método deve retornar *true* se o segmento entre os pontos `p1` e `p2` contém o ponto `p`.

`double distance(CPoint q1, CPoint q2)` - este método implementa a primitiva de distância definida sobre `r`, entre `q1` e `q2` na geometria. O apêndice B discute a dificuldade em se calcular distâncias na geometria $\mathcal{G}_\mathcal{F}$.

`int orientation(CPoint p3)` - este método resolve a primitiva de orientação entre três pontos. O método devolve -1 se o trio de pontos (`p1`, `p2`, `p3`) tem orientação negativa, $+1$ se o pontos têm orientação positiva e 0 se eles forem $\mathcal{G}_\mathcal{F}$-colineares.

`bool intersectRaySegment(GFLine *l)` - este método verifica se o $\mathcal{G}_\mathcal{F}$-raio que parte de `p1` para `p2` intercepta o $\mathcal{G}_\mathcal{F}$-segmento formado pelos pontos usados na construção da $\mathcal{G}_\mathcal{F}$-reta `l`.

Como vemos, através destes métodos abstratos, a classe `GFLine` requer que qualquer $\mathcal{G}_\mathcal{F}$-reta criada implemente as primitivas básicas necessárias para o funcionamento dos

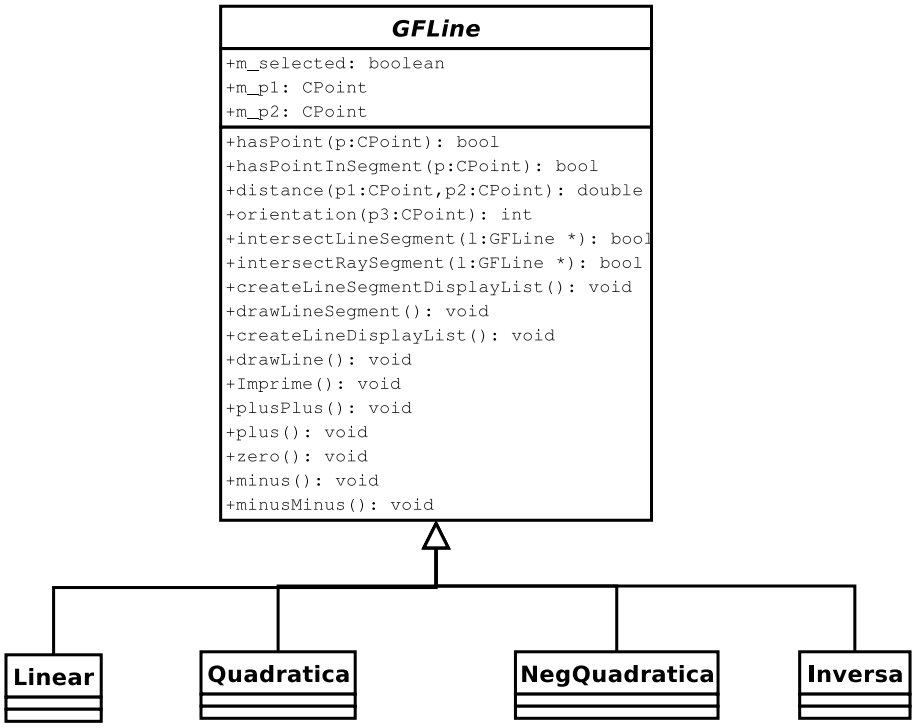


Figura 4.8: Diagrama de classes para representação de $\mathcal{G}_{\mathcal{F}}$-retas. As classes `Linear`, `Quadratica`, `NegQuadratica` e `Inversa` são as implementações das famílias de curvas $x, x^2, -x^2$ e $1/x$ respectivamente.

algoritmos e do resto do programa. Já as primitivas que independam do tipo de $\mathcal{G}_{\mathcal{F}}$-reta implementada, como por exemplo a verificação de interseção entre $\mathcal{G}_{\mathcal{F}}$-reta e $\mathcal{G}_{\mathcal{F}}$-segmento, que é feita através da verificação de duas orientações, são implementadas pela classe base.

4.5 Algoritmos

Além da criação e manipulação de objetos, foi também inserido no `GFViewer` um ambiente para a execução de algoritmos. Os algoritmos a serem executados devem ser selecionados no menu principal do programa, como mostrado na figura 4.9. Uma vez selecionados os algoritmos, o programa tenta executá-los a cada movimentação de objetos ou modificação no conjunto de objetos selecionados. Se ao tentar ser executado, os objetos selecionados não provêm as entradas necessárias para o algoritmo, a execução não acontece.

Dois algoritmos foram implementados no `GFViewer`, o de localização de pontos em um $\mathcal{G}_{\mathcal{F}}$-polígono simples, discutido na seção 3.1.1 e o de construção de $\mathcal{G}_{\mathcal{F}}$-casco convexo

incremental, descrito na seção 3.5.2[2].

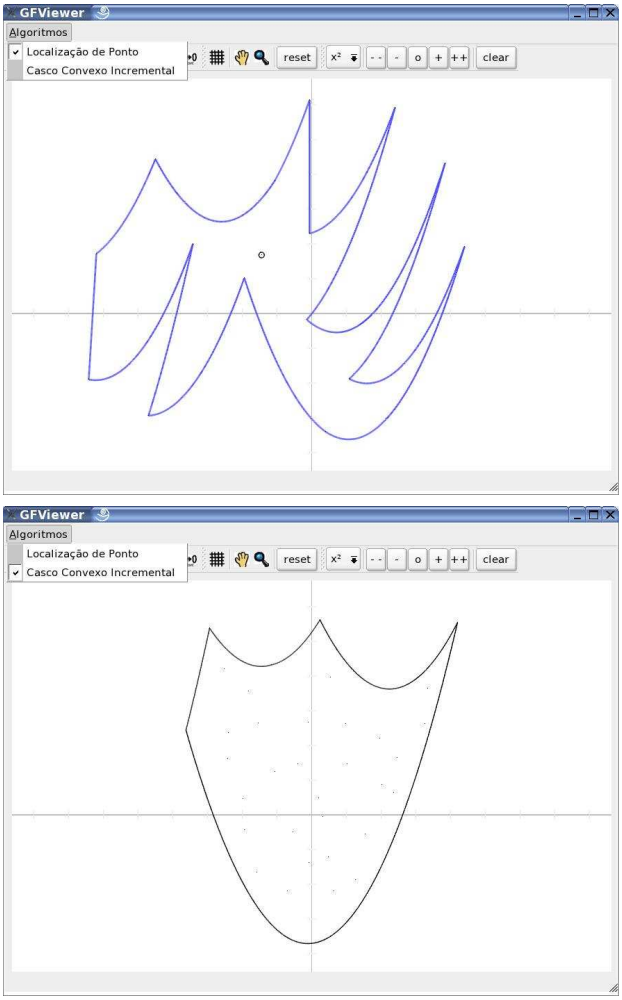


Figura 4.9: Um exemplo da execução dos algoritmos. Na figura superior, o pequeno círculo preto em volta do ponto indica que este é interno ao $\mathcal{G}_{\mathcal{F}}$-polígono selecionado (em azul). Na segunda figura, o polígono preto é o casco $\mathcal{G}_{\mathcal{F}}$-convexo do conjunto de pontos selecionado.

[2]Esses algoritmos foram implementados com a colaboração de Flávio Ivan da Silva.

4.5.1 Implementação de Novos Algoritmos

A estrutura utilizada no desenvolvimento do programa visou também tornar o mais simples possível a inclusão, futuramente, de novos algoritmos. Assim, foi desenvolvida uma classe `Algorithms` que faz o gerenciamento de todos algoritmos e a interface dos mesmos com o resto da aplicação. Esta classe é responsável pela criação e pelo gerenciamento dos objetos que representam cada algoritmo existente no programa. Cada algoritmo deve ser implementado como uma classe que estende a classe abstrata `Algorithm`, mostrada na figura 4.10. Essa classe obriga que uma classe que represente um algoritmo, e portanto a estenda, implemente duas funções:

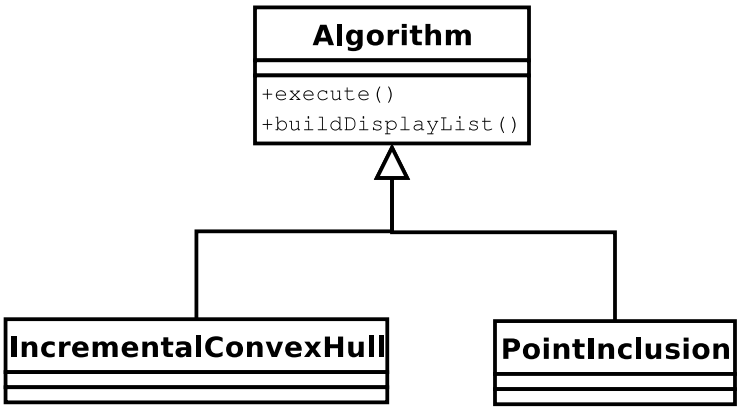


Figura 4.10: Diagrama de classes para a implementação de algoritmos. Cada nova classe, com um novo algoritmo deve estender a classe `Algorithm`.

`long execute(GFState *s)` - este é o método responsável por executar o algoritmo. Antes dele ser executado, as entradas do mesmo já devem estar disponíveis para o objeto instanciado da classe que estende `Algorithm`. Uma vez executado o algoritmo, o resultado do mesmo também fica armazenado no mesmo objeto, aguardando que o método buildDisplayList() seja executado para criar uma saída para o usuário.

`void buildDisplayList()` - este método cria uma resposta do algoritmo para o usuário. A *display list* aqui criada é chamada na função de renderização da aplicação. Assim, de acordo com o resultado obtido pela função `execute`, esta função cria uma nova saída. Por exemplo, ela pode criar o desenho do casco $\mathcal{G}_{\mathcal{F}}$-convexo dos pontos, ou um círculo em volta de um ponto indicando que este é interno a um $\mathcal{G}_{\mathcal{F}}$-polígono.

Capítulo 5

Conclusão

Neste trabalho foi feito um estudo da geometria $\mathcal{G}_\mathcal{F}$ sobre diferentes aspectos. Primeiramente, fizemos a caracterização de algumas famílias de curvas que satisfazem as condições básicas para se formar uma geometria $\mathcal{G}_\mathcal{F}$, como as famílias de curvas $\mathcal{F}$ formadas a partir de um conjunto gerador e as famílias geradas a partir de distorções do plano euclidiano.

Estudamos também predicados e primitivas recorrentes em problemas sobre essa geometria. Neste estudo, pudemos observar que predicados como o de orientação de três pontos, interseção entre $\mathcal{G}_\mathcal{F}$-reta e $\mathcal{G}_\mathcal{F}$-segmento e interseção entre dois $\mathcal{G}_\mathcal{F}$-segmentos são muito úteis para adaptação de diversos algoritmos da geometria euclidiana para a geometria $\mathcal{G}_\mathcal{F}$. Outros predicados precisam de uma adaptação para que possam ser resolvidos na $\mathcal{G}_\mathcal{F}$, como os predicados de precedência, que necessitam de um referencial. Porém, alguns predicados e primitivas não puderam ser utilizados do mesmo modo que na geometria euclidiana, como a distância entre duas retas paralelas. Outro predicado que não é diretamente adaptável a partir da geometria euclidiana é o de pertinência de um ponto em um $\mathcal{G}_\mathcal{F}$-círculo formado por outros três pontos.

O estudo dos predicados e primitivas facilitou a análise de diversos problemas geométricos na $\mathcal{G}_\mathcal{F}$. Na maioria dos problemas analisados, foi possível propor algoritmos de complexidade equivalente à dos já conhecidos para o caso euclidiano. A tabela 5.1 mostra os resultados obtidos para os problemas estudados.

Ademais, foi desenvolvido o visualizador `GFViwer` que facilitou o entendimento e a nossa percepção de conceitos como o de $\mathcal{G}_\mathcal{F}$-convexidade e regiões de interesse. O visualizador também permitiu observarmos na prática a resolução das primitivas e predicados analisados na seção 2.4 e a execução de algoritmos na geometria $\mathcal{G}_\mathcal{F}$.

Problema	**Seção**	**Geometria Euclidiana**	**Geometria $\mathcal{G}_\mathcal{F}$**	**Quota inferior**
Localização de ponto em polígono simples	3.1.1	$O(n)$	$O(n)$	$\Omega(n)$
Localização de ponto em polígono convexo	3.1.2	$O(\log n)$	$O(\log n)$	$\Omega(\log n)$
Localização de ponto em polígono estrelado	3.1.3	$O(\log n)$	$O(\log n)$	$\Omega(\log n)$
Interseção de polígonos convexos	3.2	$O(m+n)$	$O(m+n)$	$\Omega(n+m)$
Teste de simplicidade	3.3	$O(n \log n)$	$O(n \log n)$	$\Omega(n \log n)$
Separabilidade de conjuntos	3.4.1	$O(m+n)$	$O((m+n)\log(m+n))$	$\Omega(m+n)$
Separabilidade de polígonos	3.4.2	$O(m+n)$	$O(m+n)$	$\Omega(m+n)$
Casco convexo (incremental)	3.5	$O(n^2)$	$O(n^2)$	$\Omega(n \log n)$
Casco convexo de polígono simples	3.6	$O(n)$	$O(n)$	$\Omega(n)$
Núcleo de polígono	3.7	$O(n)$	$O(n)$	$\Omega(n)$

Tabela 5.1: Complexidade dos algoritmos para os problemas estudados nesta dissertação. As quotas inferiores apresentadas são relativas aos problemas na geometria euclidiana.

5.1 Trabalhos Futuros

Esta dissertação abordou um pequeno conjunto de problemas na geometria $\mathcal{G}_\mathcal{F}$ e, na maioria deles, foram propostos algoritmos de mesma complexidade assintótica que os algoritmos conhecidos no caso euclidiano. No caso do problema da separabilidade de conjuntos, visto na seção 3.4.1, a não transferência direta dos passos exigidos pelo algoritmo $O(n)$ de Megiddo [Meg83] para resolução de problemas de programação linear fez com que a eficiência do caso euclidiano não fosse atingida. Assim, um trabalho em aberto é o estudo do problema de programação não-linear, restrito a uma geometria de curvas, como a $\mathcal{G}_\mathcal{F}$.

Além disso, muitos outros problemas são interessantes sobre a geometria $\mathcal{G}_\mathcal{F}$, devendo ser estudados. Entre eles, podemos destacar os que envolvem a noção de distância, como o problema do diâmetro de um conjunto de pontos, maior círculo vazio ou menor círculo envolvente. Outra questão relevante que pode ser estudada na geometria $\mathcal{G}_\mathcal{F}$ é a idéia de dualidade entre ponto e reta utilizada para resolver diversos problemas na geometria euclidiana, como os problemas: colinearidade de pontos, ordenação circular de pontos,

triângulo de menor área ou vizinho mais próximo para retas.

Através do estudo desses problemas pode-se estender a lista de predicados geométricos e a forma de resolvê-los sobre a $\mathcal{G}_{\mathcal{F}}$.

Por fim, o visualizador `GFViewer` pode ter suas funcionalidades ampliadas com a adição novos algoritmos, novas famílias de curvas ou ainda ter sua interface melhorada com o acréscimo de novos objetos geométricos, como $\mathcal{G}_{\mathcal{F}}$-raios. A possibilidade de se colocar rótulos para identificação de objetos, ou ainda um ambiente para execução passo a passo dos algoritmos são também funcionalidades interessantes para o uso do visualizador no ensino de geometria computacional.

Apêndice A

Sobre as Análises de Complexidade

Como definido no capítulo 2, tratamos nessa dissertação de uma geometria, denotada $\mathcal{G}_{\mathcal{F}}$, baseada em uma família de curvas $\mathcal{F}$ escolhida de forma que satisfaça as seguintes propriedades:

1. Toda curva $C \in \mathcal{F}$ é uma curva de Jordan que passa pelo pólo norte da esfera de Riemann.

2. Dados dois pontos quaisquer no plano, existe apenas uma curva em $\mathcal{F}$ que passa por eles.

3. Existe um limitante superior para o número de partes diferenciáveis das curvas de $\mathcal{F}$.

Podemos observar que essas três propriedades nada dizem sobre a complexidade das funções que descrevem as curvas de $\mathcal{F}$. Dependendo dessa complexidade, podemos ter dificuldades em solucionar primitivas resolvidas trivialmente na geometria euclidiana, como o cálculo da interseção entre retas e o cálculo da distância entre pontos.

Assim, a análise de complexidade de algoritmos que usem o cálculo da interseção entre retas ou o cálculo da distância entre pontos será afetada pela dificuldade da resolução dessas primitivas.

Além das funções que descrevem a família de curvas $\mathcal{F}$, um outro fator que influi na análise de complexidade de algoritmos para a geometria $\mathcal{G}_{\mathcal{F}}$ é o número de partes diferenciáveis da família de curvas escolhida, como as famílias de curvas construídas conforme indicado na seção 2.2.

A seguir, será feita uma breve análise de como cada um dos três pontos citados influencia a análise de complexidade dos algoritmos na geometria $\mathcal{G}_{\mathcal{F}}$.

A.1 Interseção entre $\mathcal{G}_\mathcal{F}$-retas

Decorre trivialmente do ítem 2 que duas $\mathcal{G}_\mathcal{F}$-retas distintas podem ter no máximo um ponto de interseção. Não havendo interseção, dizemos que as retas são paralelas. A dificuldade em se determinar o ponto de interseção, ou saber que ele não existe, depende da família de curvas $\mathcal{F}$ escolhida. Se as funções que representam suas partes diferenciáveis são descritas por polinômios de até quarto grau, podemos determinar a interseção através da resolução de uma equação. Porém, para casos de funções não polinomiais, ou ainda polinômios de grau maior que quatro, onde é impossível determinar a solução algébrica por um número finito de adições, subtrações, multiplicações, divisões ou radiciações, pode ser necessário o uso de um método numérico para determinar o ponto de interseção.

Assim, sempre que um passo de um algoritmo na $\mathcal{G}_\mathcal{F}$ necessitar encontrar a interseção entre duas $\mathcal{G}_\mathcal{F}$-retas, na sua análise de complexidade devemos levar em conta uma constante dependente de um eventual método numérico utilizado para o cálculo da interseção, além de termos a exatidão da resposta dependente desse método.

A.2 Distância entre Pontos

A distância entre dois pontos é uma noção essencial para o desenvolvimento de algoritmos para problemas geométricos de proximidade.

Dados dois pontos p e q, seja $\widetilde{pq}$ o $\mathcal{G}_\mathcal{F}$-segmento que liga p a q e que não passa pelo ponto do infinito. A distância entre p e q é definida como o comprimento de $\widetilde{pq}$.

Como cada segmento diferenciável de curvas de $\mathcal{F}$ pode ser representado como uma curva parametrizada, achar a distâcia entre dois pontos envolve um cálculo de uma ou mais integrais de linha sobre os segmentos em questão. Tal cálculo, dependendo da complexidade das curvas de $\mathcal{F}$, pode necessitar um método numérico para a solução da integral. O uso de um método numérico faz com que a complexidade do cálculo de distância dependa da precisão desejada na solução. Isso acrescenta um novo fator constante à análise de complexidade de algoritmos que exijam o cálculo de distância, e faz com que a exatidão da resposta dependa do método.

A.3 Número de Partes Diferenciáveis de Curvas de $\mathcal{F}$

O número de pontos não diferenciáveis das curvas de uma família $\mathcal{F}$ tem influência direta no tempo necessário para achar tanto a interseção entre duas $\mathcal{G}_\mathcal{F}$-retas como também para calcular distâncias entre pontos.

No primeiro caso, dadas duas $\mathcal{G}_{\mathcal{F}}$-retas ℓ_1 e ℓ_2 com múltiplas partes diferenciáveis, a interseção entre elas pode ocorrer entre qualquer uma das partes de ℓ_1 e qualquer uma das partes de ℓ_2. Se conhecermos apenas as funções parametrizadas que representam cada parte diferenciável das $\mathcal{G}_{\mathcal{F}}$-retas, saber se duas partes se interceptam implica na resolução de um sistema de equações. Portanto, saber se alguma das partes de ℓ_1 intercepta ℓ_2 toma um tempo que depende do número de partes diferenciáveis de ℓ_1. Assim, a complexidade de um método para o cálculo de interseções entre curvas de $\mathcal{F}$ depende do número de partes diferenciáveis das mesmas.

Já no cálculo de distância entre dois pontos é fácil perceber que, caso a $\mathcal{G}_{\mathcal{F}}$-reta que passa por eles possua mais de uma parte diferenciável no $\mathcal{G}_{\mathcal{F}}$-segmento que os liga, será necessário calcular a integral de linha de cada parte separadamente. Assim, o tempo para o cálculo da distância entre dois pontos depende diretamente do número de partes diferenciáveis no $\mathcal{G}_{\mathcal{F}}$-segmento que os liga, e da dificuldade do cálculo da integral de linha de cada uma dessas partes.

Apêndice B

Distância entre Pontos

O cálculo de distância sobre a geometria $\mathcal{G}_{\mathcal{F}}$ depende do tipo de curvas que formam aquela geometria. Para isso, precisamos calcular a integral de linha sobre a curva que passa pelos dois pontos, entre os quais queremos saber a distância.

A fim de ilustrar a dificuldade em se implementar essa primitiva, este apêndice mostra o cálculo para o acharmos a distância entre pontos na família de curvas "x^2" e "$-x^2$", que são as curvas descritas pelas funções $f(x) = ax^2 + bx + c$ para um valor fixo de $a \neq 0$, juntamente com as retas verticais. A distância é dada pelo comprimento do $\mathcal{G}_{\mathcal{F}}$-segmento de reta $L(t_0, t_1)$ que os une. Obviamente, caso os pontos estejam sobre uma mesma vertical, a distância calculada é a diferença das ordenadas.

Podemos parametrizar a função que define as retas não verticais da geometria, com:

$$\begin{aligned} x(t) &= t \\ y(t) &= at^2 + bt + c \\ f(t) &= (x(t), y(t)) \end{aligned}$$

O comprimento de arco $L(t_0, t_1)$ sobre uma curva parametrizada $f : I \to \mathbb{R}^2$ regular, a partir de uma origem $t_0 \in I$ é dado por:

$$L(t_0, t_1) = \int_{t_0}^{t_1} |f'(t)|\, dt, t_1 \in I$$

onde $|f'(t)| = \sqrt{(x'(t))^2 + (y'(t))^2}$. A norma da derivada de $f(t)$ é dada por:

$$|f'(t)| = \sqrt{1^2 + (2at + b)^2} = \sqrt{4a^2t^2 + 4abt + b^2 + 1}$$

E o comprimento do arco é:

$$L(t_0, t_1) = \int_{t_0}^{t_1} \sqrt{4a^2t^2 + 4abt + b^2 + 1}\, dt$$

Substituindo $A = 2a$, $B = 4ab$ e $C = b^2 + 1$ na expressão acima chegamos a:

$$L(t_0, t_1) = \int_{t_0}^{t_1} \sqrt{A^2t^2 + Bt + C}\, dt \tag{B.1}$$

Completando os quadrados:

$$\begin{aligned} A^2t^2 + Bt + C &= A^2\left(t^2 + \left(\frac{Bt}{A^2}\right)\right) + C \\ &= A^2\left(t + \frac{B}{2A^2}\right)^2 + \left(C - \frac{B^2}{4A^2}\right) \end{aligned}$$

Tomando $u = t + \dfrac{B}{2A^2}$:

$$A^2t^2 + Bt + C = A^2u^2 + \left(C - \frac{B^2}{4A^2}\right)$$

Mas:

$$C - \frac{B^2}{4A^2} = \left(b^2 + 1\right) - \frac{16a^2b^2}{16a^2} = 1$$

Assim, a equação B.1 se reduz a:

$$\begin{aligned} L(t_0, t_1) = \int_{t_0}^{t_1} \sqrt{A^2t^2 + Bt + C}\, dt &= \int_{u_0}^{u_1} \sqrt{A^2u^2 + 1}\, du \\ &= |A| \int_{u_0}^{u_1} \sqrt{u^2 + \frac{1}{A^2}}\, du \end{aligned}$$

onde $u_0 = t_0 + \dfrac{B}{2A^2}$ e $u_1 = t_1 + \dfrac{B}{2A^2}$. Fazendo $k = \dfrac{1}{A^2}$:

$$L(t_0, t_1) = |A| \int_{u_0}^{u_1} \sqrt{u^2 + k}\, du$$

Resolvendo a integral:

$$L(t_0, t_1) = |A| \int_{u_0}^{u_1} \sqrt{u^2 + k}\, du = |A| \left[\left(\frac{u}{2}\sqrt{u^2 + k} + \frac{k}{2}\ln\left|u + \sqrt{u^2 + k}\right|\right)\right]_{u_0}^{u_1}$$

$$= |A| \left(\left(\frac{u_1}{2}\sqrt{u_1^2 + k} + \frac{k}{2}\ln\left|u_1 + \sqrt{u_1^2 + k}\right|\right) - \left(\frac{u_0}{2}\sqrt{u_0^2 + k} + \frac{k}{2}\ln\left|u_0 + \sqrt{u_0^2 + k}\right|\right)\right)$$

Esta foi a fórmula utilizada no programa `GFViewer` para o cálculo de distância das curvas "x^2" e "$-x^2$". A família "x" é a própria geometria euclidiana, onde a distância entre dois pontos é dada por:

$$d(p_1, p_2) = \sqrt{(p_1.x - p_2.x)^2 + (p_1.y - p_2.y)^2}$$

Já a família "$1/x$" é formada por algumas retas euclidianas, cujas distâncias são calculadas como acima, e curvas que têm equação:

$$f(x) = \frac{a}{x - b} + c$$

Onde a é constante para todas as curvas. Parametrizando a função:

$$\begin{aligned} x(t) &= t \\ y(t) &= \tfrac{a}{t-b} + c \\ f(t) &= (x(t), y(t)) \end{aligned}$$

Então, a norma da derivada da função é dada por:

$$|f'(t)| = \sqrt{1^2 + \left(\frac{-a}{(t-b)^2}\right)^2} = \sqrt{1 + \frac{a^2}{(t-b)^4}}$$

Portanto, o comprimento procurado é:

$$L(t_0, t_1) = \int_{t_0}^{t_1} |f'(t)|\, dt, t_1 \in I = \int_{t_0}^{t_1} \sqrt{1 + \frac{a^2}{(t-b)^4}}\, dt$$

Como esta última integral não possui anti-derivada simples, foi implementada uma aproximação por segmentos de retas para a distância entre dois pontos nesta família.

Apêndice C

Tabela de Símbolos

a, b, c	números reais
$\overline{e}$	aresta
$\widetilde{e}$	$\mathcal{G}_{\mathcal{F}}$-aresta
i, j	índices
l	reta
ℓ	$\mathcal{G}_{\mathcal{F}}$-reta
k	auxiliar para índices, cardinalidade de conjuntos ou constante
m, n	cardinalidade de conjunto
p	ponto
$p.x$	abcissa do ponto p
$p.y$	ordenada do ponto p
$p.\mathrm{z}$	altura do ponto p
q	auxiliar para ponto
$\overline{pq}$	segmento de reta euclidiana entre p e q
$\widetilde{pq}$	$\mathcal{G}_{\mathcal{F}}$-segmento de reta entre p e q
$\overline{r}$	semi-reta
$\widetilde{r}$	$\mathcal{G}_{\mathcal{F}}$-semi-reta
$\overline{s}$	segmento de reta
$\widetilde{s}$	$\mathcal{G}_{\mathcal{F}}$-segmento de reta
v	vértice
u, w	auxiliares para vértice
$\vec{v}$	vetor
x	coordenada no eixo das abcissas
y	coordenada no eixo das ordenadas
α, β	ângulos

ρ	raio de uma circunferência
C	curva
$H(\vec{v})$	meio plano fechado à esquerda de $\vec{v}$
$K(\mathcal{P})$	núcleo do polígono $\mathcal{P}$
L	conjunto de retas
P	conjunto de pontos
S	conjunto de curvas
$\mathbb{R}$	conjunto dos reais
$\mathbb{R}^2$	plano
$\mathbb{R}^3$	espaço
$\mathcal{S}$	pilha (estrutura de dados)
$\mathcal{Q}$	fila (estrutura de dados)
$\mathcal{E}$	fila de prioridade (estrutura de dados)
$\mathcal{L}$	dicionario (estrutura de dados)
$\mathcal{F}$	família de curvas
$\mathcal{G}$	geometria
$\mathcal{H}$	$\mathcal{G}_\mathcal{F}$-semi-plano
$\mathcal{K}$, $\mathcal{P}$, $\mathcal{Q}$	polígonos
$\mathcal{P}_\mathcal{F}$, $\mathcal{Q}_\mathcal{F}$	$\mathcal{G}_\mathcal{F}$-polígonos
$\mathcal{R}^2$	esfera de Riemman
$B(p,\rho)$	bola euclidiana com centro em p e raio $\rho > 0$
$\partial B(p,\rho)$	fronteira de $B(p,\rho)$
$d(p_1,p_2)$	distância euclidiana entre p_1 e p_2
$B_\mathcal{F}(p,\rho)$	bola da $\mathcal{G}_\mathcal{F}$ com centro em p e raio $\rho > 0$
$\partial B_\mathcal{F}(p,\rho)$	fronteira de $B_\mathcal{F}(p,\rho)$
$d_\mathcal{F}(p_1,p_2)$	distância entre p_1 e p_2 na geometria $\mathcal{G}_\mathcal{F}$, dada pelo comprimento de arco do $\mathcal{G}_\mathcal{F}$-segmento $\widetilde{p_1p_2}$
$\Delta(p_1,p_2,p_3)$	orientação do trio de pontos (p_1,p_2,p_3)

Referências Bibliográficas

[Aur87] F. Aurenhammer. Power Diagrams: Properties, Algorithms and Applications. *SIAM J. Comput.*, 16:78–96, 1987.

[BO79] J. L. Bentley and T. A. Ottmann. Algorithms for Reporting and Counting Geometric Intersections. *IEEE Trans. Comput.*, C-28(9):643–647, September 1979.

[Dra52] M. Drandell. Generalized Convex Sets in the Plane. *Duke Mathematical Journal*, 19(3):537–547, 1952.

[dRG96] P. J. de Rezende and C. N. Gon. Um Ambiente Distribuído de Visualização Com Suporte para Geometria Projetiva Orientada. In *IX SIBGRAPI: IX Brazilian Symposium on Computer Graphics and Image Processing*, pages 71–78, 1996.

[dRLW89] P. J. de Rezende, D. T. Lee, and Y. F. Wu. Rectilinear Shortest Paths in the Presence of Rectangular Barriers. *Discrete Comput. Geom.*, 4:41–53, 1989.

[dRS94] P. J. de Rezende and J. Stolfi. *Fundamentos de Geometria Computacional.* IX Escola de Computação, Recife (Brazil), July 1994.

[dRW99] P. J. de Rezende and R. B. Westrupp. An Optimal Algorithm to Construct All Voronoi Diagrams for k Nearest Neighbor Search in T^2. In *XII SIBGRAPI: XII Brazilian Symposium on Computer Graphics and Image Processing*, pages 7–15, 1999.

[Gue98] G. B. Guerra-Filho. Problemas de Proximidade e de Caminhos Mínimos Em Superfícies Poliédricas. Master's thesis, Universidade Estadual de Campinas, 1998.

[Har00] M. M. Harada. *Convexidade e Proximidade em Geometrias não Euclidianas.* PhD thesis, Universidade Estadual de Campinas, November 2000.

[Lee80] D. T. Lee. Two-dimensional Voronoi Diagrams in the L_p-metric. *J. ACM*, 27(4):604–618, 1980.

[Lee83] D. T. Lee. On Finding the Convex Hull of a Simple Polygon. *International Journal of Computer and Information Science*, 12:87–98, 1983.

[LP79] D. T. Lee and F. P. Preparata. An Optimal Algorithm for Finding the Kernel of a Polygon. *J. ACM*, 26(3):415–421, July 1979.

[Meg83] N. Megiddo. Linear-time Algorithms for Linear Programming in R^3 and Related Problems. *SIAM J. Comput.*, 12:759–776, 1983.

[Mit87] J. S. B. Mitchell. Shortest Rectilinear Paths Among Obstacles. In *Proc. 1st Internat. Conf. Indust. Applied Math.*, pages 39–84, 1987.

[MP91] J. S. B. Mitchell and C. H. Papadimitriou. The Weighted Region Problem: Finding Shortest Paths Through a Weighted Planar Subdivision. *J. ACM*, 38:18–73, 1991.

[OCON82] J. O'Rourke, C.-B. Chien, T. Olson, and D. Naddor. A New Linear Algorithm for Intersecting Convex Polygons. *Comput. Graph. Image Processing*, 19:384–391, 1982.

[OdRS05] A. G. Oliveira, P. J. de Rezende, and F. P. Selmi-Dei. An Extension of CGAL to the Oriented Projective Plane T^2 and its Dynamic Visualization System. In *SCG '05: Proceedings of the twenty-first annual symposium on Computational geometry*, pages 380–381, New York, NY, USA, 2005. ACM Press.

[O'R94] J. O'Rourke. *Computational Geometry in C.* Cambridge University Press, 1994.

[PdR00] G. Pinto and P. J. de Rezende. Additively Weighted Voronoi Diagram in the Oriented Projective Plane. In *Proc. 12th Canadian Conference on Computational Geometry*, pages 119–126, 2000.

[Pei49] M. M. Peixoto. On Convexity. In *Anais da Academia Brasileira de Ciências*, volume 21, pages 291–302, 1949.

[PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry: an Introduction.* Springer-Verlag Inc., Berlin, 1985.

[Raw87] G. J. E. Rawlins. *Explorations in Restricted Orientation Geometry*. Ph.D. thesis, Univ. Waterloo, Waterloo, ON, 1987.

[RW91] G. J. E. Rawlins and D. Wood. Restricted-Oriented Convex Sets. *Inform. Sci.*, 54:263–281, 1991.

[SH75] M. I. Shamos and D. Hoey. Closest-point Problems. In *Proceedings 16h Annual IEEE Symposium on Foundations of Computer Science*, pages 151–162, 1975.

[SH76] M. I. Shamos and D. Hoey. Geometric Intersection Problems. In *Proceedings 17h Annual IEEE Symposium on Foundations of Computer Science*, pages 208–215, 1976.

[Sha78] M. I. Shamos. *Computational Geometry*. PhD thesis, Yale University, 1978.

[Sto91] J. Stolfi. *Oriented Projective Geometry: A Framework for Geometric Computations*. Academic Press, New York, NY, 1991.

[SW70] J. Stoer and C. Witzgall. *Convexity and Optimization in Finite Dimensions I*. Springer-Verlag, Berlin, 1970.

[WWSW87] Y. F. Wu, P. Widmayer, M. D. F. Schlag, and C. K. Wong. Rectilinear Shortest Paths and Minimum Spanning Trees in the Presence of Rectilinear Obstacles. *IEEE Trans. Comput.*, C-36(3), March 1987.