Reconstrução automática de objetos fragmentados

Helena Cristina da Gama Leitão

Tese de Doutorado

Reconstrução automática de objetos fragmentados

Este exemplar corresponde à redação final da Tese devidamente corrigida e defendida por Helena Cristina da Gama Leitão e aprovada pela Banca Examinadora.

Campinas, 21 de outubro de 1999.

Dr. Jorge Stolfi (Orientador)

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

Instituto de Computação
Universidade Estadual de Campinas

Reconstrução automática de objetos fragmentados

Helena Cristina da Gama Leitão

outubro de 1999

Banca Examinadora:

- Dr. Jorge Stolfi (Orientador)
- Dr. Sóstenes Lins
  DMA - Universidade Federal de Pernambuco
- Dr. Antônio Fernandes Oliveira
  COPPE – Universidade Federal do Rio de Janeiro
- Dr. Alexandre Xavier Falcão
  IC - Universidade Estadual de Campinas
- Dr. João Meidanis
  IC - Universidade Estadual de Campinas
- Dra. Aura Conci(Suplente)
  IC - Universidade Federal Fluminense
- Dr. João Carlos Setubal(Suplente)
  IC - Universidade Estadual de Campinas

Dedicatória

em memória de minha avó Diva Soley da Gama

Agradecimentos

De forma especial, a minha super mãe e a minha irmã Isabel Cristina.

Ao grande amigo e orientador prof. Jorge Stolfi, pela tese e por tudo mais...

Aos amigos, Fábio Tanaka, Edgard Leitão, Maria Vitória, Simara Matsubara, Anamaria Gomide, Maria Emília Walter, Marcus Vinícios Andrade, Flávia Andrade, Nuccio Zuquello, Daniel da Silva Andrade, Luiz Arturo Lozada, Guilherme Albuquerque, Alexandre Oliva, Islene Garcia, Roland Scialom, Luiz Satoru Ochi, Lúcia Drummond, Maria Cristina Boeres, Vinod Rebello, e todos os outros que foram esquecidos neste parágrafo...

A todos os colegas das salas de estudos pelas quais passei.

A todos os professores e funcionários do Instituto de Computação da Unicamp.

Ao prof. Ricardo Anido por me apresentar ao prof. Stolfi.

À Islene pelo auxílio na montagem manual dos ladrilhos de cerâmica usados no capítulo 11, pela ajuda na conversão de alguns gráficos apresentados na tese e na preparação das transparências da apresentação.

À Selma Leitão por ajudar a descobrir, baseando-se apenas nos contornos, os pares de fragmentos adjacentes que foram usados para verificação dos resultados.

Ao meu fusca ZD-8526, por tantas idas e vindas, e a todos aqueles que deram um empurrãozinho quando ele hesitou em ir (ou em vir).

A todos (e foram tantos...de freiras a um motorista de general) que participaram destes anos de Campinas.

À Universidade Federal Fluminense, em particular ao Instituto de Computação e à sua diretora profa. Míriam Aparecida Marques.

À CAPES/PICD por ter financiado meu trabalho.

Resumo

Esta tese aborda o seguinte problema: dados um ou mais objetos que tenham sido quebrados ou partidos em um grande número de fragmentos irregulares, achar os pares de fragmentos que eram adjacentes nos objetos originais.

Nossa abordagem é baseada na comparação das curvaturas codificadas dos contornos dos fragmentos, usando uma variação do algoritmo de programação dinâmica para casamento de seqüências.

Objetivando reduzir o custo assintótico do casamento de um grande número de contornos de resolução alta, usamos uma técnica de casamento em múltiplas escalas. Depois de filtrar e reamostrar os contornos dos fragmentos em diferentes escalas de detalhes, procuramos casamentos iniciais na escala mais grosseira possível. Então, repetidamente, selecionamos os pares mais promissores, e refinamos os mesmos numa escala cada vez mais fina de detalhes. No final, obtemos um conjunto pequeno de pares de fragmentos que são os que mais parecem ser adjacentes nos objetos originais.

Abstract

This thesis addresses the following problem: given one or more unknown objects that have been broken or torn into a large number of irregular fragments, find the pairs of segments that were adjacent in the original objects.

Our approach is based on comparison of the curvature-encoded fragment outlines with a variation of the dynamic programming sequence-matching algorithm.

In order to reduce the asymptotic cost of matching a large number of high-resolution outlines, we use a multiple scale matching technique. After filtering and resampling the fragment outlines at many different scales of detail, we look for initial matchings at the coarsest possible scale. We then repeatedly select the most promising pairs, and refine them at the next finer scale of detail. In the end, we are left with a small set of fragment pairs that are most likely to be adjacent in the original objects.

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

O tema deste trabalho é a reconstrução auxiliada por computador de objetos desconhecidos que foram quebrados, ou rasgados, em um grande número de fragmentos irregulares, tais como os da figura 1.1.

Figura 1.1: Alguns fragmentos de cerâmica de um sítio pré-histórico na Grécia [11].

A reconstrução pode ser dividida conceitualmente em duas etapas: a *identificação de fragmentos adjacentes* e a *montagem* propriamente dita. Neste trabalho, estamos interessados no primeiro passo, em que são identificados os pares de fragmentos que eram adjacentes nos objetos originais. Quando os pares adjacentes estão misturados com centenas ou milhares de outros fragmentos, pode ser impossível encontrá-los manualmente.

Nossa contribuição é um conjunto de técnicas matemáticas e algoritmos que podem ser usados para localizar tais pares a partir dos contornos digitalizados dos fragmentos. Estas técnicas foram implementadas por nós, e sua validade foi comprovada por experimentos.

Uma vez identificado o conjunto aproximado de pares de fragmentos adjacentes, a etapa seguinte seria a *montagem global*, onde o objetivo é reconstruir a topologia e a geometria dos objetos fraturados. Esta etapa não foi abordada neste trabalho visto que sua dificuldade e a qualidade do resultado dependerão muito da correção e completude da lista de pares adjacentes encontrados na primeira etapa. Além disso, a etapa de montagem global requer intervenção humana para introduzir informações que não são capturadas pelos contornos — tais como textura, cor e decoração dos fragmentos, localização etc.

1.1 Motivação

A montagem de fragmentos é um problema bastante comum e importante em arqueologia, como mostra o texto abaixo:

> *Pottery forms one of the most common and abundant types of artifacts found at archaeological sites. As such, pottery provides important information about chronology, technology, trade, and art. Although most handbooks illustrate whole or nearly complete vessels, the vast majority of ceramic remains from any excavation consists of thousands of small, broken pieces which need to be cleaned, sorted, counted, weighed, and analyzed.* [11]

Além da cerâmica, as técnicas que apresentaremos podem ser aplicadas a outros tipos de materiais arqueológicos, como por exemplo: objetos de vidro [15], ferramentas de pedra [11], tabuinhas cuneiformes [23], manuscritos antigos [3], pinturas murais [14], estatuetas e decorações esculpidas em pedra, etc.

Esperamos que nossas técnicas também possam ser aplicadas aos problemas de reconstrução de objetos de outras áreas, tais como paleontologia (fósseis), cirurgia e medicina legal (reconstrução de ossos), análise de falhas, etc.

1.2 Trabalhos relacionados

Até onde temos conhecimento, técnicas computacionais complexas ainda não são usadas na reconstrução de fragmentos arqueológicos. O trabalho de classificação é, quase sempre, artesanal, exigindo muita paciência e muito tempo para ser realizado. Os computadores são usados, quando muito, em tarefas de classificação, indexação, apresentação e digitalização das imagens dos fragmentos [3, 14, 35], sendo indexados e recuperados baseando-se somente nas descrições textuais fornecidas pelo usuário.

Há vários artigos sobre técnicas de realce de imagens aplicadas a materiais arqueológicos [35, 14]. Entretanto, poucos autores têm considerado o uso da visão por computador e técnicas para extração e indexação automáticas de informações, a partir de imagens digitais, visando auxiliar a solução de problemas na área de arqueologia.

O problema específico de identificar fragmentos de cerâmica adjacentes foi recentemente abordado por Üçoluk e Toroslu [34]. No algoritmo apresentado por eles, os contornos são considerados apenas em uma escala de resolução fixa, e portanto, é esperado um alto custo assintótico (veja a seção 1.6.3). Nenhum experimento prático é relatado no artigo.

O problema da montagem de fragmentos é similar ao problema da *montagem automática de quebra-cabeças*, que já foi abordado antes, principalmente como uma tarefa inteligente em robótica e visão por computador [9, 13, 8, 31]. Em particular, H.Wolfson e G.D.Burdea desenvolveram um programa que descobre as peças adjacentes em um jogo de quebra-cabeça típico [37]. Essas informações são usadas por um braço de robô que monta fisicamente o quebra-cabeça. Entretanto, as técnicas utilizadas neste trabalho se aproveitam de características particulares do formato das peças de um quebra-cabeça típico (bordas suaves e cantos bem definidos) que não são encontradas em artefatos arqueológicos.

Além disso, o problema pode ser visto como um caso especial do casamento de contornos aproximados, uma área de extensa bibliografia [18, 26, 17, 5, 29, 40, 21, 33]. Em especial, existem muitos artigos em visão computacional que versam sobre o *reconhecimento de objetos a partir de suas linhas de contorno*, com aplicações principalmente nas áreas industrial e militar. Alguns resultados obtidos nestas áreas podem ser aplicados também a fragmentos irregulares. Todavia, a maioria destes trabalhos considera que as linhas de contorno são testadas contra um número relativamente pequeno de *gabaritos*, e o que eles desejam é diminuir o custo de comparação entre as linhas de contorno do objeto e o gabarito. Na aplicação que nos interessa os gabaritos podem ser da ordem de milhares, pois são os próprios fragmentos. Nós estamos portanto interessados em técnicas que possam ser usadas para eliminar grandes conjunto de gabaritos a custo relativamente baixo.

1.3 Abordagem

Nossa abordagem é baseada na informação extraída das linhas de contornos dos fragmentos. O modelo idealizado pode ser resumido como a seguir. Nós supomos que os objetos fragmentados têm uma superfície bem definida e relativamente plana. Esta superfície é dividida em duas ou mais partes, os *fragmentos ideais*. Veja a figura 1.2(a). Os fragmentos são separados por *linhas de fratura*, que são curvas irregulares de largura zero. Dois fragmentos dão ditos *adjacentes* se eles compartilham uma mesma linha de fratura. As fraturas também dividem o contorno da superfície original do objeto em uma ou mais *linhas de borda.*

As linhas de fratura podem ser vistas como um grafo G desenhado sobre a superfície do objeto. O ponto onde três ou mais linhas (de fratura ou de borda) se encontram é chamado *canto ideal.* A concatenação das linhas de fratura na fronteira de um fragmento ideal constitui seu *contorno ideal.*

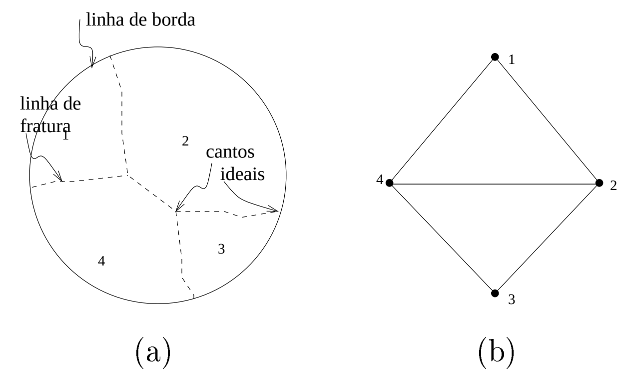


Figura 1.2: Rede de fratura ideal(a) e o grafo de adjacência (b).

Neste trabalho, supomos que cada fragmento tem a topologia de um disco, isto é, seu contorno consiste de uma única curva fechada. Nos casos onde isto não é verdade (por exemplo, no caso da alça de um jarro ou de um vaso cilíndrico quebrado em ambos os extremos), nós podemos tratar cada curva fechada como sendo a fronteira de um fragmento separado. O mesmo acontece para objetos com duas faces suaves de características similares (por exemplo, ladrilhos de cerâmica): podemos considerar as duas faces de cada fragmento como sendo dois fragmentos diferentes. A conexão física entre estes dois contornos só precisa ser feita na etapa de montagem global do objeto (que não nos interessa aqui).

O dual topológico da rede de fratura é chamado grafo de adjacências G^*. Neste grafo cada fragmento u é representado por um vértice u^*; para cada linha de fratura e, que separa os fragmentos u e v, existe uma aresta e^* em G^* conectando u^* e v^*. Veja a figura 1.2(b). Observe que G é um objeto geométrico e topológico enquanto G^* é somente um objeto topológico.

1.4 Os dados reais

As linhas de fratura e os contornos ideais são meras abstrações. O que obtemos a partir dos fragmentos recuperados são os *contornos observados*, que diferem dos contornos ideais devido a rebarbas (comuns em fragmentos de papel), manchas, perda de migalhas de material nas regiões de fratura, ou outras razões. Por causa destas perturbações, uma única linha de fratura que separa dois fragmentos adjacentes ideais dá origem a duas seções ligeiramente diferentes dos contornos observados correspondentes. Esta diferença é onde reside a principal dificuldade do problema da identificação de fragmentos adjacentes.

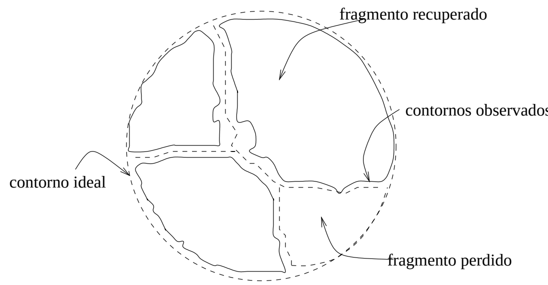


Figura 1.3: Contornos observados.

Chamaremos de segmento um trecho de contorno estritamente menor que o contorno completo. Um par de *segmentos correspondentes* é um par de trechos maximais de dois contornos observados que correspondem à mesma linha de fratura. Veja a figura 1.4.

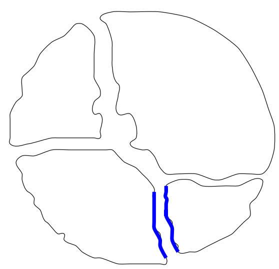


Figura 1.4: Segmentos correspondentes.

1.5 A geometria das linhas de fraturas

Para as aplicações que nos interessam, as linhas de fratura possuem um certo nível de auto-similaridade. Ou seja, se examinarmos um contorno ampliado verificaremos que suas irregularidades são visualmente similares às do contorno original. Veja a figura 1.5. Sendo assim, podemos dizer que as linhas de fratura são um exemplo clássico de *curvas fractais* [10].

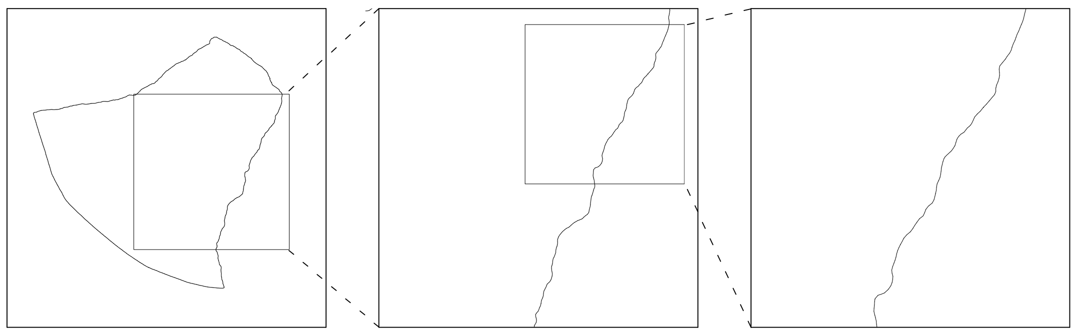


Figura 1.5: Carater fractal das linhas de fratura.

Este modelo é apropriado para fraturas em materiais rígidos e granulares como cerâmica, pedra, estuque, etc. Ele pode também ser apropriado para fraturas de papel muito antigo que tende a se esfarelar em vez de rasgar.

Como veremos no capítulo 5, são as irregularidades aleatórias da linha de fratura que permitem identificar segmentos correspondentes com confiança razoável.

Fraturas de vidro ou de objetos de cerâmica vitrificada são muito mais suaves, possuindo portanto menos irregularidades características. Por outro lado, são mais bem definidas e podemos adquirí-las e compará-las com grande precisão; por isso, esperamos que o modelo sirva também para este tipo de material.

1.6 Características físicas dos contornos obtidos

Em aplicações típicas, podemos esperar que existam fragmentos que não foram recuperados, ou seja, provavelmente teremos *fragmentos perdidos*. Portanto, existe uma ambigüidade intrínseca neste modelo: a perda de uma pequena porção de material na borda de um fragmento pode ser atribuída a *ruído* (figura 1.6(a)), ou pode ser modelada como um fragmento ideal que foi perdido (figura 1.6(b)). Nossos algoritmos resolverão arbitrariamente esta ambiguidade, de um modo ou de outro, dependendo do tamanho da parte que estiver faltando.

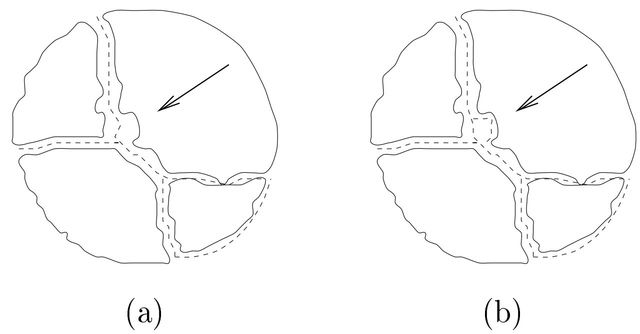


Figura 1.6: Ambigüidade do modelo de fraturas.

Outra ambiguidade do modelo é que não é possível definir com precisão o início e o término dos segmentos correspondentes. Além disso, não sabemos a posição dos cantos ideais de fratura, e de qualquer modo, a presença de ruído não nos possibilita definir uma correspondência ponto a ponto precisa entre os contornos ideais e observados. Novamente nossos algoritmos definirão arbitrariamente o início e o fim de segmentos correspondentes, baseando-se na distância entre dois contornos.

Uma importante propriedade das linhas de fratura físicas é que cada canto ideal no interior do objeto é geralmente incidente a somente três linhas de fratura, sendo que duas destas linhas normalmente fazem um ângulo de 180 graus. Veja a figura 1.7. Deste modo, a existência de um canto é aparente somente em dois dos três fragmentos incidentes. Por estas razões, não podemos esperar identificar os segmentos dos contornos adjacentes procurando por cantos.

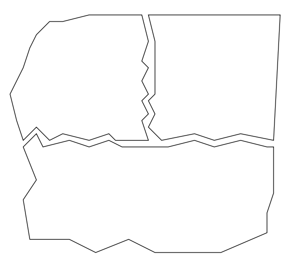


Figura 1.7: Geometria típica dos cantos ideais

Outra propriedade dos fragmentos reais é que o comprimento de uma linha de fratura dificilmente ultrapassa metade do comprimento do fragmento.

1.6.1 Objetivo da reconstrução

Com os conceitos acima, formulamos o problema da *reconstrução de objetos fragmentados* da seguinte maneira: Dada uma coleção $\mathcal{F}$ de contornos observados, queremos determinar a maior parte possível do grafo de adjacências G^*.

1.6.2 Identificação dos contornos

A primeira questão que devemos responder é se o problema pode de fato ser resolvido. É realmente possível identificar fragmentos adjacentes, apenas com base nos seus contornos observados (que contém ruídos)? A resposta é sim: por experimentação com fragmentos reais, podemos verificar que a natureza irregular das linhas de fratura permite que pares de fragmentos adjacentes sejam identificados com boa confiabilidade, por comparação de suas linhas de contorno em escala sub-milimétrica. Veja a figura 1.8.

(a) (b)

Figura 1.8: (a) e (b) são trechos dos contornos de dois fragmentos adjacentes.

Nossa estratégia para resolver o problema da reconstrução explora estas irregularidades dos contornos. Para isto, é preciso adquirir e trabalhar com os contornos dos fragmentos em escala sub-milimétrica.

Se os trechos forem suficientemente compridos, e a diferença de forma for suficientemente pequena, podemos descartar a hipótese de que essa semelhança seja devida a mera coincidência. No capítulo 5, analisaremos quantitativamente a relação entre precisão de medida, magnitude do ruído e probabilidade de identificar corretamente os fragmentos correspondentes de um dado tamanho.

1.6.3 Dificuldades Computacionais

Do ponto de vista computacional, a dificuldade principal deste problema é o grande número de pares de segmentos que precisam ser comparados.

Para identificar fragmentos adjacentes com confiança suficiente, nós precisaremos trabalhar com comparação de segmentos de contorno com milhares de pontos de amostras independentes. Além disso, para cada par de pontos precisamos testar todos os pares de segmentos possíveis. Em instâncias típicas do problema, onde temos milhares de fragmentos, esta busca exaustiva é muito custosa. Como veremos no capítulo 10, se nós temos N fragmentos, cada fragmento com n pontos de amostra, então uma comparação exaustiva precisará de $O(N^2 n^4)$ operações.

Objetivando reduzir assintoticamente este custo, usaremos duas idéias complementares: Primeiro, diminuiremos o número de pontos de amostra em cada fragmento, por filtragem e reamostragem deste contorno. Segundo, reduziremos o número de pares de segmentos a serem comparados em detalhe, através de técnicas de escalas múltiplas [2]. Na seção 10.1, mostraremos como estas técnicas [2, 22] nos permitem reduzir o custo para $O(N^2 + Mn^2)$, onde M é o número de pares de fragmentos adjacentes.

Usando *hashing* geométrico, como proposto por Haim J. Wolfson [38], seria possível reduzir o termo $O(N^2)$, possivelmente para $O(N \log N)$. Nós não implementamos esta otimização.

1.7 Etapas do processamento

Portanto, nosso processo de solução consiste das seguintes etapas:

- Aquisição e separação das imagens dos fragmentos;
- Segmentação e extração dos contornos;
- Filtragem dos contornos em várias escalas de resolução;
- Codificação dos contornos por cadeias de curvatura;
- Análise estatística dos contornos;
- Identificação de segmentos similares nas escalas mais grosseiras;
- Localização e refinamento dos pares similares nas escalas mais finas;
- Alinhamento geométrico ótimo dos segmentos semelhantes;
- Apresentação gráfica dos resultados.

Nos próximos capítulos, descreveremos cada estágio com mais detalhes.

Capítulo 2

Aquisição das imagens e obtenção dos contornos

A primeira etapa na solução do problema é aquisição das imagens dos fragmentos, que pode ser feita de diversas maneiras, dependendo da natureza dos mesmos. No caso de fragmentos de papel, por exemplo, pode-se utilizar diretamente um *scanner* de mesa. (Veja o exemplo da figura 2.1).

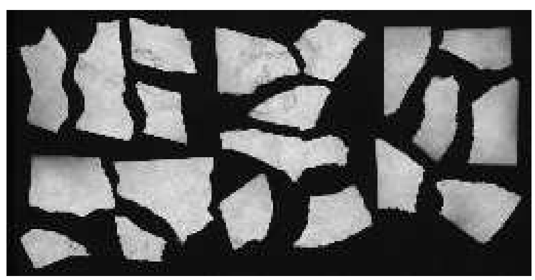


Figura 2.1: Imagem digitalizada contendo 20 fragmentos.

Para outros objetos, tais como ladrilhos, tabuinhas cuneiformes e pinturas murais, uma câmera (fotográfica ou de TV) talvez seja mais adequada. Em ambos os casos os contornos são essencialmente planos.

No caso de fragmentos de vasos de cerâmica ou estátuas, onde os contornos dos fragmentos são curvas tridimensionais, seria necessário adquirir duas ou mais imagens de cada fragmento, de ângulos diferentes, e usar técnicas de visão estereoscópica para extrair os contornos. Neste texto trataremos apenas do caso de imagens planas.

Uma vez obtidas as imagens dos fragmentos, devemos identificar os fragmentos separadamente e extrair os contornos dos mesmos. A extração de contornos é uma operação comum em reconhecimento de padrões. Nesta aplicação, entretanto, ela exige cuidados especiais, devido ao fato que a identificação de fragmentos vizinhos depende, em grande parte, de detalhes sub-milimétricos.

Na apresentação dos algoritmos de identificação e extração dos contornos dos fragmentos, utilizaremos a seguinte notação: n_x e n_y denotam as dimensões horizontais e verticais da imagem digitalizada, em pixels; $R = [0..\, n_x - 1] \times [0..\, n_y - 1]$ é o domínio da imagem. A posição genérica de um pixel p é, portanto, um par $(p_x, p_y) \in R$. O valor (intensidade) da imagem no pixel p é $v[p]$.

2.1 Identificação e separação dos fragmentos

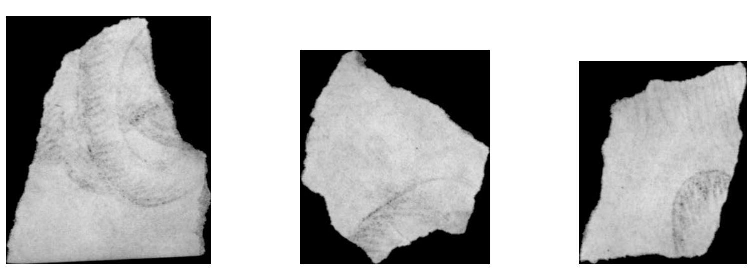
Por razões práticas, frequentemente são digitalizados dois ou mais fragmentos juntos. Veja a figura 2.1. Portanto, antes de extrair os contornos, devemos identificar e separar os fragmentos presentes em cada imagem.

Para identificar os fragmentos usamos um algoritmo baseado em simples limiarização (*thresholding*). O algoritmo supõe que a superfície dos fragmentos tem uma cor aproximadamente uniforme δ_{max}, e eles são digitalizados contra um fundo uniforme de cor $\delta_{min} < \delta_{max}$. Supomos portanto que existe um nível de cinza único δ_{sep}, tal que *pixels* com cor $v[p] \leq \delta_{sep}$ pertencem ao fundo e *pixels* com cor $v[p] > \delta_{sep}$ pertencem ao fragmento.

Uma vez localizado um *pixel* $v[p] > \delta_{sep}$, usamos uma variante do algoritmo de propagação (*flood-fill*), usado em muitos editores gráficos para colorir regiões de contornos arbitrários. Esse algoritmo identifica todos os *pixels* q com $v[q] > \delta_{sep}$ que são conectados ao pixel p, na topologia de vizinhança 4 [28].

Além do fragmento propriamente dito, extraímos uma borda de segurança ao redor do fragmento com um *pixel* de largura. Os pixels nesta borda tem por definição valores menores que δ_{sep}. Esta borda é importante para a determinação precisa do contorno do fragmento, como será explicado na seção 2.2. Supomos que os fragmentos estão suficientemente afastados, entre si e das bordas da imagem, para que estas bordas de segurança existam e sejam disjuntas.

A saída deste estágio é um conjunto de imagens, cada uma contendo um único fragmento com sua borda de segurança, contra um fundo uniforme de cor $\delta_{min} < \delta_{sep}$. Veja a figura 2.2.



(a) (b) (c)

Figura 2.2: Três dos fragmentos contidos na figura 2.1.

2.2 Extração dos contornos

Uma vez separados os fragmentos, extraímos o contorno bruto de cada um, sob a forma de uma curva fechada poligonal. Como na seção 2.1, nós supomos que a superfície dos fragmentos tem uma cor aproximadamente uniforme δ_{max}, e eles foram digitalizados contra um fundo uniforme de cor conhecida δ_{min}, como mostrado na figura 2.2. O contorno pode então ser extraído por um algoritmo simples que segue a curva de nível com uma intensidade intermediária δ_{med}, algum valor entre δ_{min} e δ_{max}.

Na verdade, é importante que o limiar δ_{med} seja exatamente $(\delta_{min} + \delta_{max})/2$, de modo a garantir que contornos de fragmentos adjacentes sejam congruentes. Este requisito pode ser deduzido das hipóteses de que as imagens digitalizadas são versões linearmente borradas de uma imagem ideal (de resolução infinita), e de que o objeto original tinha cor uniforme δ_{max} em ambos os lados da fratura.

Uma escolha incorreta do limiar δ_{med} mudará a curvatura das partes concâvas e convexas em direções opostas, dificultando o reconhecimento dos contornos correspondentes. A figura 2.3 ilustra este problema. Nos três exemplos, a cor do fundo δ_{min} é 10/255, e a cor do fragmento δ_{max} é 200/255.

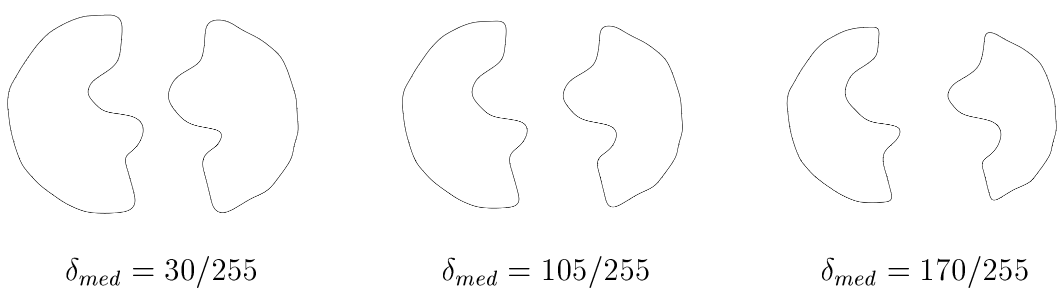


Figura 2.3: Efeito do limiar na forma dos contornos.

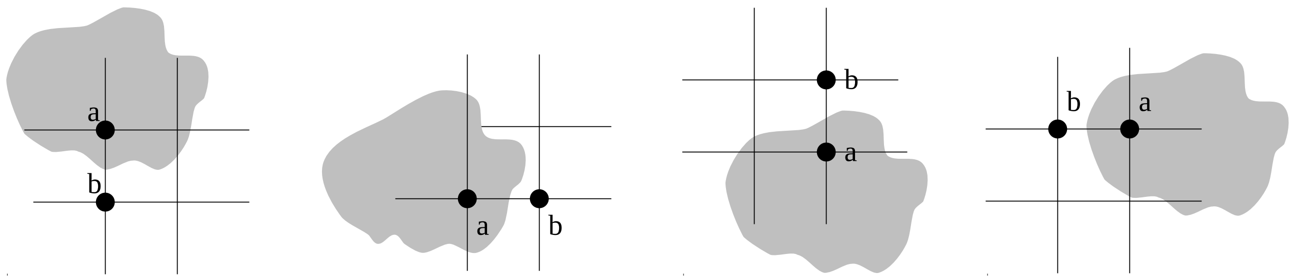
Observa-se nesta figura que os contornos extraídos são congruentes somente quando δ_{med} é próximo à média de δ_{min} e δ_{max}.

Para evitar inconsistências entre os procedimentos de separação e extração de contorno, é recomendável que o limiar δ_{sep} utilizado para a separação dos fragmentos seja igual ao limiar de extração do contorno δ_{med}.

2.2.1 Algoritmo de extração de contorno

Para extraírmos o contorno de um fragmento, primeiro localizamos um par de pontos adjacentes (a, b) na grade de pixels R, com a dentro do fragmento ($v[a] > \delta_{med}$), e b fora ($v[b] \leq \delta_{med}$). Então, seguimos a fronteira do fragmento no sentido anti-horário mantendo estas condições.

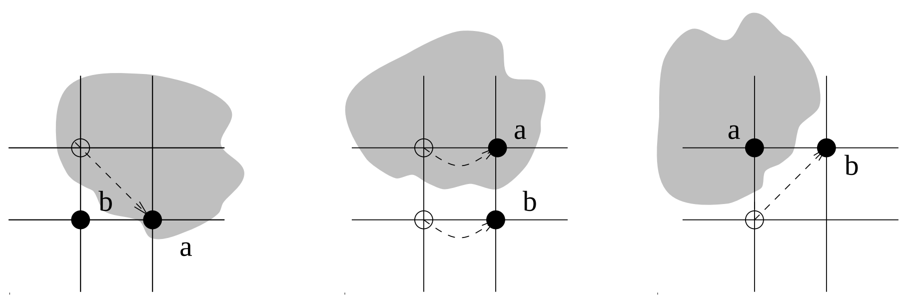
Mais precisamente, a cada passo do algoritmo podemos ter uma das situações ilustradas na figura 2.4.



(a) (b) (c) (d)

Figura 2.4: Situações possíveis durante a extração do contorno.

Suponha que a está diretamente acima de b, como na figura 2.4(a). A partir desta posição, o algoritmo pode avançar a, b ou ambos de modo a manter a dentro e b fora do fragmento, como mostrado nas figuras 2.5 (a)− 2.5 (c). O tratamento dos outros casos (figuras 2.4(b–c)) é análogo.



(a) (b) (c)

Figura 2.5: Passos alternativos do algoritmo para a situação da figura 2.4(a).

Obtemos por este processo uma seqüência de pares (a_i, b_i), onde a_i acompanha o contorno pelo lado de dentro (isto é, $v[a_i] \geq \delta_{med}$), e b_i pelo lado de fora (isto é, $v[b_i] \leq \delta_{med}$). Para maior precisão na comparação dos fragmentos, calculamos por interpolação linear o ponto (fracionário) p_i da aresta (a_i, b_i) onde o nível de cinza é exatamente δ_{med}. Veja a figura 2.6.

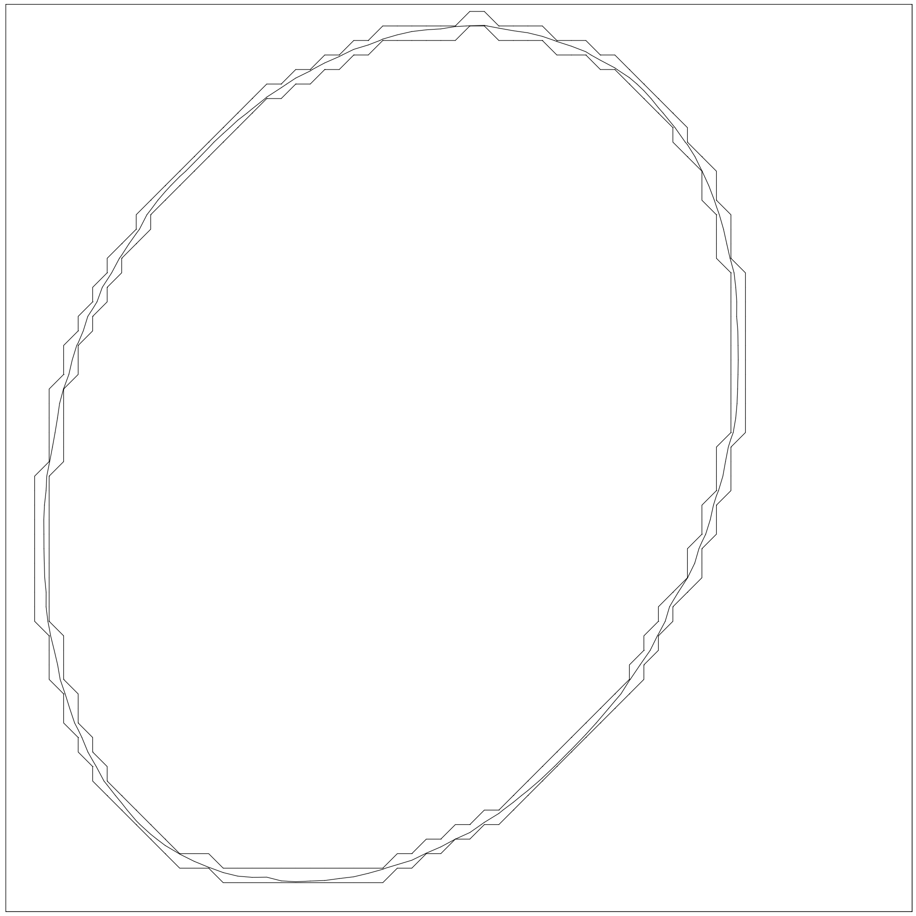


Figura 2.6: Contornos extraídos: externo, interpolado e interno.

O resultado deste processo é uma curva poligonal fechada $p_0, p_1, .. p_n = p_0$, onde cada p_i é um ponto do plano $\mathbf{R}^2$. Os lados desta poligonal têm comprimento variável entre 0 e $\sigma\sqrt{2}$, onde σ é a distância entre *pixels* consecutivos. Supõe-se que a sequência p_i é periódica, isto é $p_{i+n} = p_i$, para todo i.

Este algoritmo simples de extração de contorno deixa muito a desejar. Mesmo para objetos planos e finos, como os fragmentos de papel da figura 2.2, o algoritmo é freqüentemente enganado por rebarbas e fiapos, como os que são visíveis no lado direito da figura 2.2(a).

Outra fraqueza séria do algoritmo é o fato de que, na prática, a cor do fragmento δ_{max} varia bastante de ponto para ponto, e portanto não existe um único valor correto do limiar δ_{med}. Para tais aplicações, seria necessário usar um algoritmo de segmentação mais sofisticado, no qual δ_{max} varia (e portanto δ_{med} também varia) de pixel para pixel baseado nos valores da imagem dos pixels vizinhos.

Capítulo 3

Filtragem de sinais

Após a obtenção dos contornos, como descrito no capítulo 2, submetemos os mesmos a um processo de filtragem que é uma das etapas mais importantes para o sucesso da solução proposta.

No capítulo 4 estudaremos o problema da filtragem de curvas e sua função no nosso algoritmo. Antes disso, precisamos recapitular aqui alguns conceitos básicos de processamento de sinais que serão úteis para entendermos a filtragem dos contornos.

3.1 Representação de Fourier

Sabemos que um sinal contínuo com período T pode ser aproximado com qualquer grau de precisão desejado por uma *série de Fourier finita*, também conhecida como um *polinômio trigonométrico* – uma soma finita de senóides e cossenóides, cujos períodos dividem T:

$$x(t) = a_0 + \sum_{k=1}^{n} \left[a_k \cos\left(\frac{2\pi}{T}kt\right) + b_k \sin\left(\frac{2\pi}{T}kt\right)\right] \tag{3.1}$$

Os coeficientes a_k, b_k são números reais, os *coeficientes de Fourier* do sinal. Cada termo $a_k \cos(2\pi kt/T) + b_k \sin(2\pi kt/T)$ é chamado *componente* do sinal; o índice k é a *freqüência* da componente, e o valor $\sqrt{a_k^2 + b_k^2}$ é a *amplitude* da mesma. É fácil ver que a função (3.1)é contínua e periódica, de período T.

3.2 Representação complexa

A fórmula (3.1) pode ser escrita também como

$$x(t) = \sum_{k=-n}^{n} c_k \exp(\frac{2\pi\mathbf{i}}{T}kt) \tag{3.2}$$

onde $\mathbf{i}$ é a unidade imaginária ($\sqrt{-1}$), e os parâmetros c_k são os *coeficientes complexos de Fourier*,

$$c_k = \begin{cases} \frac{a_k}{2} + \mathbf{i}\frac{b_k}{2} & \text{para } k > 0 \\ a_0 & \text{para } k = 0 \\ \frac{a_k}{2} - \mathbf{i}\frac{b_k}{2} & \text{para } k < 0 \end{cases} \tag{3.3}$$

Inversamente, temos

$$a_0 = c_0, \qquad a_k = c_k + c_{-k}, \quad b_k = c_k - c_{-k}, \quad \text{para} \quad k = 1, .. n \tag{3.4}$$

Note que o número de coeficientes c_k na fórmula 3.2 é igual ao número total de coeficientes a_k, b_k na fórmula (3.1). Observe também que c_{-k} é o conjugado complexo de c_k para todo k; e que c_0 é real. Portanto, em qualquer das duas representações, a série de Fourier tem $2n + 1$ graus de liberdade.

Verifica-se que os coeficientes c_k podem ser calculados pela fórmula

$$\frac{1}{T}\int_0^T x(t)\, \exp\left(-2\pi k \mathbf{i} t\right) dt \tag{3.5}$$

3.3 Derivadas de um sinal

Se $x(t)$ é um sinal da forma (3.1) ou (3.2), sua derivada primeira em relação a t é

$$\begin{aligned} x^{\bullet}(t) &= \frac{dx}{dt}(t) \\ &= \sum_{k=1}^{n} \frac{2\pi k}{T}\left[-a_k \sin\left(\frac{2\pi k}{T}t\right) + b_k \cos\left(\frac{2\pi k}{T}t\right)\right] \\ &= \sum_{k=-n}^{n} \frac{2\pi \mathbf{i} k}{T} c_k \exp(\frac{2\pi \mathbf{i}}{T}kt) \end{aligned} \tag{3.6}$$

e sua derivada segunda é

$$\begin{aligned} x^{\bullet\bullet}(t) &= \frac{d^2x}{dt^2}(t) \\ &= \sum_{k=1}^{n} \left(\frac{2\pi k}{T}\right)^2 \left[-a_k \cos\left(\frac{2\pi k}{T}t\right) - b_k \sin\left(\frac{2\pi k}{T}t\right)\right] \\ &= \sum_{k=-n}^{n} -\left(\frac{2\pi k}{T}\right)^2 c_k \exp(\frac{2\pi \mathbf{i}}{T}kt) \end{aligned} \tag{3.7}$$

3.4 Representação por amostragem

O *teorema de amostragem de Nyquist* diz que se n é a freqüência máxima que ocorre na representação de Fourier de um sinal periódico, então podemos representar este último por uma seqüência de valores espaçados uniformemente no tempo, sem perder qualquer informação, desde que tomemos pelo menos $m = 2n + 1$ amostras por período.

Mais exatamente, se tomarmos m amostras igualmente espaçadas ($t_i = iT/m$) no intervalo $(0..T]$ de cada um dos elementos da base de Fourier, podemos verificar que os m vetores de $\mathbf{R}^m$ resultantes

$$u_k = \left(\cos(\frac{2\pi \mathbf{i} k t_i}{T}) : i = 1..m\right) = \left(\cos(\frac{2\pi \mathbf{i} k i}{m}) : i = 1..m\right), k = 0..n$$

$$v_k = \left(\sin(\frac{2\pi \mathbf{i} k t_i}{T}) : i = 1..m\right) = \left(\sin(\frac{2\pi \mathbf{i} k i}{m}) : i = 1..m\right), k = 1..n$$

são linearmente independentes. Portanto, dadas m amostras $x_i = x(t_i)$ de um polinômio trigonométrico, existe uma única combinação linear dos vetores $(u_k : k = 0..n)$ e $(v_k : k = 1..n)$ que reproduz essas amostras; e portanto uma única combinação linear da forma (3.1) ou (3.2) que coincide com as amostras x_i nos instantes t_i.

3.5 Reconstrução do sinal a partir das amostras

Implícita no teorema de Nyquist é a afirmação de que, dado um número $m = 2n + 1$ de amostras $x_i = x(t_i)$ ($t_i = iT/m$) de um sinal periódico x, igualmente espaçadas, é possível reconstruir o valor $x(t)$ do mesmo em qualquer instante t dado.

Este processo, inverso da amostragem, é denominado *reconstrução* do sinal, e conceitualmente pode ser descrito pela fórmula

$$x(t) = \sum_{i=1}^{m} x_i \psi_i(t)$$

onde as funções ψ_i são polinômios trigonométricos de período T e freqüência máxima n, tais que

$$\psi_i(t_j) = \delta_{ij} = \begin{cases} 1 & \text{se } i = j \\ 0 & \text{se } i \neq j \end{cases} \tag{3.8}$$

para $i, j \in \{1..m\}$. É fácil verificar que as funções

$$\psi_i(t) = \frac{1}{m} \sum_{k=1}^{m} \cos\left(\frac{2\pi k}{T}(t - t_i)\right)$$

satisfazem a condição (3.8).

3.6 Norma de um sinal

A *norma* $\mathbf{L}_2$ de um sinal (real ou complexo) x com período T é definida por

$$||x||_2 = \left[\frac{1}{T}\int_0^T |x(t)|^2 dt\right]^{1/2}$$

Esta norma está associada ao *produto escalar* $\mathbf{L}_2$ de dois sinais de período T,

$$\langle f|g\rangle = \frac{1}{T}\int_0^T f(t)g^*(t)dt$$

pela identidade

$$||x||_2 = \langle x|x\rangle^{1/2}$$

Verifica-se que as funções da base de Fourier — $\exp(\frac{2\pi \mathbf{i} k}{T}t)$, ou $\cos(\frac{2\pi k}{T}t)$ e $\sin(\frac{2\pi k}{T}t)$ — são mutuamente ortogonais em relação ao produto escalar $\langle \ | \ \rangle$; e que

$$\begin{aligned}
\|1\|_2 &= 1 \\
\left\|\frac{1}{T}\exp(\frac{2\pi \mathbf{i} k}{T}t)\right\|_2 &= 1 \\
\left\|\cos(\frac{2\pi k}{T}t)\right\|_2 &= \sqrt{1/2} \\
\left\|\sin(\frac{2\pi k}{T}t)\right\|_2 &= \sqrt{1/2}
\end{aligned}$$

para $k \neq 0$.

Segue-se que, se x é uma série trigonométrica da forma (3.1) ou (3.2), então

$$\left[a_0^2 + \frac{1}{2}\sum_{k=1}^{n}\left(|a_k|^2 + |b_k|^2\right)\right]^{1/2} = \left[\sum_{k=-n}^{n}|c_k|^2\right]^{1/2} \tag{3.9}$$

ou seja,

$$\|x\|_2 = \left[\sum_{k=-n}^{n}|c_k|^2\right]^{1/2} \tag{3.10}$$

3.7 Transformada discreta de Fourier

Como observado na seção 3.5, o teorema de Nyquist diz que os $m = 2n + 1$ coeficientes c_k de um polinômio trigonométrico da forma (3.2) podem ser determinados a partir de m amostras x_i igualmente espaçadas no período T. Se considerarmos a seqüência de amostras e a seqüência de coeficientes c_k como vetores do espaço $\mathbf{C}^m$, podemos considerar o cálculo dos coeficientes a partir das amostras, ou vice-versa, como sendo uma transformação linear (mudança de base) neste espaço. Verifica-se que esta é uma transformação ortogonal, ou seja,

$$\sum_{i=1}^{m} |x_i|^2 = \sum_{k=-n}^{n} |c_k|^2 = (\|x\|_2)^2 \tag{3.11}$$

3.8 Filtros

Um *filtro* é um termo geral para qualquer operador $\mathcal{F}$ que transforma um sinal x em outro sinal $\mathcal{F}x$. Aqui, discutiremos alguns filtros de especial interesse para o nosso trabalho.

Um filtro $\mathcal{F}$ é dito *linear* se satisfaz $\mathcal{F}(ax) = a\mathcal{F}x$ e $\mathcal{F}(x+y) = (\mathcal{F}x) + (\mathcal{F}y)$, para qualquer $a \in \mathbf{R}$ e quaisquer sinais x e y. Um filtro é *invariante no tempo* se satisfaz $\mathcal{F}(\Delta_\tau x) = \Delta_\tau(\mathcal{F}x)$, onde $\Delta_\tau x$ denota o sinal x retardado por τ, isto é, $(\Delta_\tau x)(t) = x(t-\tau)$.

Pode-se mostrar [6] que o efeito de todo filtro linear e invariante no tempo em um sinal trigonométrico é equivalente a multiplicar cada coeficiente complexo de Fourier, c_k, por um número complexo $w_k = \alpha_k + \mathbf{i}\beta_k$. O número w_k é o *fator de atenuação* do filtro *para a freqüência* k. Para que os valores do sinal filtrado sejam números reais, é necessário que w_{-k} seja o conjugado complexo de w_k. Nesse caso, a filtragem substitui cada par de coeficientes reais a_k, b_k da fórmula 3.1 por novos coeficientes $\alpha_k a_k - \beta_k b_k$,$\beta_k a_k + \alpha_k b_k$. Este resultado nos dá um modo conveniente e padrão para descrever um filtro linear e invariante no tempo: basta especificar a lista de fatores de atenuação $(w_k : k = -n, ...n)$.

Dizemos que um filtro $\mathcal{F}$ é *simétrico no tempo* se satisfaz $\mathcal{F}(\overleftarrow{x}) = \overleftarrow{(\mathcal{F}x)}$ onde $\overleftarrow{x}$ é o sinal x invertido no tempo, isto é $\overleftarrow{x}(t) = x(-t)$. Verifica-se que o filtro é simétrico no tempo se e somente se w_k é real e $w_{-k} = w_k$.

3.8.1 Filtragem por convolução

Prova-se que todo filtro linear e invariante no tempo transforma um sinal x num sinal x' dado por:

$$x'(t) = \int_{-\infty}^{\infty} x(t-\tau) f(\tau) d\tau$$

onde f é o *núcleo* do filtro, uma função de $\mathbf{R}$ em $\mathbf{C}$ [6].

Para certos filtros, a "função" f pode ter valores infinitos para certos valores de τ, desde que sua integral seja finita em qualquer intervalo. Isto é, o núcleo f pode ser uma *distribuição* e não uma função. O exemplo clássico é o *filtro identidade* $\mathcal{I}x = x$, cujo núcleo é o *impulso de Dirac* $\delta(t)$ [6], que é zero para $t \neq 0$, e cuja integral é 1 em qualquer intervalo que contenha $t = 0$.

Se x é um sinal periódico de período T, prova-se que o efeito do filtro equivale também a multiplicar cada coeficiente c_k da série de Fourier de x pelo número complexo

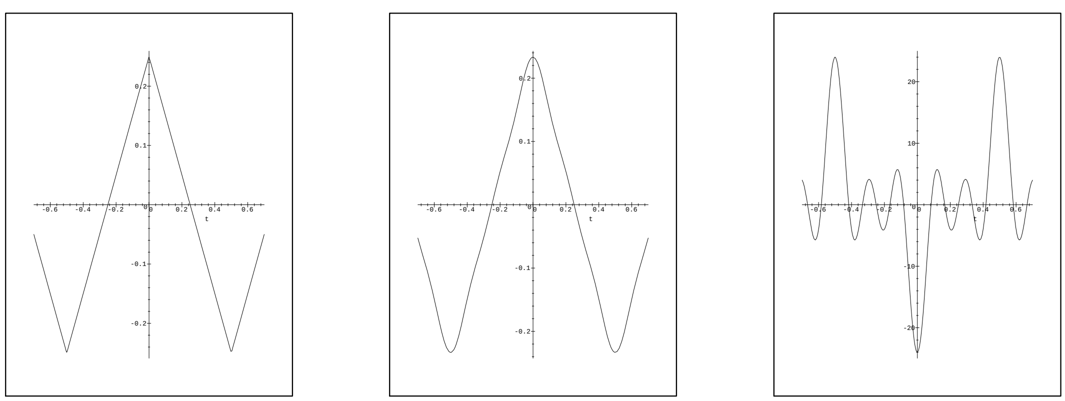
$$w_k = \int_{-\infty}^{\infty} \exp\left(-\frac{2\pi k \mathbf{i} \tau}{T}\right) f(\tau) d\tau$$

ou seja, pelo coeficiente complexo de freqüência k do núcleo $f(\tau)$.

3.8.2 Filtro retangular

Um *filtro retangular passa-baixas de largura* k_{max} tem $w_k = 1$ para $|k| \leq k_{max}$, e $w_k = 0$ para $|k| > k_{max}$. Isto é, o filtro retorna apenas os componentes do sinal cuja freqüência é menor ou igual a k_{max}. (Note que ao aproximarmos um sinal arbitrário por uma série trigonométrica finita, como na equação 3.1, já estamos implicitamente filtrando o sinal por um filtro retangular passa-baixas [6].)

Este tipo de filtro tem uma característica indesejável, o *efeito Gibbs*: ele dá origem a oscilações espúrias no sinal filtrado. A figura 3.1 mostra este efeito: se removermos do sinal ilustrado em (a) todas as componentes com freqüência > 5, obtemos a função ilustrada em (b). As oscilações espúrias desta última ficam mais evidentes no gráfico da sua segunda derivada, mostrado em (c).



(a) (b) (c)

Figura 3.1: O efeito Gibbs.

O efeito Gibbs é um inconveniente sério para nós, pois o nosso algoritmo de reconhecimento de fragmentos adjacentes depende muito da curvatura dos contornos, que é calculada a partir de sua segunda derivada (vide seção 4.1). Portanto, precisamos utilizar filtros que não produzam tais oscilações espúrias.

3.8.3 Filtro gaussiano

Os *filtros gaussianos* são ideais para os nossos objetivos, porque, como mostrado no artigo de J.Babaud, A.Witkin e outros [2], são praticamente os únicos filtros que não introduzem novos máximos e mínimos na segunda derivada da função. Por definição um *filtro gaussiano* é um filtro linear e invariante no tempo cujo núcleo é uma distribuição gaussiana.

$$G_\sigma(\tau) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{\tau^2}{2\sigma^2}) \tag{3.12}$$

O parâmetro σ, que é o desvio padrão da distribuição, é também chamado de *duração característica* ou *escala de tempo* do filtro gaussiano.

3.8.4 Extensão do filtro gaussiano

Da fórmula (3.12), conclui-se que o núcleo $G_\sigma(\tau)$ do filtro gaussiano tem valor máximo em $\tau = 0$, e diminui rapidamente à medida que τ aumenta. Em particular,

$\lvert\tau\rvert$	$\int G_\sigma(\tau)d\tau$
$> \sigma$	< 0.318
$> 2\sigma$	< 0.0456
$> 3\sigma$	< 0.00270
$> 4\sigma$	< 0.0000634
$> 5\sigma$	$< 0.574 \;\; 10^{-6}$
$> 6\sigma$	$< 0.198 \;\; 10^{-8}$

Segue também da fórmula (3.13) e pela tabela acima que, se x é um sinal tal que $|x(t)| < M$ num intervalo finito $[a, b]$, e $x(t) = 0$ para $t \notin [a, b]$, então para todo $t \notin [a - 6\sigma, b + 6\sigma]$,

$$|(f * G_\sigma)(t)| \; < \; M \int_{6\sigma}^{\infty} G_\sigma(\tau)d\tau \; < \; 10^{-8}M$$

3.8.5 Filtragem gaussiana na representação de Fourier

Conforme visto na seção 3.8.1, a filtragem gaussiana equivale a multiplicar a componente de freqüência k do sinal pelo fator real

$$\begin{aligned} w_k &= \int_{-\infty}^{\infty} \exp\left(-\frac{2\pi k \mathbf{i} \tau}{T}\right) G_\sigma(\tau)d\tau \\ &= \exp\left(-\left(\frac{k\sigma\pi}{T}\right)^2\right) \end{aligned} \tag{3.13}$$

Segue-se da fórmula (3.13) que o peso w_k decresce muito rapidamente à medida que k aumenta. Em particular, uma componente cujo período T/k é menor que a duração característica σ do filtro tem freqüência $k > T/\sigma$, e portanto, será atenuada por um fator menor que

$$\exp\left(-\left(\frac{T}{\sigma}\frac{\sigma\pi}{T}\right)^2\right) = \exp(-\pi^2) \approx 10^{-4}$$

Mais geralmente, se $k > mT/\sigma$, o valor de w_k é limitado conforme a tabela abaixo:

k	w_k
$> 1T/\sigma$	$< 0.518\ \ 10^{-4}$
$> 2T/\sigma$	$< 0.268\ \ 10^{-8}$
$> 3T/\sigma$	$< 0.139\ \ 10^{-12}$
$> 4T/\sigma$	$< 0.716\ \ 10^{-17}$
$> 5T/\sigma$	$< 0.371\ \ 10^{-21}$
$> 6T/\sigma$	$< 0.192\ \ 10^{-25}$

Note que as componentes com período T/k muito menores que σ são praticamente eliminadas. Portanto, na prática, w_k pode ser considerado nulo quando $k \geq 2T/\sigma$. Isto significa que, na representação por série de Fourier de um sinal filtrado com G_σ, só precisamos calcular os termos da série até freqüência $k_{max} = 2\lceil T/\sigma \rceil$.

3.8.6 Composição de filtros gaussianos

Segue da fórmula (3.13), que se aplicarmos a uma curva c dois filtros gaussianos em seqüência, com durações características σ_1 e σ_2, o efeito total é equivalente a um único filtro gaussiano de duração característica $\sigma = \sqrt{\sigma_1^2 + \sigma_2^2}$.

3.8.7 Reconstrução de sinais com filtragem gaussiana

Estritamente falando, quando amostramos um sinal $\tilde{x}$ que resulta da aplicação de um filtro gaussiano a um sinal arbitrário x, não é em geral possível reconstruir nem x nem $\tilde{x}$ a partir de um número finito de amostras, pois o critério de Nyquist não é necessariamente satisfeito (seção 3.4).

Entretanto, se tivermos pelo menos $4\lceil T/\sigma \rceil$ amostras igualmente espaçadas ao longo do período, segue da análise feita na seção 3.8.5 que é possível reconstruir uma aproximação do sinal filtrado $\tilde{x}$, com erro relativo inferior a $10^{-8}||x||_2$.

3.9 Sinais lineares por partes

No nosso caso, freqüentemente precisamos filtrar e reamostrar um sinal x que é dado por amostras $(t_1, x_1), .. (t_n, x_n)$ irregularmente espaçadas. Para tornarmos o problema solúvel, vamos interpretar tais sinais como funções lineares por partes (LP) que interpolam as amostras dadas. Ou seja,

$$x(t) = \begin{cases} a_i t + b_i & \text{se } t_i \leq t < t_{i+1}, \quad i = 1..n-1 \\ 0 & \text{caso contrário} \end{cases} \tag{3.14}$$

onde os coeficientes a_i, b_i são tais que $x(t_i) = x_i$. Vamos supor também que $x_1 = x_n = 0$ de modo que a função é contínua.

3.9.1 Filtragem gaussiana por difusão

O filtro gaussiano também pode ser visto como o resultado de um processo evolutivo análogo à difusão do calor em um domínio homogêneo e unidimensional. Neste processo, a distribuição de temperatura inicial sobre o domínio evolui ao longo do tempo de acordo com equações diferenciais estritamente locais, de tal forma que o calor passa das partes mais quentes para as partes mais frias vizinhas.

Nesta abordagem, o sinal $x(t)$ é visto como um membro inicial $x(t) = x(t, 0)$ de uma família de sinais $x(t, \tau)$, definida pela equação diferencial

$$\frac{\partial x}{\partial \tau}(t, \tau) = \frac{\partial^2}{\partial t^2} x(t, \tau) \tag{3.15}$$

Uma família de sinais que satisfaz a equação acima é

$$\begin{aligned} x(t, 0) &= \delta(t), \\ x(t, \tau) &= \frac{1}{2\sqrt{\tau\pi}} \exp\left(-\frac{t^2}{4\tau}\right) = G_{\sqrt{2\tau}}(t) \quad \text{para} \quad \tau > 0 \end{aligned} \tag{3.16}$$

A partir desta observação, pode-se concluir que, numa solução geral $x(t, \tau)$ da equação (3.15), o sinal $x'(t) = x(t, \tau)$, para τ fixo, é uma versão do sinal original $x(t, 0)$, suavizada por um filtro gaussiano de duração característica $\sigma = \sqrt{2\tau}$.

3.10 Filtragem de um sinal linear por partes

Nesta seção, nosso objetivo é aplicar um filtro gaussiano a uma função LP $x(t)$, e reamostrar a função filtrada $x'(t)$ em m pontos igualmente espaçados $t'_1, t'_2 .. t'_m$. Isto significa calcular os valores

$$x'_i = x'(t'_i) = \int_{-\infty}^{\infty} x(t'_i - \tau) G_\sigma(\tau) d\tau$$

Uma função LP da forma (3.14) pode ser vista como a soma de $n-2$ termos

$$x(t) = \sum_{i=2}^{n-1} x_i \Lambda_{t_{i-1},t_i,t_{i+1}}(t)$$

onde $\Lambda_{a,b,c}(t)$ é a *função tenda* ilustrada na figura 3.2:

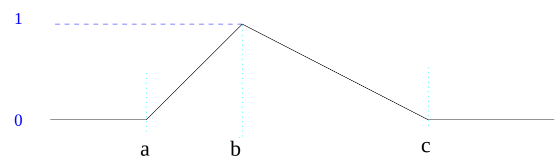


Figura 3.2: Função tenda.

Em teoria, poderíamos filtrar cada tenda Λ separadamente, e somar os resultados. Entretanto, esta abordagem é numericamente instável, pois leva a cancelamento de termos grandes. Para reduzir este problema, decompomos a função LP $x(t)$ numa tenda $x_r \Lambda_{t_1,t_r,t_n}(t) + x^{(0)}(t) + x^{(1)}(t)$, onde $r = n/2$, e $x^{(0)}(t)$ e $x^{(1)}(t)$ são duas funções LP definidas por amostras nos instantes $t_1 .. t_r$ e $t_r .. t_n$ respectivamente. Veja as figuras 3.3 e 3.4.

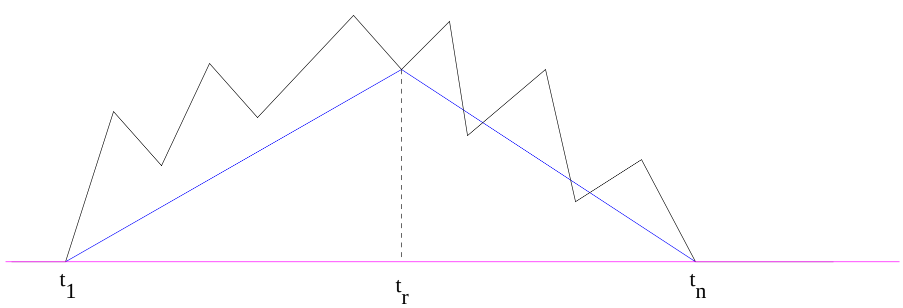


Figura 3.3: Função linear por partes genérica.

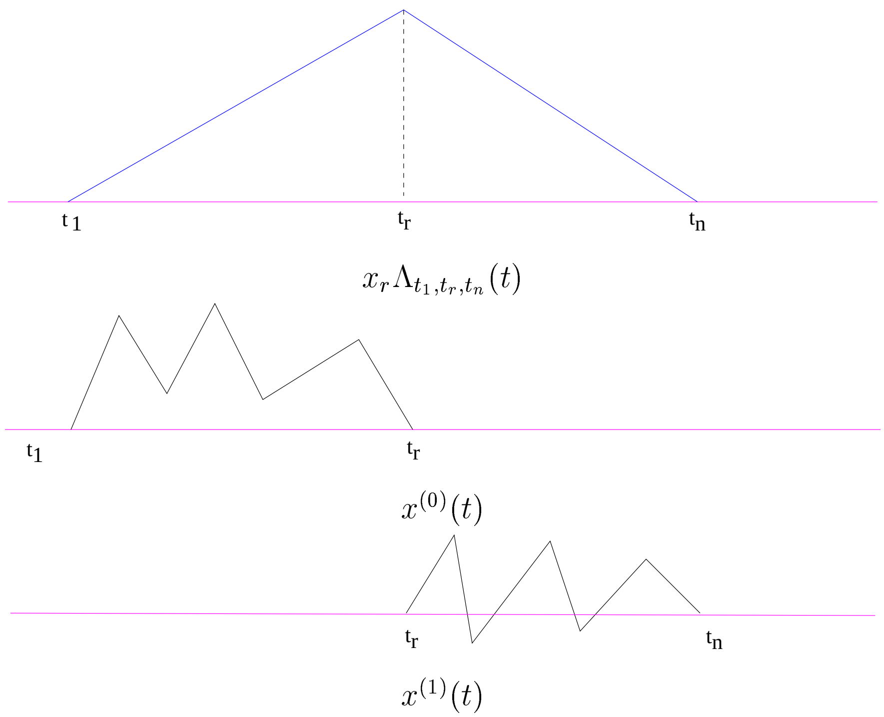


Figura 3.4: Decomposição da função da figura 3.3 em três componentes: uma tenda e duas funções LP.

Aplicando recursivamente esta decomposição, obtemos uma soma de tendas cujas alturas diminuem rapidamente à medida que suas larguras diminuem. Cada componente da figura 3.4 é filtrada e reamostrada, e os resultados são somados. As componentes $x^{(0)}(t)$ e $x^{(1)}(t)$ são filtradas aplicando-se recursivamente o mesmo procedimento. A componente Λ_{t_1,t_r,t_n} é filtrada pela fórmula:

$$(\Lambda_{a,b,c} * G_\sigma)(t) = \int_{-\infty}^{\infty} \Lambda_{a,b,c}(t-\tau)G_\sigma(\tau)d\tau$$

Por sua vez, $\Lambda_{a,b,c}(t)$ pode ser escrita como

$$\Lambda_{a,b,c}(t) = \frac{1}{b-a}\mathrm{ramp}(t-a) - \left(\frac{1}{b-a} + \frac{1}{c-b}\right)\mathrm{ramp}(t-b) + \frac{1}{c-b}\mathrm{ramp}(t-c)$$

onde

$$\mathrm{ramp}(y) = \begin{cases} y & \text{se } y \geq 0 \\ 0 & \text{se } y < 0 \end{cases} \tag{3.17}$$

portanto,a tenda filtrada pode ser expandida em

$$(\Lambda_{a,b,c} * G_\sigma)(t) = \frac{1}{b-a}\psi_\sigma(t-a) - \left(\frac{1}{b-a} + \frac{1}{c-b}\right)\psi_\sigma(t-b) + \frac{1}{c-b}\psi_\sigma(t-c) \tag{3.18}$$

onde

$$\psi_\sigma(y) = (\text{ramp} * G_\sigma)(y) = \frac{y}{2}\left(1 + \text{erf}\left(\frac{y\sqrt{2}}{\sigma}\right)\right) + \frac{\sigma}{\sqrt{2\pi}}\exp\left(\frac{-y^2}{2\sigma^2}\right)$$

e

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-\tau^2} d\tau$$

Conforme observado na seção 3.8.6, podemos supor que $(\Lambda_{a,b,c} * G_\sigma)(t)$ é desprezível fora do intervalo $[a - 6\sigma, c + 6\sigma]$

Este algoritmo é descrito em detalhes a a seguir. Ele recebe uma função LP, dada pelas amostras $(t_1, x_1), ..(t_n, x_n)$ (com x_1 e x_n arbitrários) e filtra a função LP $\hat{x}$ definida pelos pontos $(t_i, \hat{x}_i)$, onde

$$\hat{x}_i = x_i - \left(\frac{t_n - t_i}{t_n - t_1}x_1 + \frac{t_i - t_1}{t_n - t_1}x_n\right)$$

Veja a figura 3.5. O algoritmo soma às amostras x'_i a função $\hat{x}$, filtrada e calculada nos instantes t'_i. Na primeira chamada, x_1, x_n e as amostras x'_i devem ser nulas.

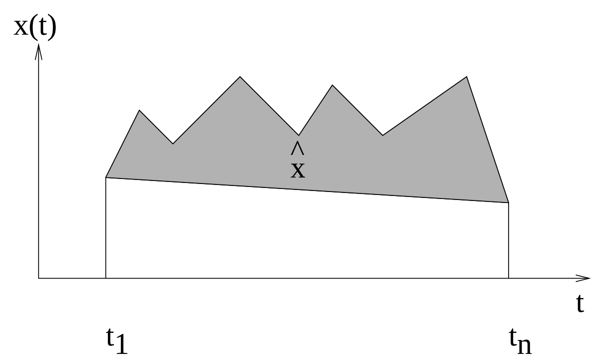


Figura 3.5: Função $\hat{x}$.

Algoritmo 1 Filtragem de uma função linear por partes
Entrada:

- *a duração característica σ do filtro gaussiano;*
- *as amostras $(t_1, x_1), ..(t_n, x_n)$ da função LP.*
- *os instantes de amostragem $(t'_1..t'_m)$ para a curva filtrada*

Entrada e saída:

- *os valores $(x'_1, ..x'_m)$ da função filtrada nos tempos $t'_1, t'_2..t'_m$.*

Passos:

1. Se $n = 2$, a função $\hat{x}$ é identicamente nula, e não há nada a fazer.

2. Senão, seja $r = \lceil n/2 \rceil$. Aplique o algoritmo 1 às amostras $(t_1, x_1)..(t_r, x_r)$, e depois às amostras $(t_r, x_r)..(t_n, x_n)$, acumulando o resultado nas amostras $(t'_1, x'_1)..(t'_r, x'_r)$ e $(t'_r, x'_r)..(t'_m, x'_m)$ respectivamente.

3. Calcule
$$\hat{x}_r = x_r - \left(\frac{t_n - t_r}{t_n - t_1} x_1 + \frac{t_r - t_1}{t_n - t_1} x_n \right)$$

4. Usando a fórmula (3.18), calcule $\delta'_j = \hat{x}_r(\Lambda_{t_1,t_r,t_n} * G_\sigma)(t'_j)$, e some δ'_j a x'_j, para todo j tal que $t_1 - 6\sigma \leq t'_j \leq t_n + 6\sigma$.

Capítulo 4

Filtragem de curvas

Neste capítulo, vamos estender os conceitos de filtragem de sinais para curvas, e apresentar o algoritmo que utilizamos para filtrar os contornos.

No nosso trabalho, a filtragem é utilizada com duas funções diferentes. Primeiramente, é utilizada para remover irregularidades não significativas dos contornos — irregularidades causadas por artefatos da aquisição das imagens e do processo de extração dos contornos, ou por erosão do material após sua fragmentação. Para este fim é necessário remover as componentes de freqüências mais altas dos contornos, que são as mais afetadas por estas perturbações. Em segundo lugar, a filtragem é necessária para evitar "aliasing" quando sub-amostramos os contornos a fim de aumentar a velocidade da identificação de segmentos similares. Este uso será explicado com mais detalhes no capítulo 10.

Todos os conceitos e algoritmos deste capítulo podem ser estendidos trivialmente para curvas no espaço $\mathbf{R}^3$, ou espaços de dimensão maior.

A etapa de filtragem é muito mais crítica na nossa aplicação do que na maioria das aplicações de reconhecimento de padrões [9]. Nestas, há geralmente uma nítida separação entre as escalas do sinal e do ruído, pois os objetos têm contornos suaves e o ruído é principalmente devido a erros de quantização.

Na nossa aplicação, entretanto, a natureza fractal dos objetos significa que há sinal útil em todas as escalas de detalhe, e na verdade desejamos preservar o sinal nas escalas mais finas possíveis.

Além disso, como veremos, a comparação dos contornos é feita com base nas curvaturas, e portanto é essencial que a filtragem dos dois trechos correspondentes de mesmo contorno produzam curvas bem semelhantes, mesmo que os dois tenham sido digitalizados em posições e ângulos diferentes.

Esta exigência se torna mais crítica pela presença freqüente de reentrâncias estreitas nos contornos, devidas a rachaduras e riscos nos fragmentos (Veja a figura 4.1).

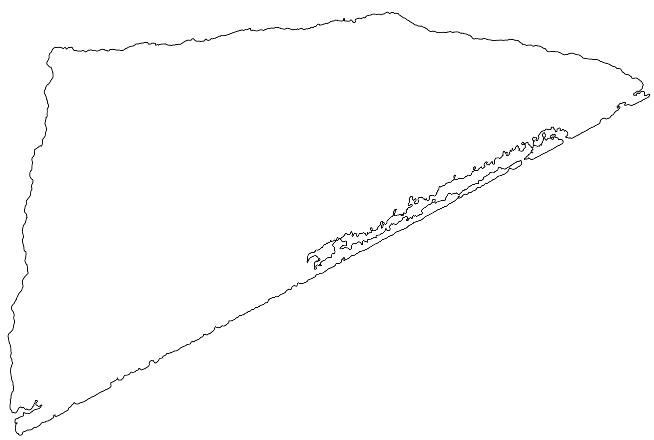


Figura 4.1: Contorno de um fragmento de cerâmica com uma rachadura.

Com técnicas de filtragem tradicional, esses defeitos afetam um trecho relativamente extenso da curva filtrada, destruindo a semelhança com o trecho correspondente do fragmento vizinho.

4.1 Conceitos básicos

O contorno de um fragmento simples (sem buracos) pode ser convenientemente descrito por uma seqüência contínua de pontos no plano $\mathbf{R}^2$, isto é uma *curva plana*. Mais especificamente, nos referimos a uma curva contínua, fechada, no plano cartesiano $\mathbf{R}^2$, que é uma função periódica $c(t) = (x(t), y(t))$ de $\mathbf{R}$ para $\mathbf{R}^2$.

A *velocidade* da curva num instante t é a derivada $v(t) = c^{\bullet}(t)$ da posição $c(t)$ em relação ao parâmetro t. A velocidade $v(t)$ pode ser interpretada como outra curva, denominada o *hodógrafo* da curva $c(t)$. Note-se que o comprimento da curva entre os instantes t_0 e t_1 é $\displaystyle\int_{t_0}^{t_1} |v(t)|dt$. A *aceleração* da curva é sua derivada segunda, $a(t) = c^{\bullet\bullet}(t)$.

Note que há uma infinidade de curvas com a mesma forma, que diferem entre si na escolha do parâmetro t. Na *parametrização natural*, t é o comprimento da curva, medido a partir de um ponto inicial arbitário; de modo que a velocidade $|v(t)|$ é sempre igual a 1.

4.2 Filtragem paramétrica de curvas

Poderíamos pensar que filtrar uma curva $c(t)$ equivale a filtrar separadamente as coordenadas $x(t)$ e $y(t)$ consideradas como sinais — isto é funções reais do "tempo" t. Esta abordagem é dita *filtragem paramétrica*. Entretanto, como veremos nas próximas seções, a filtragem paramétrica de uma curva, ainda que por um filtro linear, modifica o comprimento da mesma de maneira não linear. Devido a este problema, a filtragem de curvas é um processo intrinsecamente não linear, e portanto a teoria clássica de análise de sinais não se aplica facilmente a este problema.

4.2.1 Representação de Fourier

Na abordagem paramétrica, uma curva $c(t)$ com período T pode ser encarada como dois sinais contínuos $x(t)$ e $y(t)$, de mesmo período. Portanto, como visto no capítulo 3, $c(t)$ pode ser aproximada com qualquer grau de precisão desejado por uma série de Fourier finita.

$$c(t) = a_0 + \sum_{k=1}^{n} \left[a_k \cos\left(\frac{2\pi k}{T} t\right) + b_k \sin\left(\frac{2\pi k}{T} t\right)\right] \tag{4.1}$$

onde os coeficientes a_k, b_k são vetores do $\mathbf{R}^2$. Cada termo $a_k \cos(2\pi kt/T) + b_k \sin(2\pi kt/T)$ é chamado *componente* da curva; o índice k é a freqüência da componente. Curvas com esta representação são ditas *curvas trigonométricas*. A curva (4.1) é obviamente fechada, contínua e periódica de período T.

Assim como no caso de sinais, a fórmula (4.1) pode ser escrita também como

$$c(t) = \sum_{k=-n}^{n} c_k \exp(\frac{2\pi \mathbf{i}}{T} kt) \tag{4.2}$$

onde c_k são vetores de $\mathbf{C}^2$, dados por

$$c_k = \begin{cases} \dfrac{a_k}{2} + \mathbf{i}\dfrac{b_k}{2} & \text{para } k > 0 \\ a_0 & \text{para } k = 0 \\ \dfrac{a_k}{2} - \mathbf{i}\dfrac{b_k}{2} & \text{para } k < 0 \end{cases}$$

Note que o termo $a_0 = c_0$ afeta somente a posição da curva no plano. Se $c(t)$ é a curva trigonométrica (4.1), sua velocidade é dada por

$$v(t) = c^{\bullet}(t) = \frac{dc}{dt}(t) = \sum_{k=1}^{n} \frac{2\pi k}{T}(-a_k \sin\left(\frac{2\pi k}{T} t\right) + b_k \cos\left(\frac{2\pi k}{T} t\right))$$

e sua aceleração por

$$a(t) = c^{\bullet\bullet}(t) = \sum_{k=1}^{n} (\frac{2\pi k}{T})^2 (-a_k \cos\left(\frac{2\pi k}{T} t\right) - b_k \sin\left(\frac{2\pi k}{T} t\right))$$

4.2.2 Comprimento de onda aparente

A grosso modo, uma variação no coeficiente da componente de freqüência k da curva dá origem a oscilações na curva de amplitude $\alpha \leq \sqrt{|a_k|^2 + |b_k|^2} = 2|c_k|$. A distância entre duas oscilações sucessivas dessa componente, em média, é

$$\lambda_k = \frac{L}{k} \tag{4.3}$$

onde L é o comprimento total da curva. O parâmetro λ_k é dito o *comprimento de onda médio* da componente k (Veja a figura 4.2).

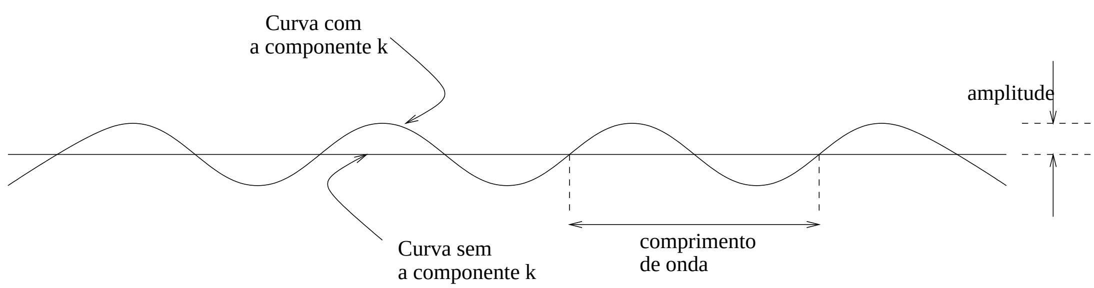


Figura 4.2: Amplitude e comprimento de onda.

Na verdade, o comprimento de onda das oscilações devidos a uma determinada componente varia conforme a velocidade da curva em cada ponto.

A velocidade escalar $|v(t)|$ da curva precisa ser constante, $|v(t)| = \bar{v}$ (ou quase) para que o comprimento de onda λ_k da fórmula (4.3) seja bem definido. Nesse caso podemos escrever

$$\lambda_k = \frac{T\bar{v}}{k}$$

Em particular, na parametrização natural ($T = L$), o comprimento de onda será $\lambda_k = L/k = T/k$.

Note que esta condição de velocidade constante $|v(t)|$ equivale a dizer que o hodógrafo da curva é o círculo de raio $\bar{v}$ e centro na origem, com uma parametrização arbitrária.

Uma questão teórica interessante é se existem curvas *trigonométricas* (com banda limitada) de velocidade constante, além das triviais (um círculo percorrido várias vezes). Não conseguimos resolver esta questão de maneira exata. Porém, verificamos empiricamente que existem curvas trigonométricas de velocidade quase constante, com formas suficientemente variadas para representar formas arbitrárias.

4.2.3 Filtros isotrópicos

Um filtro de curvas é *isotrópico* se satisfaz $\mathcal{F}(\mathcal{R}_\theta c) = \mathcal{R}_\theta(\mathcal{F}c)$, onde $(\mathcal{R}_\theta c)$ é a curva c rodada de um ângulo θ. Se as componentes x e y da curva são filtradas pelo mesmo filtro linear, verifica-se que a combinação é um filtro isotrópico.

Concluímos que o efeito de todo filtro linear, isotrópico e invariante no tempo sobre uma curva trigonométrica é equivalente a multiplicar cada coeficiente de Fourier c_k da fórmula (4.2) (um vetor de $\mathbf{C} \times \mathbf{C}$) por um número escalar complexo $w_k = \alpha_k + \mathbf{i}\beta_k$; isto é, substituir cada par de coeficientes a_k, b_k (dois vetores de $\mathbf{R} \times \mathbf{R}$) da fórmula 4.1 por novos coeficientes $\alpha_k a_k - \beta_k b_k$,$\beta_k a_k + \alpha_k b_k$.

O filtro $\mathcal{F}$ é *posicionalmente invariante* se $\mathcal{F}(c + u) = (\mathcal{F}c) + u$, para todo vetor u no $\mathbf{R}^2$. É fácil ver que um filtro linear, isotrópico e invariante no tempo é posicionalmente invariante se e somente se $w_0 = 1$.

4.2.4 Filtragem gaussiana de curvas

Vamos considerar a filtragem da curva $c(t)$ por um filtro gaussiano de duração característica σ, ou seja, calcular a convolução

$$c(t) * G_\sigma(t) = \int_{-\infty}^{\infty} c(t - \tau) G_\sigma(\tau) d\tau$$

onde

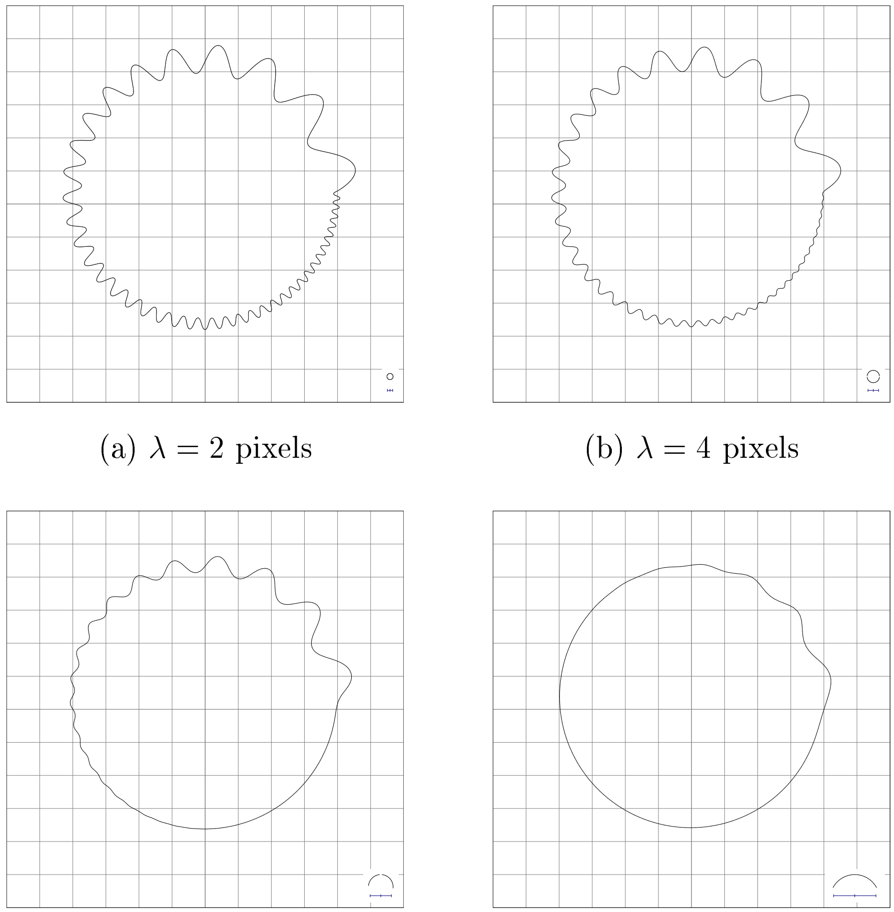
$$G_\sigma(\tau) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\tau^2}{2\sigma^2}\right)$$

Como vimos no capítulo 3, o efeito da filtragem é multiplicar a amplitude c_k da componente de freqüência k pelo fator $w_k = \exp(-(k\sigma\pi/T)^2)$.

Suponha que a curva $c(t)$ tem comprimento L, período T, e velocidade constante $v = L/T$. A duração característica σ determina o tamanho dos detalhes da curva que são removidos e/ou suavizados pelo filtro gaussiano, segundo a fórmula

$$\lambda = v\sigma = \frac{L\sigma}{T}$$

Veja a figura 4.3. O parâmetro λ é dito *comprimento característico* ou *escala de filtragem* do filtro.



(a) $\lambda = 2$ pixels (b) $\lambda = 4$ pixels

(c) $\lambda = 8$ pixels (d) $\lambda = 16$ pixels

Figura 4.3: Filtragem paramétrica gaussiana de uma curva em várias escalas de λ. Cada quadrado da grade tem 25 pixels de lado. A barra no canto inferior esquerdo tem comprimento 2λ.

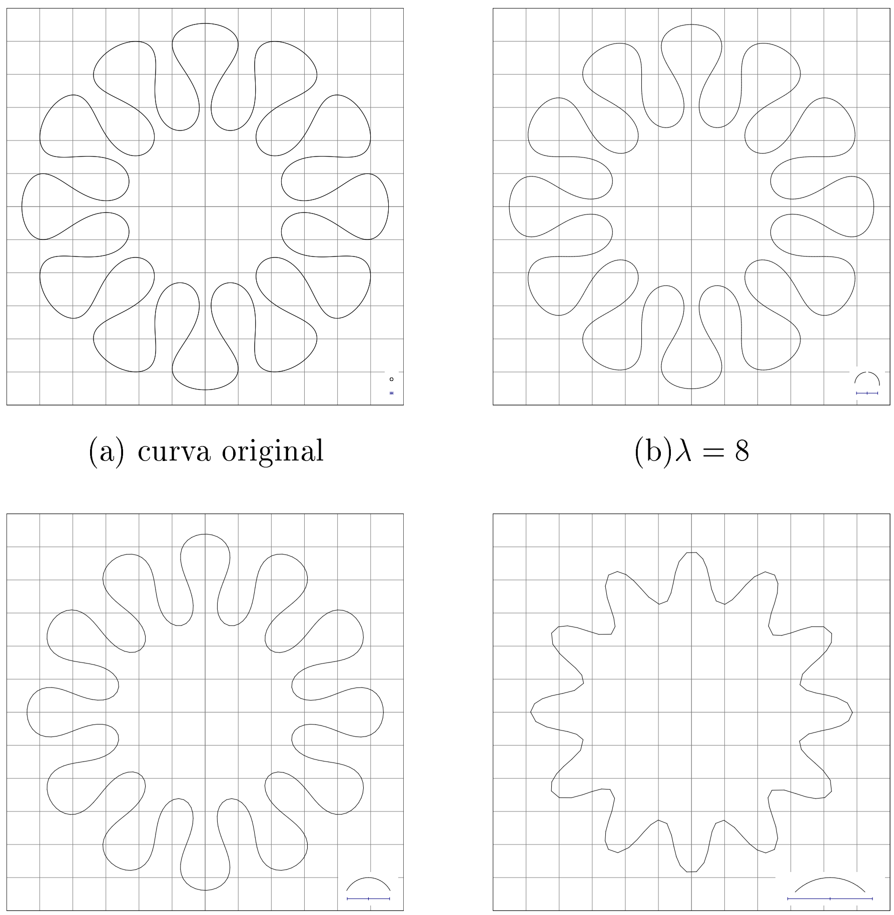
Informalmente, pequenas oscilações num contorno suave, com comprimento de onda λ_k próximo a $\lambda = L\sigma/T$, têm freqüência $k = L/\lambda_k \approx T/\sigma$, e portanto (conforme visto na seção 3.8.3) têm sua amplitude reduzida por um fator

$$\exp\left(-\left(\frac{T}{\sigma}\frac{\sigma\pi}{T}\right)^2\right) = \exp(-\pi^2) \approx 10^{-4}$$

Uma vez que o parâmetro λ tem significado geométrico claro, em geral o filtro gaussiano de curvas é especificado por esse parâmetro, sendo então a duração característica σ calculada por $\sigma = \lambda T/L$. Observe que σ depende de L e T, e portanto seu valor será diferente para cada curva, para uma mesma escala de filtragem λ.

4.3 Limitações da filtragem paramétrica

A abordagem descrita acima para filtragem de curvas, em que as coordenadas são vistas como sinais temporais, tem vários problemas sérios. Em primeiro lugar, mudanças na forma da curva mudam seu comprimento, e portanto alteram sua parametrização. Logo, quando removemos as componentes de freqüência k maior que um certo k_{max}, nem sempre estamos removendo todos os detalhes da forma com tamanho menor que L/k_{max}. Como podemos ver na figura 4.4, tal operação poderá inclusive resultar numa curva com detalhes menores que os da curva original.



(a) curva original

(b)$\lambda = 8$

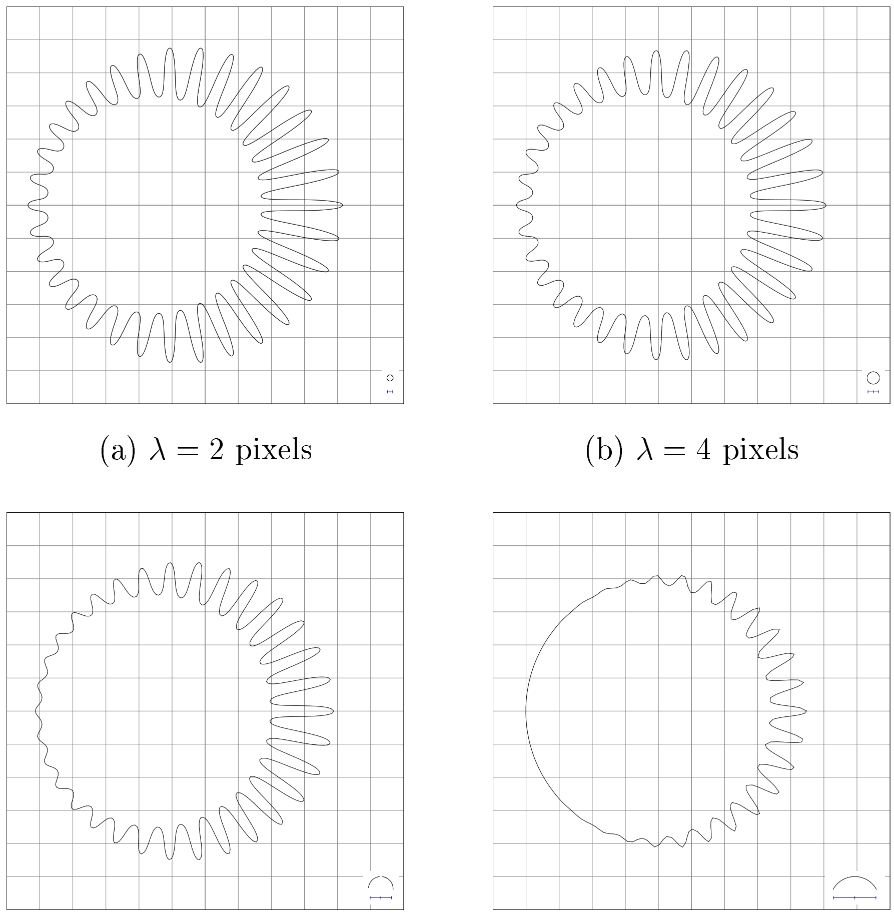
(c)$\lambda = 16$

(d)$\lambda = 32$

Figura 4.4: curva original (a) e versões filtradas da mesma (b),(c) e (d). Cada quadrado da grade tem 25 pixels de lado. A barra no canto inferior esquerdo tem comprimento 2λ.

Note que, nas protuberâncias da curva, a filtragem faz o raio de curvatura diminuir em vez de aumentar. A explicação para este efeito paradoxal é que a filtragem diminui a velocidade da curva, e portanto diminui os comprimentos de onda λ_k das componentes que sobrevivem.

Outra manifestação mais sutil deste problema é que oscilações nos contornos de amplitude maior são removidas com menos eficiência. Veja a figura 4.5.



(a) $\lambda = 2$ pixels

(b) $\lambda = 4$ pixels

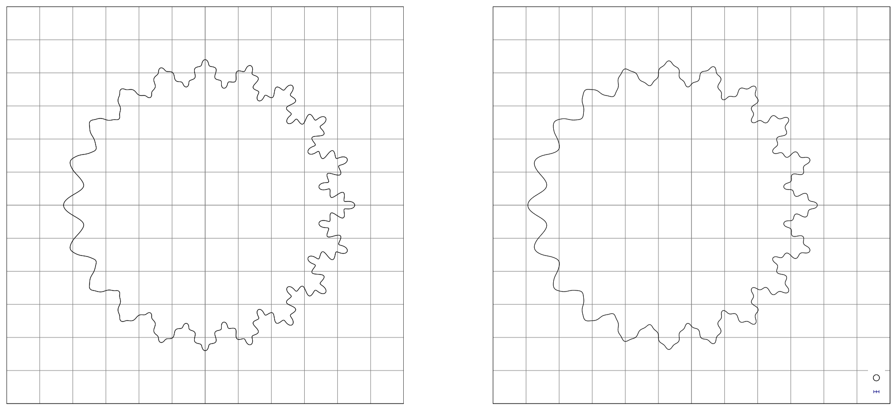
(c) $\lambda = 8$ pixels

(d) $\lambda = 16$ pixels

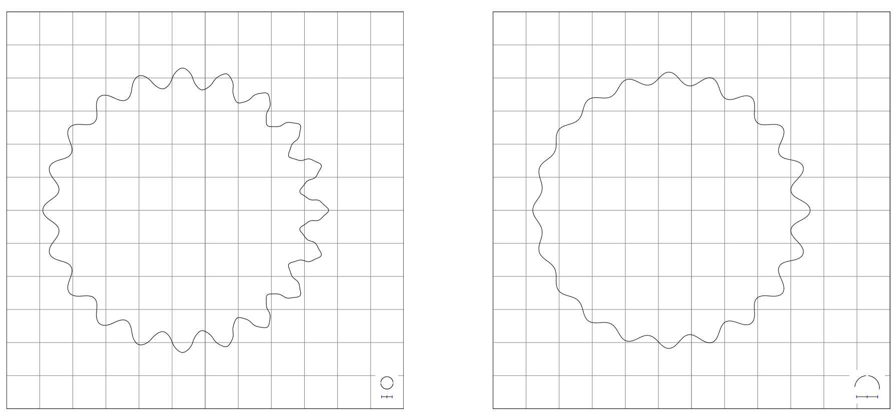
Figura 4.5: Filtragem paramétrica de uma curva com detalhes de diferentes amplitudes e mesmo comprimento aparente de onda. Cada quadrado da grade tem 25 pixels de lado. A barra no canto inferior esquerdo tem comprimento 2λ.

Lembramos que a filtragem de sinais temporais diminui a componente de freqüência k por um fator w_k, independente de sua amplitude. Entretanto, no caso de curvas, a presença de uma oscilação com comprimento de onda aparente λ_k aumenta o comprimento da curva, e, deste modo, diminui sua freqüência efetiva para fins de filtragem. Portanto, ondas de grande amplitude serão removidas de modo menos eficaz do que ondas pequenas de mesma freqüência aparente.

Outro problema relacionado é que o efeito do filtro linear gaussiano em detalhes com tamanho próximo a λ depende da presença ou ausência de detalhes de tamanho menor que λ. Especificamente, partes da curva que são mais afetadas por ruído serão menos filtradas do que o resto da curva. Veja a figura 4.6.



(a) curva original (b)$\lambda = 2$ pixels

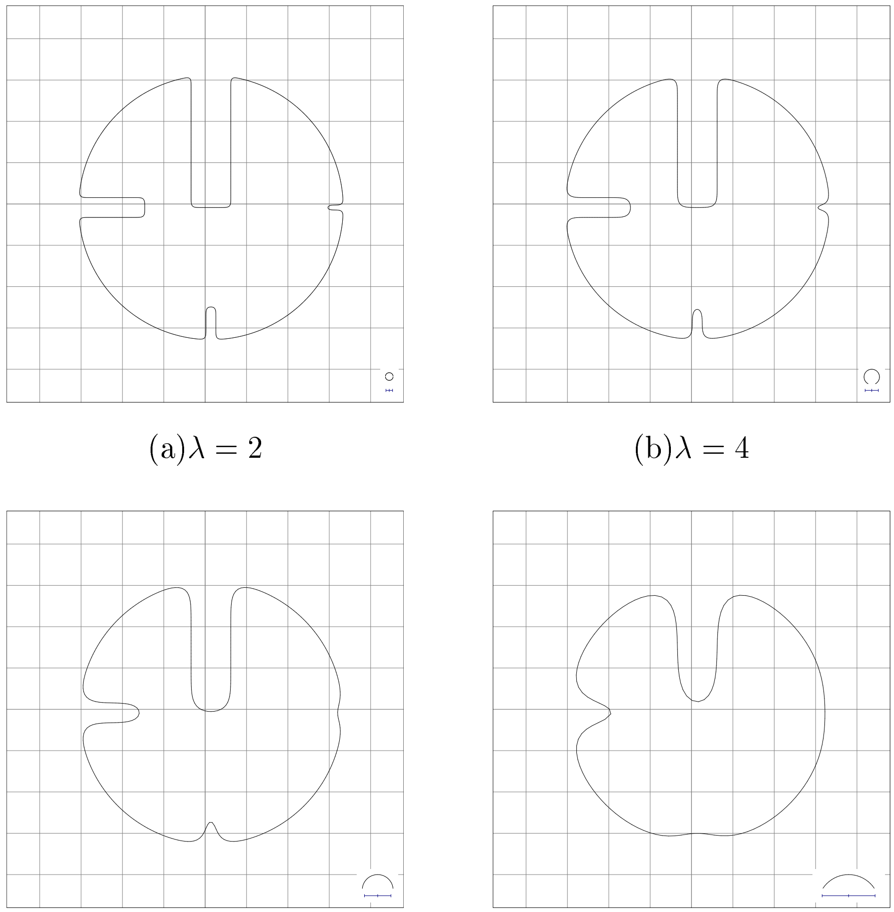


(c)$\lambda = 4$ pixels (d)$\lambda = 8$ pixels

Figura 4.6: Filtragem paramétrica de um contorno com distribuição não uniforme de ruído de alta freqüência.

Note que, nessas figuras, as ondas maiores (comprimento $\lambda \approx 30$) são menos atenuadas no lado direito, onde a curva original estava contaminada por ruído em escala bem menor ($\lambda \approx 10$).

Outra manifestação deste problema é que talhos e picos agudos num contorno não serão removidos por filtragem, conforme mostrado nas figuras 4.7.



(a)$\lambda = 2$ (b)$\lambda = 4$

(c)$\lambda = 8$ (d)$\lambda = 16$

Figura 4.7: Filtragem paramétrica. O círculo tem raio 80, e os talhos têm larguras 24, 12, 6 e 3 pixels. Cada quadrado da grade tem 25 pixels de lado. A barra no canto inferior esquerdo tem comprimento 2λ.

Observe que a curva filtrada, nas figura 4.7(b) e (c), ainda tem detalhes de curvatura alta (raio menor que λ).

Todos estes problemas ocorrem porque não existe nenhuma maneira natural de decompor uma curva na "soma" de curvas com diferentes escalas de detalhes análoga à decomposição de Fourier para sinais temporais. Sem isso, não é possível desenvolver uma teoria consistente de filtragem linear de curvas.

4.4 Suavização por evolução contínua

Uma alternativa popular para definir o conceito de filtragem de curvas é defini-lo como o resultado de uma *evolução contínua*, onde cada trecho da curva original é gradualmente suavizado por um processo local, que redistribui a curvatura dos trechos mais curvos para os trechos vizinhos mais retos.

Este modelo é justificado por analogia com processo de filtragem de um sinal por difusão (seção 3.9.1). Mais precisamente, sob este ponto de vista, a filtragem é modelada pelo processo de *difusão de curvatura* [32]. Este processo define uma família de curvas $c(t, \tau)$ onde t é o parâmetro da curva, como na seção anterior, e τ é o índice da curva na família (que pode ser considerado um instante de tempo). Por definição, $c(t, 0)$ é a curva original e $c(t, \tau)$ para $\tau > 0$ são as versões filtradas da mesma. O processo evolutivo é definido pela equação:

$$\frac{\partial c}{\partial \tau}(t, \tau) = \vec{\kappa}_c(t, \tau) = \frac{(c^{\bullet}(t, \tau) \ \times \ c^{\bullet\bullet}(t, \tau)) \times c^{\bullet}(t, \tau)}{|c^{\bullet}(t, \tau)|^4} \tag{4.4}$$

Este processo é uma versão não linear da equação padrão de difusão (3.15), em que a segunda derivada $x^{\bullet\bullet} = \partial^2 x / \partial t^2$ (um conceito paramétrico) é substituída pelo vetor curvatura $\vec{\kappa}_c$ (um conceito geométrico, independente da parametrização).

Este método alivia um pouco os problemas descritos na seção 4.3, pois equivale a fazer a reparametrização para velocidade constante de forma contínua e simultaneamente com a suavização. Entretanto, ele não resolve completamente os problemas. Para certas curvas, a equação de difusão de curvatura, assim como a filtragem paramétrica, pode aumentar temporariamente a curvatura de certos pontos. Em particular, isso acontece se a curva incluir um trecho de raio constante.

Além disso, a filtragem por difusão de curvatura também sofre do problema ilustrado na figura 4.6, ou seja, a presença de ruído de freqüência alta interfere na filtragem de componentes de freqüência mais baixa.

Finalmente, continua ocorrendo o problema ilustrado na figura 4.7: O grau de filtragem necessária para eliminar certos detalhes de alta curvatura pode destruir detalhes significativos de curvatura bem menor.

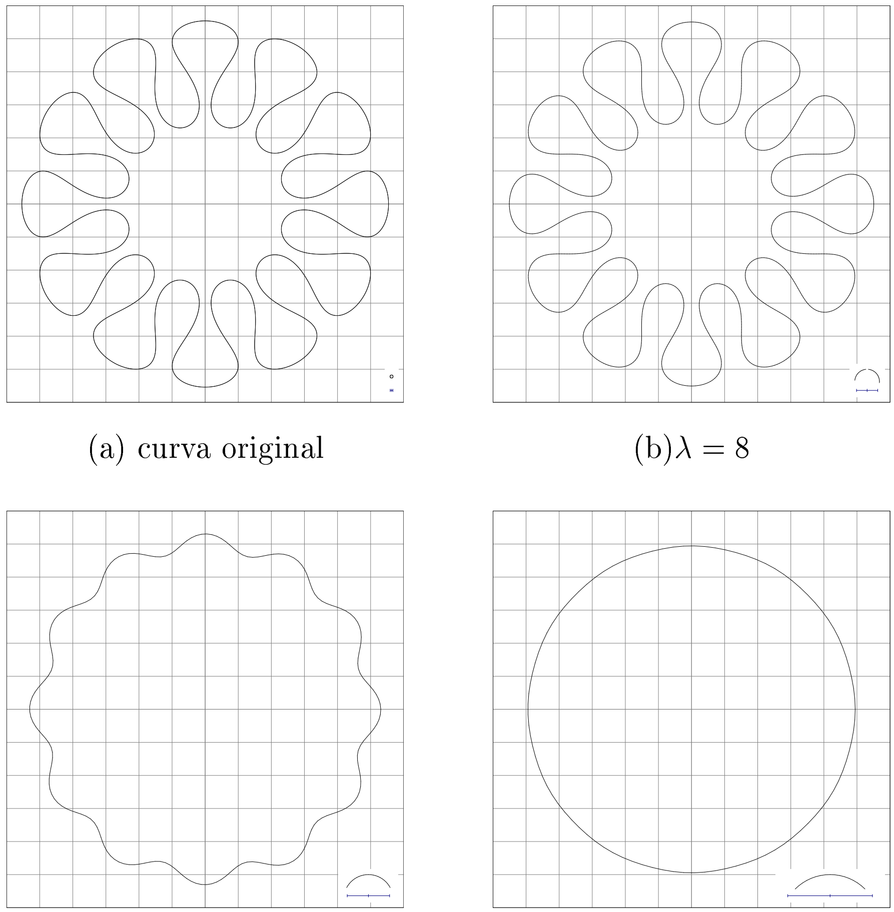
Conceitualmente, outra desvantagem deste método é que a filtragem deixa de ser uma transformação idempotente. Ou seja, a filtragem de uma curva sempre altera sua forma, mesmo quando a curva original já foi filtrada. Portanto, o conjunto das curvas filtradas não pode ser caracterizado "a priori".

4.5 Filtragem geométrica

Para corrigir o problema da reparametrização, é necessário que o fator w_k que multiplica a componente de freqüência k leve em conta a mudança no comprimento da curva, e portanto no valor λ_k, que será causada pelo processo de filtragem.

Em outras palavras, a curva original deve ser parametrizada de tal modo que, a filtragem resulte numa curva de velocidade quase constante. Desta forma, após a filtragem, cada componente de freqüência k da curva terá um comprimento bem definido λ_k. Chamamos este processo de *filtragem geométrica.*

Esta precaução elimina vários problemas de filtragem paramétrica mencionados na seção 4.4. Em primeiro lugar, com a filtragem geométrica, a curva filtrada geralmente não tem detalhes de tamanho muito menor que λ. Compare por exemplo a figura 4.8 (filtragem geométrica) com a figura 4.4 (filtragem paramétrica).



(a) curva original

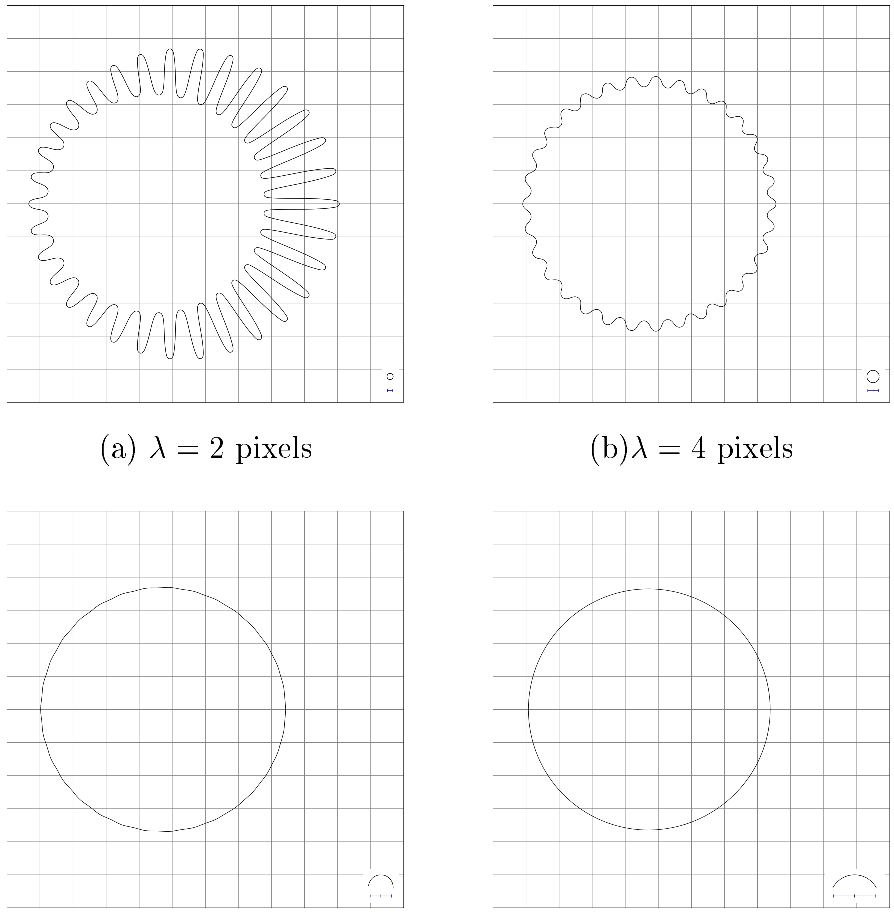
(b)$\lambda = 8$

(c)$\lambda = 16$

(d)$\lambda = 32$

Figura 4.8: Filtragem geométrica: curva original (a) e filtrada em diferentes escalas (b),(c) e (d). Cada quadrado da grade tem 25 pixels de lado. A barra no canto inferior esquerdo tem comprimento 2λ.

Além disso, detalhes de largura comparável a λ são eliminados de maneira mais uniforme, independentemente da sua amplitude; compare por exemplo as figuras 4.5 e 4.9.



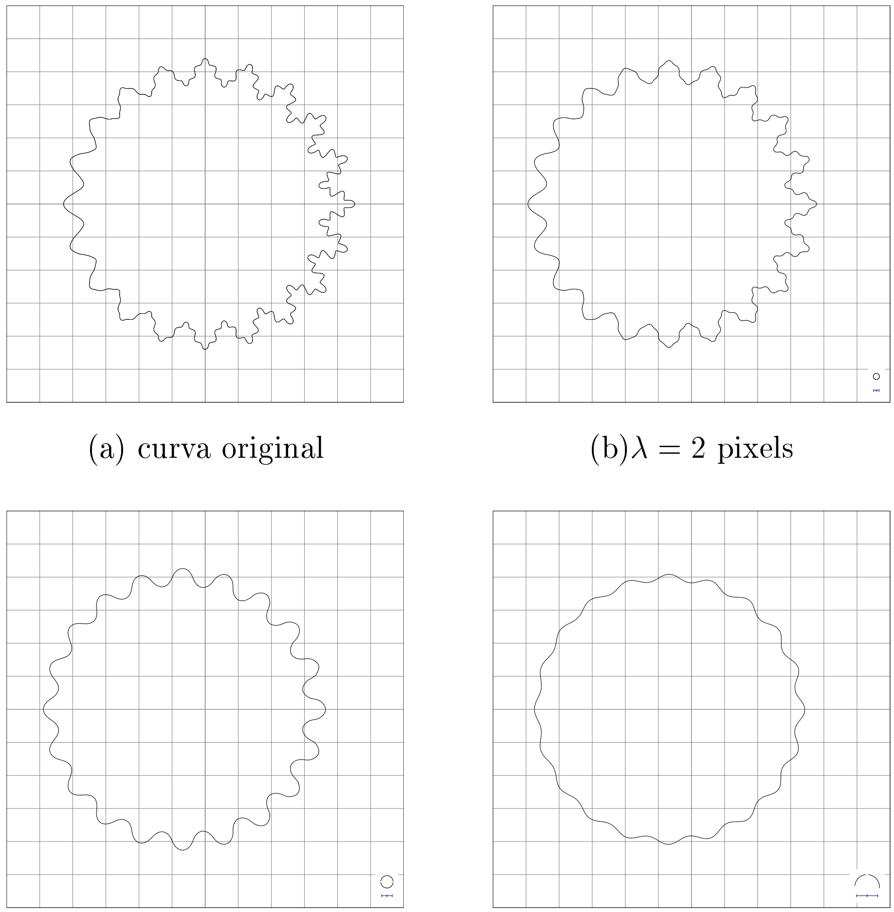
(a) $\lambda = 2$ pixels (b)$\lambda = 4$ pixels

(c)$\lambda = 8$ pixels (d)$\lambda = 16$ pixels

Figura 4.9: filtragem geométrica de uma curva com detalhes de diferentes amplitudes e mesmo comprimento aparente de onda. Cada quadrado da grade tem 25 pixels de lado. A barra no canto inferior esquerdo tem comprimento 2λ.

Observe-se que, com filtragem geométrica, ondas de comprimento aparente próximo a λ são eliminadas simultaneamente qualquer que seja sua amplitude.

Observamos igualmente que, a presença de ruído em escalas pequenas em algumas partes do contorno não terá efeito na filtragem de componentes com comprimento de onda maior. Compare a figura 4.10 (filtragem geométrica) com a figura 4.6 (filtragem paramétrica).

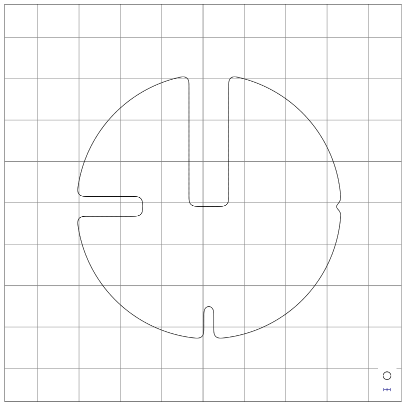


(a) curva original (b)$\lambda = 2$ pixels

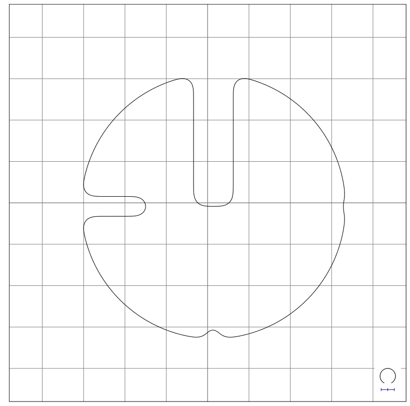
(c)$\lambda = 4$ pixels (d)$\lambda = 8$ pixels

Figura 4.10: Filtragem geométrica com distribuição não uniforme de ruído. Cada quadrado da grade tem 25 pixels de lado. A barra no canto inferior esquerdo tem comprimento 2λ.

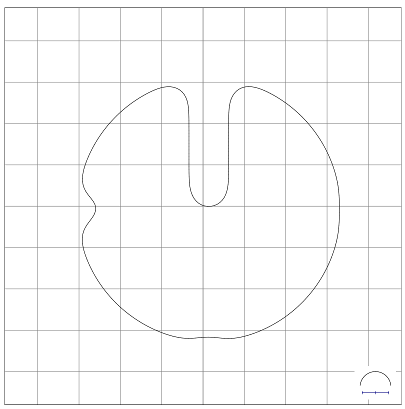
Finalmente, no exemplo da figura 4.11, observamos que com a filtragem geométrica, cada talho desaparece — exceto por uma pequena reentrância — assim que o comprimento característico λ é comparável a largura do talho. (A relação será vista na seção 4.7.) Compare a figura 4.11 com a figura 4.7.



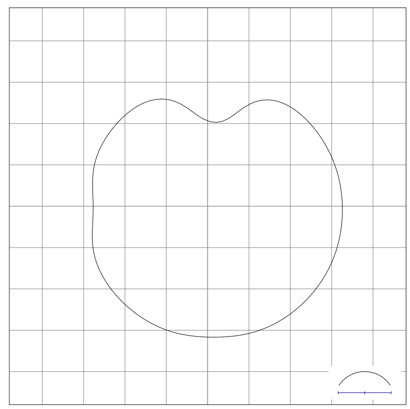
(a)$\lambda = 2$



(b)$\lambda = 4$



(c)$\lambda = 8$



(d)$\lambda = 16$

Figura 4.11: Filtragem geométrica. O círculo tem raio 80, e os talhos têm larguras 24, 12, 6 e 3 pixels. Cada quadrado da grade tem 25 pixels de lado. A barra no canto inferior esquerdo tem comprimento 2λ.

4.6 Algoritmo de filtragem geométrica

A filtragem geométrica exige que a reparametrização desejada da curva original seja calculada por um algoritmo numérico iterativo que descreveremos a seguir.

O algoritmo parte de uma curva c, que pode já ter sido filtrada geometricamente na escala λ_{old}, e retorna uma cópia de c processada com um filtro gaussiano de escala $\lambda_{new} > \lambda_{old}$. A curva de entrada é representada por amostras $c_1, c_2 .. c_n$, que são implicitamente estendidas pela identidade $c_{i+kn} = c_i$ para todo i e k; em particular $c_0 = c_n$. A curva filtrada resultante é representada por amostras $d_1, d_2 .. d_m$, igualmente espaçadas ao longo da curva, com passo de amostragem que satisfaz o critério de Nyquist como discutido na seção 3.4.

No decorrer do algoritmo, estaremos construindo iterativamente uma reparametrização da curva de entrada. A reparametrização é representada por tempos $t_1 .. t_n$, associados as amostras $c_1, c_2 .. c_n$, sendo que t_n é o período da curva (isto é $t_{i+kn} = t_i + kt_n$ para todo i e k). A curva reparametrizada $\tilde{c}$ por definição satisfaz $\tilde{c}(t_i) = c_i$, para todo i. A reparametrização é tal que a filtragem *paramétrica* da curva $\tilde{c}$ resulta numa curva d de velocidade constante $|d^{\bullet} = 1|$. Portanto, ao fim da iteração, o período t_n da curva $\tilde{c}$ será igual ao comprimento da curva filtrada d; e cada termo t_i será o comprimento da curva filtrada d, desde o ponto inicial $d(0)$ até o ponto $d(t_i)$ que corresponde à amostra original c_i.

Algoritmo 2 Filtragem geométrica

Entrada:

- *as escalas de filtragem* λ_{old} *e* λ_{new} *com* $\lambda_{new} > \lambda_{old} \geq 0$*;*
- *as amostras* $c_1, c_2 .. c_n$ *da curva original (filtrada na escala* λ_{old}*), igualmente espaçadas;*
- *duas tolerâncias* ε_{step}*,* ε_{tot}*.*

Saída:

- *As amostras igualmente espaçadas,* $d_1, d_2 .. d_m$ *da curva filtrada na escala* λ_{new}*.*
- *Os parâmetros* $t_1 .. t_n$ *que definem a reparametrização da curva* c *usada na filtragem.*

Passos:

1. Calcule os tempos iniciais $t_1^{(0)}, t_2^{(0)}, .. t_n^{(0)}$ para as amostras $c_1, c_2, .. c_n$ sendo que $t_i^{(0)}$ é o comprimento da curva entre $c_0 (= c_n)$ e c_i, para $i = 1, .. n$.
2. Calcule a escala de filtragem a usar, $\lambda_{inc} \leftarrow \sqrt{\lambda_{new}^2 - \lambda_{old}^2}$.
3. Faça $k \leftarrow 0$.
4. Repita
   - 4.1. Faça $L \leftarrow t_n^{(k)}$, $k \leftarrow k + 1$.
   - 4.2. Estime o número de amostras necessárias na curva filtrada pelo critério de Nyquist: $m \leftarrow 4\lceil L/\lambda_{new} \rceil$.
   - 4.3. Seja a curva $\tilde{c}$ definida pelas amostras $c_1, .. c_n$ com período L e tempos $t_1^{(k-1)}, .. t_n^{(k-1)}$. Calcule a convolução da curva $\tilde{c}$, com o núcleo do filtro gaussiano G_σ de duração característica $\sigma = \lambda_{inc}$, e reamostre a curva resultante em m instantes igualmente espaçados $t'_1, .. t'_m$ no intervalo $[0,\ L]$ (isto é, $t_j \leftarrow jL/m$ para $j = 1 .. m$).

4.4. Recalcule os tempos

$$t_i^{(k)} = \int_0^{t_i^{(k-1)}} |d^{\bullet}(\tau)| \, d\tau$$

para $i = 1..n$.

até que a diferença $\left|t_n{}^{(k)} - t_n{}^{(k-1)}\right| < \varepsilon_{tot}\lambda_{new}$ e

$$\left|(t_{i+1}{}^{(k)} - t_i{}^{(k)}) - (t_{i+1}{}^{(k-1)} - t_i{}^{(k-1)})\right| < \varepsilon_{step}\lambda_{new}; \text{ para todo } i$$

(ou seja, até que a reparametrização se estabilize).

5. Retorne as amostras $d_1, .. d_m$ da curva filtrada, e os tempos $t_1^{(k)}, .. t_n^{(k)}$ da curva original reparametrizada.

A filtragem no passo 4.3 é feita por convolução da curva original com o núcleo do filtro gaussiano, G_σ (seção 3.8.1). Não trabalhamos no espaço de freqüências (representação de Fourier), porque o comprimento característico λ dos filtros usados é um múltiplo pequeno do intervalo de amostragem da curva inicial, e portanto a convolução é mais rápida que a transformada rápida de Fourier (FFT).

Além disso, como vimos na seção 3.8.3, a filtragem por convolução pode ser efetuada mesmo quando as amostras não são igualmente espaçadas. Note que este é o caso dos contornos extraídos das imagens pelo algoritmo descrito na seção 2.2; e, de qualquer forma, os intervalos entre as amostras são alterados de forma irregular pelo próprio algoritmo de filtragem geométrica. Vale notar que, no exemplo da figura 4.11, a reparametrização eventualmente usada pelo algoritmo é tal que todo o talho fica comprimido num intervalo de tempo da ordem de λ. Para filtrar corretamente esta curva via FFT seria necessário reamostrá-la com passo pequeno o bastante para representar precisamente a área do talho, o que implicaria num custo bastante elevado.

Na nossa implementação, supomos implicitamente que a curva de entrada é uma poligonal com vértices $c_1, c_2, .. c_n$, e efetuamos a filtragem pelo algoritmo 1 do capítulo 3.

Para maior precisão, seria desejável supor que a curva original é uma interpolação mais suave das amostras dadas. Entretanto, para isso seria necessário utilizar uma fórmula de filtragem ainda mais complicada que a fórmula (3.18), envolvendo cinco ou mais amostras consecutivas. Além disso, a decomposição recursiva usada no algoritmo 1 ficaria mais complicada.

O uso de interpolação linear por partes introduz um erro pequeno, mas sistemático na filtragem, pois a curva de entrada supostamente é suave, e a poligonal reintroduz componentes de freqüências altas, além de deslocar a posição média da curva na direção do centro de curvatura. Entretanto, o deslocamento é proporcional ao quadrado do passo de amostragem; portanto como usamos um passo bem menor que o limite de Nyquist, o erro decorrente da aproximação poligonal é desprezível.

4.7 Convergência da filtragem geométrica

Uma vez que a filtragem geométrica foi definida de maneira implícita, cabe questionar se o resultado sempre existe, e se ele é único. Em particular, observa-se que a filtragem diminui o comprimento de onda das componentes, e portanto faz com que as mesmas sejam ainda mais atenuadas na iteração seguinte.

4.7.1 Filtragem geométrica de um círculo

Para responder em parte a essa questão, vamos considerar o caso em que a curva c a ser filtrada é um círculo de raio R. Por simetria, é evidente que todas as reparametrizações da curva c calculadas pelo algoritmo 4.6 terão velocidade constante, e portanto todas as versões filtradas da mesma serão círculos percorridos com velocidade uniforme. Nesse caso, a curva $\tilde{c}$ pode ser escrita como uma série finita de Fourier, com freqüência máxima $k = 1$ e amplitude $|c_1| = R$. Se T é o período da reparametrização corrente, a velocidade da curva será $2\pi R/T$.

Conforme visto na seção 3.8.3, a filtragem gaussiana reduz a amplitude dos termos de freqüência 1 pelo fator

$$w_1 = \exp\left(-\left(\frac{\sigma\pi}{T}\right)^2\right)$$

Portanto, a curva filtrada c' será um círculo de raio

$$R' = Rw_1 = R\exp\left(-\left(\frac{\sigma\pi}{T}\right)^2\right) \tag{4.5}$$

A curva $\tilde{c}$ é então reparametrizada em termos do comprimento da curva filtrada c'. Segue-se que seu novo período T será o perímetro $2\pi R'$ de c'. Substituindo na fórmula (4.5) temos que o algoritmo varia o raio R' do círculo filtrado pela iteração

$$R' \leftarrow R\exp\left(-\left(\frac{\sigma\pi}{2\pi R'}\right)^2\right) = R\exp\left(-\left(\frac{\sigma}{2R'}\right)^2\right) \tag{4.6}$$

É necessário portanto verificar que esta iteração não reduz toda a curva a um ponto, mas converge para uma curva de comprimento maior que zero.

Para estudar a convergência da iteração (4.6) convém reescrevê-la na forma

$$\frac{R'}{R} \leftarrow \exp\left(-\frac{1}{4}\frac{\left(\frac{\lambda}{R}\right)^2}{\left(\frac{R'}{R}\right)^2}\right)$$

ou, denotando R'/R por ω, e λ/R por α,

$$\omega \leftarrow \exp\left(-\frac{\alpha^2}{4\omega^2}\right) \tag{4.7}$$

A figura 4.12 mostra o gráfico do lado direito da iteração (4.7), $f(\omega) = \exp(-\alpha^2/(4\omega^2))$, para diversos valores da razão $\alpha = \lambda/R$:

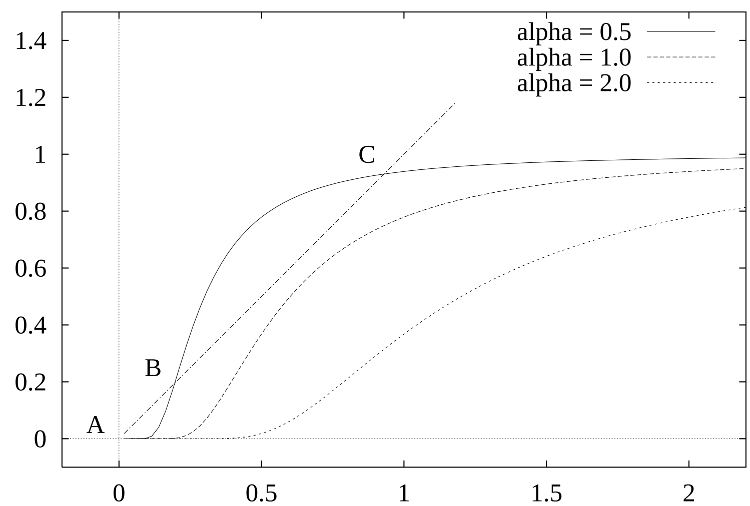


Figura 4.12: Gráfico de $f(\omega) = \exp(-\alpha^2/(4\omega^2))$.

As soluções da iteração (4.7) correspondem às interseções do gráfico de $f(\omega)$ com a reta $r(\omega) = \omega$. Como pode ser visto na figura, a iteração possui sempre um ponto fixo trivial $\omega = 0$ (ou seja, $R' = 0$) , correspondente ao ponto A da figura. Além disso, se α é menor que um certo valor crítico $\alpha^\star$ há outras duas soluções, correspondentes aos pontos B e C. A solução B é instável, do ponto de vista do algoritmo, que portanto convergirá para a solução A ou C, conforme a estimativa inicial para ω esteja à esquerda ou à direita de B. Em particular, se a estimativa inicial for $w = 1$ (isto é $R' = R$), verifica-se que a iteração converge rapidamente para a solução C.

4.7.2 Raio crítico

O valor crítico da razão $\alpha = \lambda/R$ é $\alpha^\star = \sqrt{2/e} \approx 0.858$. Portanto, círculos com raio R menor que $\lambda/\alpha^\star \approx 1.2\lambda$ são reduzidos a um ponto pelo algoritmo de filtragem geométrica, enquanto que círculos de raio maior que $\lambda/\alpha^\star$ são reduzidos por um fator maior ou igual a $\omega^\star = 1/\sqrt{e} \approx 0.607$.

Com base neste caso particular, pode-se intuir que a filtragem de curvas mais gerais exibe um comportamento similar. Verifica-se que detalhes (protuberâncias ou reentrâncias) na curva com raio $< 1.2\lambda$ são eliminadas, enquanto que detalhes com raio $> 1.2\lambda$ são reduzidos para 60% do seu tamanho original, se tanto. Em particular, como visto na figura 4.11, talhos de largura $< 2.4\lambda$ são eliminados, qualquer que seja a profundidade, enquanto que talhos de largura $> 2.4\lambda$ praticamente não são alterados.

4.7.3 Soluções múltiplas

Na verdade esta análise também revela que a solução pode não ser única. Dependendo da parametrização inicial, certos detalhes podem ser eliminados ou não, e nos dois casos a parametrização natural do resultado pode ser consistente com a parametrização inicial escolhida.

Ou seja se a curva original c têm dois pontos X e Y, cuja distância em linha reta é da ordem de λ, existe uma solução estável em que a curva filtrada $\tilde{c}$ pula diretamente de X para Y sem acompanhar a parte de curva entre estes dois pontos. Esta solução não será normalmente encontrada pelo algoritmo uma vez que a estimativa inicial para a solução é a própria curva c.

A existência de várias soluções estáveis potenciais implica que é difícil dar uma caracterização direta da curva filtrada. Ela é uma desvantagem da filtragem geométrica.

4.7.4 Velocidade de convergência

Infelizmente, não podemos limitar o número de iterações executadas pelo algoritmo 2, pois esse parâmetro depende do tamanho das irregularidades a serem removidas. Se o contorno possui um talho profundo e estreito (como na figura 4.11), cada iteração reduz o comprimento do talho de uma distância da ordem da escala de filtragem λ; portanto o número de iterações é proporcional a profundidade do talho dividida por λ.

4.7.5 Correspondência entre as curvas

A reparametrização da curva c no decorrer do algoritmo 2 significa que pontos correspondentes no contorno original e no contorno filtrado terão valores diferentes do parâmetro t. Esta discrepância é inconveniente, porque, para o processamento multi-escala, que descreveremos no capítulo 10, é necessário conhecer a correspondência entre os pontos destas duas curvas.

Para contornar esta dificuldade, associamos a cada amostra c_i de cada contorno filtrado um *rótulo* τ_i, que é o comprimento do contorno original (não filtrado) desde o ponto que corresponde à amostra c_0 até o ponto que corresponde à amostra c_i. Quando aplicamos o algoritmo de filtragem à curva c, obtemos novas amostras $d_1, .. \, d_m$ com tempos igualmente espaçados $t'_1, .. \, t'_m$, e novos tempos $t_1, .. \, t_n$ para as amostras $c_1, .. \, c_n$ que indicam a posição das mesmas na curva filtrada. Por interpolação dos valores t'_j.. na função linear por partes definida pelos pares (t_i, τ_i) obtemos, para cada amostra d_j, o rótulo τ'_j correspondente.

Uma vez determinados os rótulos τ_i para todas as escalas de filtragem, podemos mapear um ponto de uma versão filtrada para outra. Por interpolação direta calculamos o rótulo τ do ponto em questão, e por interpolação inversa encontramos o ponto correspondente na outra curva.

A figura 4.13 mostra a correspondência entre os pontos de uma curva original e de sua versão filtrada com $\lambda = 64$ pixels.

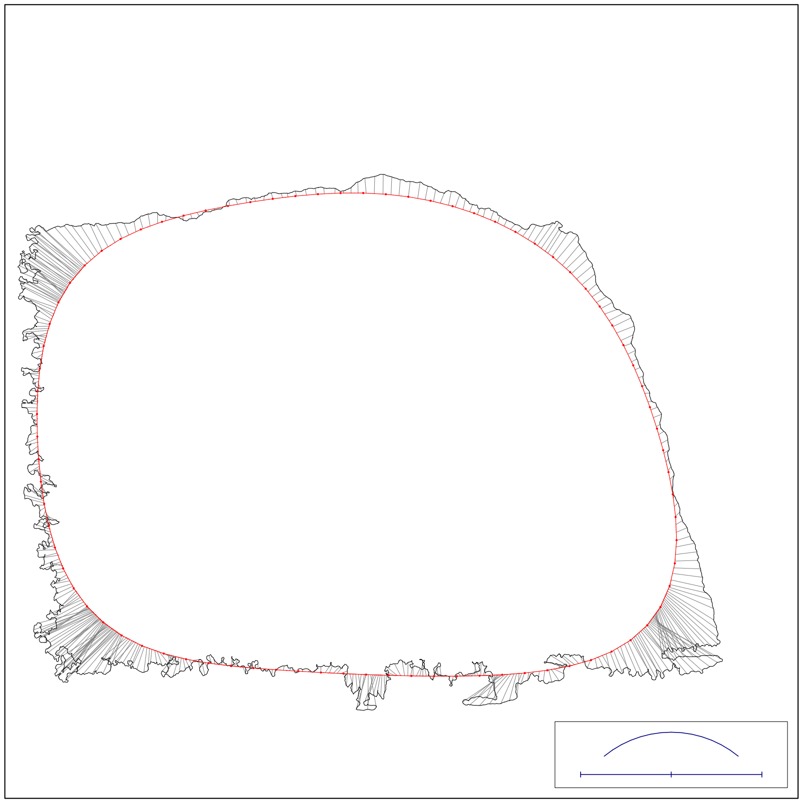


Figura 4.13: Curva original (mais externa) e sua versão filtrada (curva interna).

Capítulo 5

Informação contida nos contornos

Para descobrir se problemas de tamanho realista (milhares de fragmentos) podem ser resolvidos considerando-se apenas os contornos dos mesmos precisamos saber quanta informação útil existe no formato de um fragmento. Por "útil" queremos dizer a informação que pode ser usada para identificar os fragmentos adjacentes ao mesmo.

Neste capítulo, descreveremos um método para determinação da deste dado, especificamente a quantidade média de informação útil contida em um pedaço de contorno de fragmento de um dado comprimento. Lembramos que dois segmentos de contornos que correspondem à mesma linha de fratura nunca são precisamente congruentes: sempre existem algumas diferenças, quer reais (por exemplo, perda de pequenas migalhas de fragmento) quer artificiais (devido a erros no processo de extração dos contornos). A informação útil contida em um pedaço de contorno é determinada pela magnitude de todos estes erros, em relação ao tamanho dos detalhes característicos que podem ser usados para identificar o parceiro de um dado segmento.

5.1 Conceitos básicos de teoria da informação

Revisaremos aqui alguns conceitos básicos de teoria da informação que serão usados mais tarde.

5.1.1 Entropia e informação

Vamos denotar por $\operatorname{avg}(X)$ e $\operatorname{var}(X)$ a média e a variância de uma variável aleatória (real ou complexa) X,

$$\operatorname{avg}(X) = \int_{\mathbf{C}} \Pr(X \approx x)\, x \, dx \tag{5.1}$$

$$\operatorname{var}(X) = \operatorname{avg}(|X - \operatorname{avg}(X)|^2) \tag{5.2}$$

A *incerteza* ou *entropia* de uma variável aleatória X, que varia sobre um domínio contínuo D, é por definição

$$H(X) = -\int_D \Pr(X \approx x) \log_2 \Pr(X \approx x)\, dx \tag{5.3}$$

onde $\Pr(X \approx x)$ é a densidade de probabilidade de X nas vizinhanças do ponto x de D.

Em particular, se X é uma variável real com distribuição gaussiana de variância $\hat{X}$, verifica-se que sua entropia é

$$H(X) = \frac{1}{2} \log_2(2\pi e \hat{X}) \tag{5.4}$$

Dizemos que uma variável complexa tem distribuição *gaussiana, simétrica de variância* $\hat{X}$ se suas partes partes real e imaginária são variáveis gaussianas independentes com a mesma variância $\hat{X}/2$.

A partir da fórmula (5.4), podemos deduzir que a entropia de tal variável é

$$H(X) = \log_2(\pi e \hat{X}) \tag{5.5}$$

5.1.2 Informação condicional

Se A e B são duas variáveis aleatórias com domínios D_A e D_B, a informação que ganhamos sobre B, quando sabemos que A tem um valor particular y, é

$$I(B \mid A = y) = H(B) - H(B \mid A = y) \tag{5.6}$$

onde $H(B \mid A = y)$ é o valor da fórmula (5.3), calculado para a distribuição de probabilidade condicional $\Pr(B \approx x \mid A = y)$.

A *informação média fornecida por A a respeito de B* é o valor esperado da grandeza $I(B \mid A = y)$, para um valor genérico de y, isto é,

$$I(B \mid A) = H(B) - H(B \mid A) \tag{5.7}$$

onde

$$H(B \mid A) = \int_{D_A} H(B \mid A = y) \Pr(A \approx y)\, dy \tag{5.8}$$

5.1.3 Informação mútua

Um caso que nos interessa particularmente é quando $A = S + N$ e $B = S + P$, onde S, N e P são variáveis complexas independentes com distribuições gaussianas simétricas e variâncias $\hat{S}$, $\hat{N}$, and $\hat{P}$. Podemos pensar no número complexo S como uma“mensagem” a partir da qual são feitas duas cópias independentes, que são corrompidas por “ruídos” N

e P. Nosso objetivo é determinar quanta informação a mensagem A dá sobre a mensagem B, na média.

Neste caso, verifica-se que A e B também têm distribuições gaussianas simétricas com variâncias $\hat{A} = \hat{S} + \hat{N}$ e $\hat{B} = \hat{S} + \hat{P}$, respectivamente. Portanto, o primeiro termo da fórmula (5.7) é simplesmente

$$H(B) = \log_2(\pi e \hat{B}) = \log_2[\pi e(\hat{S} + \hat{P})] \tag{5.9}$$

Mais ainda, verifica-se que a distribuição condicional $\Pr(S \approx x \mid A = y)$ é outra gaussiana com média $y\hat{S}/\hat{A}$ e variância $\hat{S}\hat{N}/\hat{A}$. Uma vez que P é independente de N e S, a distribuição condicional de $B = S + P$, dado $A = y$, é também uma gaussiana, com a mesma média $y\hat{S}/\hat{A}$ e variância $\hat{S}\hat{N}/\hat{A} + \hat{P}$. Observe que a variância não depende de y; de modo que a fórmula (5.8) se reduz a

$$H(B \mid A) = \log_2\left[\pi e\left(\frac{\hat{S}\hat{N}}{\hat{A}} + \hat{P}\right)\right] \tag{5.10}$$

De acordo com a fórmula (5.7), a informação dada por A sobre B é, então,

$$\begin{aligned} I(B \mid A) &= H(B) - H(B \mid A) \\ &= \log_2[\pi e(\hat{S} + \hat{P})] - \log_2\left[\pi e\left(\frac{\hat{S}\hat{N}}{\hat{A}} + \hat{P}\right)\right] \\ &= \log_2\left[\frac{\hat{S} + \hat{P}}{\frac{\hat{S}\hat{N}}{\hat{A}} + \hat{P}}\right] \\ &= \log_2\left[\frac{\hat{A}\hat{B}}{\hat{S}\hat{N} + \hat{A}\hat{P}}\right] \end{aligned} \tag{5.11}$$

5.1.4 A fórmula de Shannon-Hartley

Em particular, tomando $N = 0$ (isto é, $B = S$) obtemos a fórmula de Shannon-Hartley [16]

$$I(S \mid A) = \log_2\left[\frac{\hat{A}}{\hat{N}}\right] = \log_2\left[\frac{\hat{S} + \hat{N}}{\hat{N}}\right] \tag{5.12}$$

que dá a quantidade de informação fornecida pela mensagem corrompida A sobre a mensagem original S.

Intuitivamente, a distribuição da mensagem observada A é uma nuvem gaussiana simétrica no plano complexo, com área proporcional a $\hat{S} + \hat{N}$; enquanto que a distribuição do ruído N é uma nuvem menor com área proporcional a $\hat{N}$. Portanto podemos empacotar $O((S + N)/N)$ cópias desta última dentro da primeira, com uma quantidade fixa de sobreposição; i.e. podemos distinguir $O((S + N)/N)$ valores discretos de S, com uma confiança estabelecida. Veja a figura 5.1.

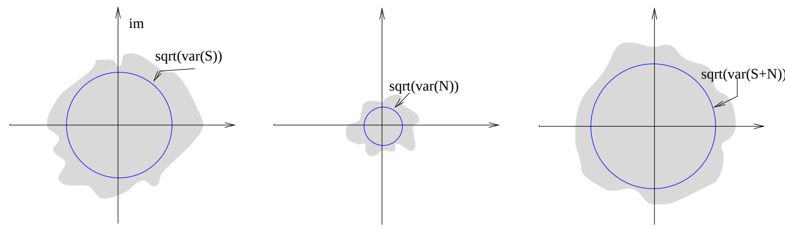


Figura 5.1: Distribuições para as variáveis complexas S (sinal original), N (ruído), e $A = S + N$ (sinal observado).

5.1.5 Informação a partir do espectro

Como visto no capítulo 4, se um sinal $s(t)$ é conhecido em apenas um conjunto finito de instantes uniformemente espaçados $t_j = j\delta$, para $0 \leq j < m$, podemos expressá-lo como uma *série discreta de Fourier*

$$s_j = s(t_j) = \sum_{k=-n}^{n} S_k \exp(\frac{2\pi \mathbf{i} k}{T} t_j) \tag{5.13}$$

onde $T = m\delta$ é o *período*, $n = \lceil m/2 \rceil$ é a *freqüência máxima*, e $\mathbf{i} = \sqrt{-1}$ é a unidade imaginária. Lembramos que os coeficientes de Fourier S_k de uma série de valores reais δ_j satisfazem $S_{-k} = S_k^*$ para todo k; além disso, S_0 é real, e S_n é real quando m é par. Portanto, temos exatamente m graus de liberdade nos coeficientes de S_k.

5.2 Interpretando curvas como sinais

Antes de aplicarmos as ferramentas da teoria da informação ao nosso problema, devemos transformar cada curva em um *sinal*, que retorna valores reais em função de um parâmetro real t. A transformação deve assegurar que segmentos de contorno adjacentes resultem em sinais parecidos, mesmo se os fragmentos tiverem sido digitalizados com orientações aleatórias.

A escolha natural para o parâmetro t é o comprimento da curva a partir de um ponto de referência. Não é válido usar as coordenadas x e y como dois sinais separados, já que elas não são independentes na parametrização pelo comprimento do arco. Não é válido tampouco usar apenas uma das duas coordenadas, pois dessa forma estaríamos descartando informação das partes da curva onde a mesma é aproximadamente paralela ao eixo correspondente.

Uma representação que é geralmente usada no reconhecimento de formas é o grafo da curvatura em função do comprimento do arco. Esta representação seria adequada para nosso propósito (tanto assim que a usamos no resto do trabalho). Contudo, uma vez que a curvatura é essencialmente uma derivada segunda, ela tende a aumentar o efeito do ruído nas escalas pequenas. Além disso, o formato do gráfico de curvatura em geral é bem diferente do formato da curva. Apesar destes problemas não afetarem a análise acima, baseada na série de Fourier, parece prudente usar uma representação que seja a mais próxima possível da curva original.

Portanto, usaremos neste capítulo uma *função de forma* derivada do segmento da curva conforme descrito abaixo. Vamos supor que o segmento de curva em questão tem comprimento L, e é dado por $n+1 = 2^k+1$ pontos amostrais $c_0, .. c_n$ no plano, igualmente espaçados ao longo da curva. A função de forma s é conceitualmente definida no intervalo $[0 \,\_\_\, L]$, e é representada por $n+1$ amostras $s_0, s_1, .. s_n$, com $s_0 = s_n = 0$, pelo seguinte procedimento recursivo:

1. se $n = 0$, retorne $s_0 = 0$.

2. seja r o índice da amostra do meio do segmento, $r = n/2$. Recursivamente converta as seqüências $c_0, .. c_r$ e $c_r, .. c_n$ em sinais $s_0, .. s_r$ e $s_r, .. s_n$.

3. seja α o ângulo entre os vetores $u = c_r - c_0$ e $v = c_n - c_r$. Some às amostras $s_0, .. s_n$ uma seqüência $g_0, .. g_n$ com $g_0 = g_n = 0$, $g_r = \alpha L/4$, e demais valores definidos por interpolação linear entre estes extremos (i.e., um pulso triangular de altura $\alpha L/4$). Retorne a seqüência $s_0, .. s_n$.

Esta transformação é completamente inversível: dado o comprimento L, o ponto c_0, a linha $c_0 c_n$, e as amostras s_i, podemos reconstruir os pontos originais c_i executando os passos do algoritmo na ordem inversa. Esta representação, ao contrário do gráfico das curvaturas, não aumenta o ruído em pequena escala, e preserva qualitativamente a forma da curva, mesmo quando esta não é o gráfico de nenhuma função. Veja figura 5.2.

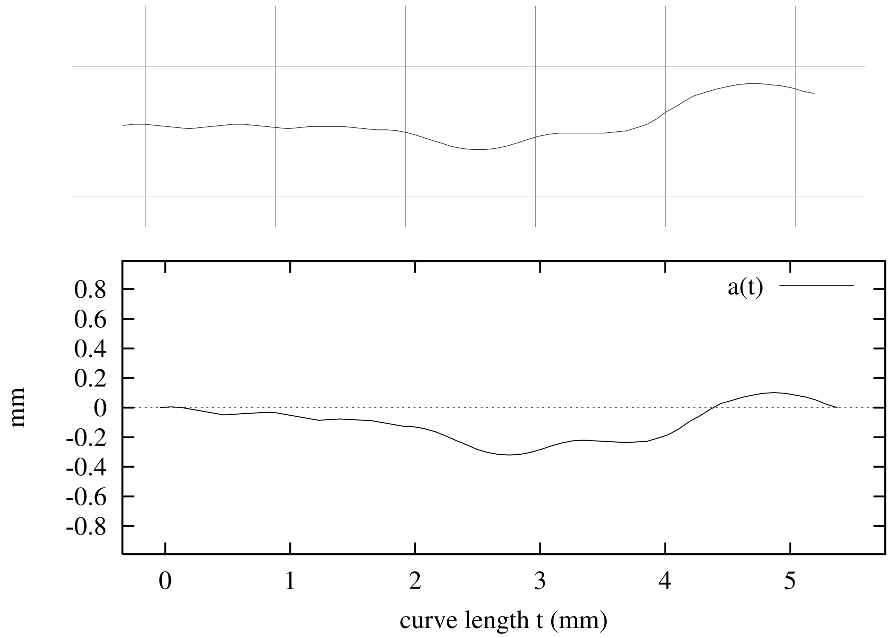


Figura 5.2: Uma curva e sua função de forma.

Um problema desta transformação é que uma perturbação local pode mudar o comprimento da curva, e portanto causar um deslocamento global da função de forma a partir do ponto onde a perturbação ocorre. Entretanto, verifica-se experimentalmente que as funções de forma de dois pedaços de contorno correspondentes, como os ilustrados nas figuras 5.3 e, 5.4 geralmente permanecem sincronizadas na maior parte dos seus comprimentos, conforme mostrado na figura 5.5.

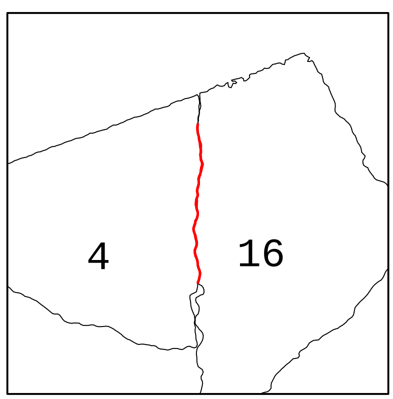


Figura 5.3: Dois fragmentos correspondentes – tamanho real.

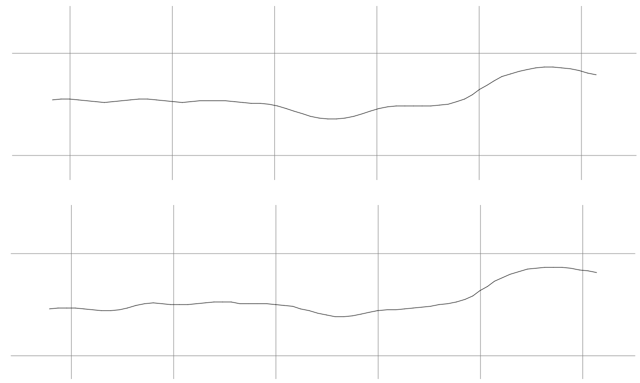


Figura 5.4: Partes correspondentes de dois fragmentos vizinhos, ampliados. As linhas da grade têm espaçamento de 1 mm. Os contornos foram digitalizados com resolução de 300dpi (0.085 mm/pixel) e suavizados por um filtro gaussiano de escala característica $\lambda = 0.152$ mm.

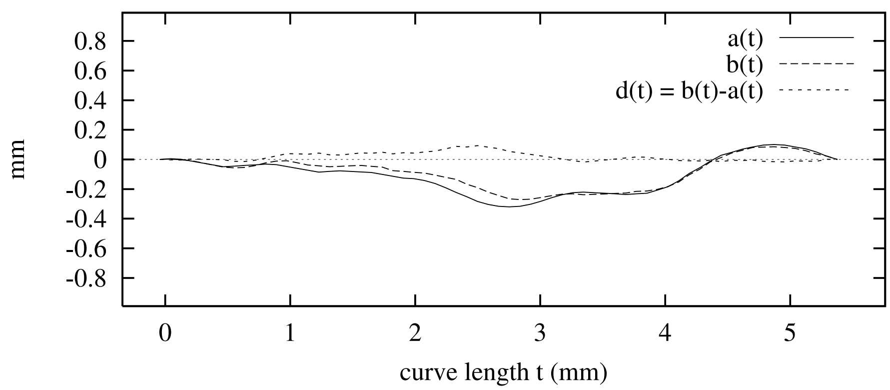


Figura 5.5: As funções de forma dos dois segmentos correspondentes mostrados na figura 5.4.

5.3 Conteúdo de informação dos contornos

A fórmula para informação mútua (5.11) pode ser muito simplificada no caso dos *sinais gaussianos*—sinais cujos coeficientes de Fourier são variáveis aleatórias independentes, com distribuições gaussianas simétricas de média zero no plano complexo. Muitos sinais naturais se enquadram neste modelo.

Abstratamente, podemos considerar os contornos dos fragmentos digitalizados como sinais gaussianos (as linhas de fratura) corrompidos por ruídos (migalhas de material perdido, erros na aquisição dos dados, etc). Especificamente, os formatos dos dois segmentos correspondentes podem ser escritos como $a(t) = s(t) + n'(t)$ e $b(t) = s(t) + n''(t)$, onde s é a forma da linha de fratura ideal, e n', n'' são as funções de "ruído" que descrevem a perda de material, erros de aquisição, etc.

Sejam A_k, B_k, S_k, N'_k, e N''_k os coeficientes de Fourier de a, b, s, n', e n'', respectivamente. Vamos supor que S_k tem variância $\hat{S}_k$, e que N'_k e N''_k tem a mesma variância $\hat{N}_k$. Neste caso, pela fórmula 5.11, a informação fornecida por cada coeficiente A_k a respeito do coeficiente correspondente B_k é

$$I_k = \log\left[\frac{\hat{A}_k\hat{B}_k}{\hat{S}_k\hat{N}_k + \hat{A}_k\hat{N}_k}\right] = \log\left[\frac{(\hat{S}_k+\hat{N}_k)^2}{(2\hat{S}_k+\hat{N}_k)\hat{N}_k}\right] \tag{5.14}$$

A informação total sobre b obtida a partir de a é então, simplesmente,

$$I_{\text{tot}} = \sum_{k=0}^{m} I_k$$

observe que a soma inclui somente termos com $k \geq 0$, uma vez que os coeficientes de Fourier com k negativo são determinados pela condição $S_{-k} = S_k^*$ característica de sinais com valores reais.

5.3.1 Determinando $\hat{S}_k$ and $\hat{N}_k$

Infelizmente, não temos informação direta sobre a variância do sinal original $\hat{S}_k$ (a função de forma da linha de fratura ideal) ou do ruído $\hat{N}_k$ (a diferença entre as linhas de fratura e os contornos observados). Entretanto, podemos estimar estes parâmetros comparando seções de contornos de fragmentos que sabemos corresponder às mesmas linhas de fratura no objeto original, como os ilustrados na figura 5.3.

Vamos denotar por $a(t)$ e $b(t)$, para $t \in [0..T]$, as funções de forma de dois segmentos de contorno correspondentes, como as da figura 5.5, selecionados de maneira que os pontos centrais $a(T/2), b(T/2)$ dos dois gráficos correspondam ao mesmo ponto da linha de fratura ideal. Seja $m(t) = [a(t)+b(t)]/2$ a média dos dois sinais, e $d(t) = a(t) - b(t)$ sua diferença. Veja a figura 5.6.

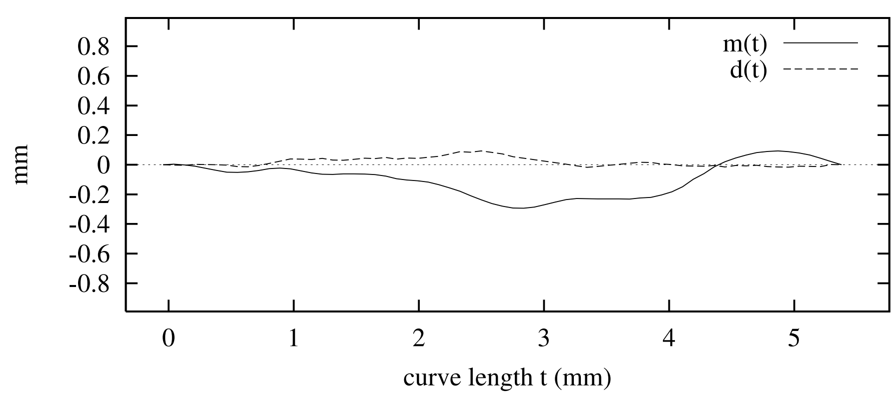


Figura 5.6: A média $m(t) = [a(t) + b(t)]/2$ e a diferença $d(t) = a(t) - b(t)$ das funções de forma da figura 5.5.

Os coeficientes de Fourier M_k e D_k dos sinais m e d têm então variância

$$\begin{aligned}\hat{M}_k &= \operatorname{var}\left(\frac{S_k + N'_k + S_k + N''_k}{2}\right) \\ &= \hat{S}_k + \frac{1}{4}\operatorname{var}(N'_k) + \frac{1}{4}\operatorname{var}(N''_k) \\ &= \hat{S}_k + \frac{1}{2}\hat{N}_k \qquad (5.15)\end{aligned}$$

$$\begin{aligned}\hat{D}_k &= \operatorname{var}((S_k + N'_k) - (S_k + N''_k)) \\ &= \operatorname{var}(N'_k - N''_k) \\ &= \operatorname{var}(N'_k) + \operatorname{var}(-N''_k) \\ &= 2\hat{N}_k\end{aligned}$$

Sendo assim, dado um conjunto de pares de segmentos casados, podemos calcular a partir dele as variâncias $\hat{M}_k$ e $\hat{D}_k$, e então estimar as variâncias $\hat{S}_k$ e $\hat{N}_k$ pelas fórmulas

$$\hat{S}_k = \hat{M}_k - \frac{1}{4}\hat{D}_k \qquad \hat{N}_k = \frac{1}{2}\hat{D}_k \qquad (5.16)$$

Portanto, pela fórmula (5.14), a quantidade de informação contida na componente de freqüência k da curva a sobre a mesma componente de seu parceiro b é

$$\begin{aligned}I_k &= \log\left[\frac{(\hat{A}_k)^2}{\left(2(\hat{M}_k - \frac{1}{4}\hat{D}_k) + \frac{1}{2}\hat{D}_k\right)\left(\frac{1}{2}\hat{D}_k\right)}\right] \\ &= \log\left[\frac{(\hat{A}_k)^2}{\hat{M}_k\hat{D}_k}\right] \qquad (5.17)\end{aligned}$$

Quando estimamos as variâncias $\hat{M}_k$ e $\hat{D}_k$, devemos notar que elas são usadas como argumentos para a função logaritmo, que é altamente não linear neste caso. Portanto, ao invés de computar as variâncias pela fórmula usual, é mais seguro expandir a fórmula (5.17), obtendo

$$I_k = 2\log\hat{A}_k - \log\hat{M}_k - \log\hat{D}_k \qquad (5.18)$$

e então estimar o termo $\log\hat{A}_k$ pela média de $\log(|A_k|^2)$ para vários segmentos, e similarmente para $\log\hat{M}_k$ e $\log\hat{D}_k$.

5.3.2 Verificação de consistência

Para verificar a consistência da fórmula (5.18), vamos supor que $a(t)$ e $b(t)$ são as funções de forma de dois segmentos, com o mesmo comprimento, de contornos não relacionados.

Neste caso, temos $a = s' + n'$ e $b = s'' + n''$, onde s' e s'' são sinais independentes. As variâncias dos coeficientes M_k e D_k são portanto

$$\begin{aligned}
\hat{M}_k &= \operatorname{var}\left(\frac{S'_k + N'_k + S''_k + N''_k}{2}\right) \\
&= \frac{1}{2}(\hat{S}_k + \hat{N}_k) = \frac{1}{2}\hat{A}_k \\
\hat{D}_k &= \operatorname{var}((S'_k + N'_k) - (S''_k + N''_k)) \\
&= 2(\hat{S}_k + \hat{N}_k) = 2\hat{A}_k
\end{aligned}$$

Pela fórmula (5.18), obtemos então

$$I_k = 2\log \hat{A}_k - \log(\frac{1}{2}\hat{A}_k) - \log(2\hat{A}_k) = 0$$

conforme esperado.

5.4 Resultados experimentais

Para testar esta teoria, preparamos cerca de 100 fragmentos oriundos de 4 ladrilhos de cerâmica não vitrificada, fragmentados para o experimento com diâmetro variando de 10 a 50 mm. Digitalizamos esses fragmentos com um scanner de mesa de 300 dpi, e extraímos suas linhas de contornos conforme descrito no capítulo 2. Para remover o ruído de quantização, suavizamos cada contorno com um filtro geométrico gaussiano (capítulo 4), com escala característica $\lambda = 1.8$ pixel ($= 0.152$ mm), e reamostramos cada conjunto com passo uniforme 1 pixel ($= 0.085$ mm). Alguns desses contornos são mostrados na figura 5.7.

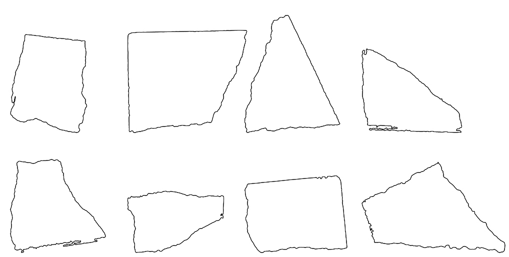


Figura 5.7: Alguns contornos obtidos a partir de fragmentos de cerâmica.

Para o teste, a partir dos contornos filtrados, selecionamos 33 pares de fragmentos adjacentes nos ladrilhos originais, e manualmente extraímos os segmentos correspondentes dos seus contornos, com cerca de 10–20 mm de comprimento. Para cada um desses pares, encontramos o melhor ajuste geométrico, como descrito no capítulo 6. Feito isso, selecionamos dois pontos correspondentes, um de cada segmento, e extraímos de cada segmento um trecho correspondente de comprimento 64 pixels (5.4 mm), centrado no ponto selecionado. Convertemos então estes trechos para funções de forma $a(t)$ e $b(t)$, como explicado na seção 5.2, e calculamos o sinal médio e o sinal diferença $m(t)$ e $d(t)$ para cada par.

As figuras 5.8 e 5.9 e a tabela 5.1 mostram as variâncias estimadas $\hat{A}_k$, $\hat{M}_k$, e $\hat{D}_k$ e o conteúdo de informação útil I_k, para cada componente de freqüência k, computados pela fórmula (5.18).

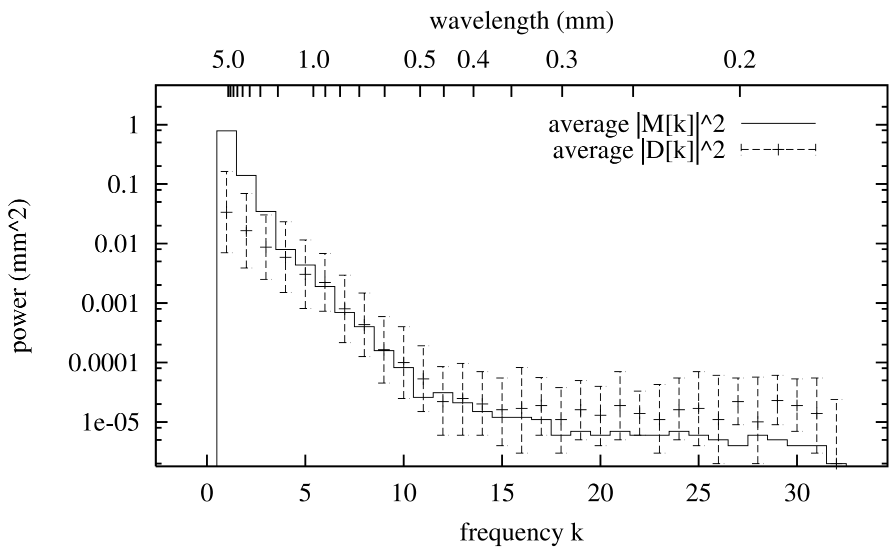


Figura 5.8: Espectro médio de potência do sinal médio($\hat{M}_k$) e do sinal diferença($\hat{D}_k$) para um conjunto de 33 segmentos correspondentes.

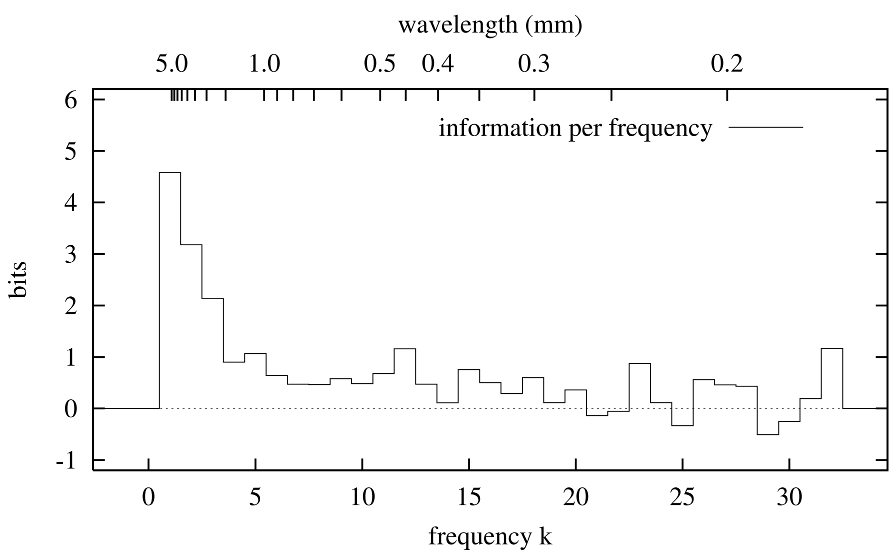


Figura 5.9: Informação útil I_k por freqüência k, calculada a partir dos dados da figura 5.8.

k	$\hat{A}_k$ mm^2	$\hat{M}_k$ mm^2	$\hat{D}_k$ mm^2	I_k bits	I_k/L $\frac{bits}{mm}$
1	0.79343	0.78351	0.03366	4.58	0.845
2	0.14348	0.13903	0.01634	3.18	0.587
3	0.03649	0.03459	0.00872	2.14	0.395
4	0.00935	0.00789	0.00592	0.90	0.166
5	0.00529	0.00437	0.00306	1.07	0.197
6	0.00256	0.00188	0.00223	0.64	0.118
7	0.00088	0.00070	0.00080	0.47	0.087
8	0.00049	0.00040	0.00043	0.47	0.086
9	0.00020	0.00016	0.00016	0.58	0.106
10	0.00011	0.00008	0.00010	0.48	0.089
11	0.00005	0.00003	0.00005	0.68	0.126
12	0.00004	0.00003	0.00002	1.16	0.214
13	0.00003	0.00002	0.00003	0.47	0.087
14	0.00002	0.00002	0.00002	0.11	0.020
15	0.00002	0.00001	0.00002	0.75	0.139
16	0.00002	0.00001	0.00002	0.50	0.093
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
total				18.2	3.356

Tabela 5.1: Resultados para um conjunto de 33 pares de segmentos de contorno correspondentes: espectro de potências do contorno($\hat{A}_k$), sinal médio($\hat{M}_k$), e sinal diferença ($\hat{D}_k$), conteúdo de informação estimado(I_k), e a densidade de informação(I_k/L), por freqüência k.

Observe que a variância $\hat{A}_k$ foi estimada calculando-se a média dos logaritmos de $|A_k|^2$ de todas as amostras, como descrito na seção 5.3, e tomando-se o anti-logaritmo da mesma. O mesmo para $\hat{M}_k$ e $\hat{D}_k$.

A tabela 5.2 mostra o conteúdo de informação I_k condensado por escalas de detalhe (bandas de freqüência logaritmicamente espaçadas), e acumuladas em cada escala.

freq	w.length mm	I_{bd} bits	I_{bd}/L $\frac{\text{bits}}{\text{mm}}$
1..1	2.71..5.42	4.58	0.845
2..3	1.36..2.71	5.32	0.982
4..7	0.68..1.36	3.08	0.568
8..15	0.34..0.68	4.70	0.867
16..32	0.00..0.34	0.50	0.092
total		18.2	3.355

Tabela 5.2: Informação útil (I_k) e densidade de informação (I_k/L), acumuladas por escala de detalhe (banda de freqüência)

A título de experimento de controle, repetimos o processo com 30 pares de segmentos não correspondentes. As figuras 5.10 e 5.11 mostram o espectro de potência médio e o conteúdo de informação útil I_k para cada amostra. Observe que I_k é praticamente nulo neste caso, como esperado (seção 5.3.2).

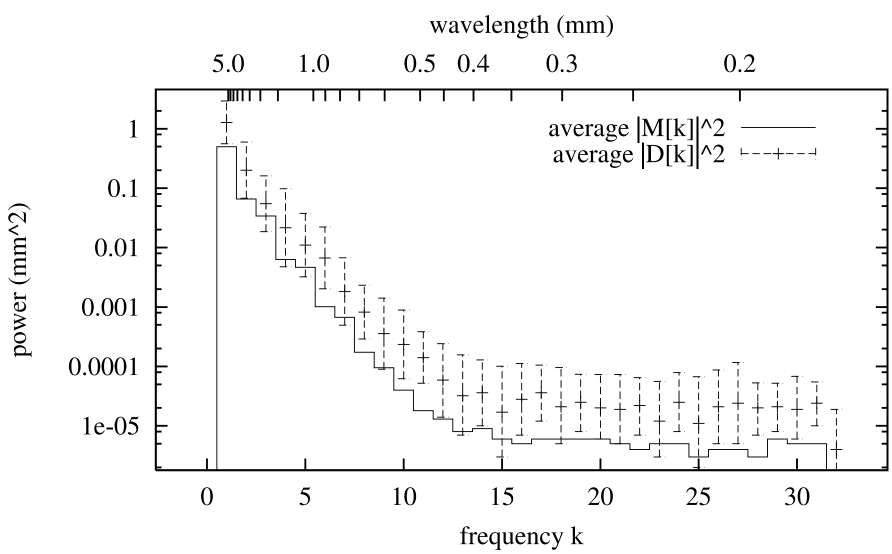


Figura 5.10: Espectro de médio de potência do sinal médio ($\hat{M}_k$) e sinal diferença ($\hat{D}_k$), para um conjunto de 30 pares de segmentos não correspondentes.

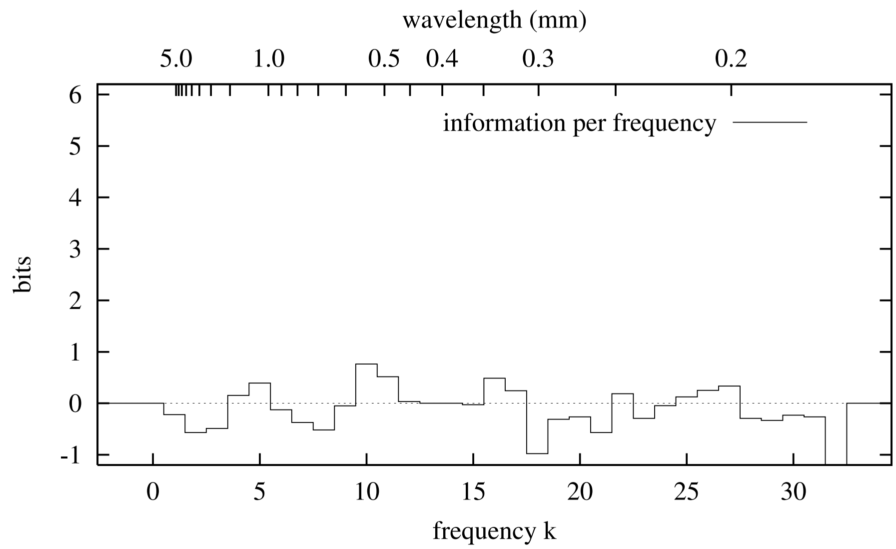


Figura 5.11: Informação útil I_k por freqüência k, calculada para os dados da figura 5.10.

5.5 Considerações finais

Baseados nestes resultados, concluímos que, nos fragmentos utilizados, o formato de um segmento de contorno de 5.4 mm de comprimento contém pelo menos 18 bits de informação útil a respeito do formato do segmento correspondente. Esta informação está quase inteiramente contida nas componentes de freqüência entre 1–16 (comprimentos de onda entre 5.4 e 0.34 mm).

O perímetro médio L dos fragmentos em nossos testes é cerca de 2000 pixels (170 mm). Uma vez que os contornos foram amostrados com passo de 0.5 pixel, cada contorno tem aproximadamente $4000 \approx 2^{12}$ segmentos que podem corresponder a um dado segmento. Portanto, os 18 bits de informação útil contida num segmento de contorno com 5 mm de comprimento permitem identificar pares de segmentos correspondentes entre $2^{18}/2^{12} = 64$ fragmentos, com uma probabilidade de acerto fixa.

Esperamos obter melhores resultados se usarmos segmentos de contorno apenas um pouco maiores do que 5 mm. Se as proporções forem mantidas, um segmento de 10 mm de comprimento conterá 36 bits de informação, suficientes para identificar seu parceiro entre 2^{24} (4 milhões) segmentos. É claro que alguns dos 36 bits seriam redundantes e portanto inúteis; entretanto, dada a natureza fractal dos contornos dos fragmentos (seção 1.6.2) espera-se que esta redundância seja relativamente pequena. Por outro lado, tudo leva a crer que a nova componente, com comprimento de onda $\lambda = 10$ mm, fornecerá sozinha outros 5 bits de informação além dos 36.

Capítulo 6

Discrepância de segmentos de contornos

Neste capítulo, vamos discutir como quantificar a semelhança (ou diferença) na forma de duas curvas e usar o resultado para identificar pares de segmentos correspondentes.

6.1 Conceitos básicos

Vamos supor que cada contorno é uma curva simples fechada, representada por uma seqüência circular de *amostras*. Cada amostra pode ser um ponto do contorno — ou qualquer outra propriedade local, geométrica ou física. Também vamos supor que as amostras são uniformemente espaçadas ao longo do contorno, e que o passo de amostragem δ é aproximadamente o mesmo para todos os contornos.

Um *candidato* é um par de segmentos pertencentes a contornos diferentes. Diremos que um candidato é *verdadeiro* se os seus segmentos correspondem a mesma linha de fratura ideal, e diferem somente pelos erros de aquisição da imagem ou pela perda de migalhas. Caso contrário diremos que o candidato é *falso*. Denotaremos por $\mathcal{V}$ e $\mathcal{F}$ os conjuntos de todos os candidatos verdadeiros e falsos, respectivamente, que existem no conjunto de contornos dados.

Note que quando extraímos o contorno de um fragmento, seguimos sua fronteira no sentido anti-horário, como explicado na seção 2.2. Sendo assim, dois segmentos que compartilham uma mesma linha de fratura são percorridos em direções opostas, como mostrado na figura 6.1. No restante deste capítulo, sempre que compararmos dois segmentos, vamos supor que um deles foi invertido, para compensar este fato.

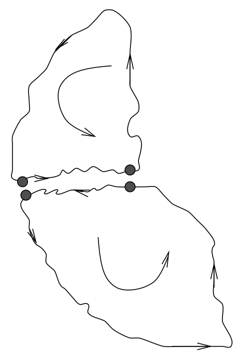


Figura 6.1: Segmentos adjacentes possuem sentidos opostos.

6.2 Classificação por discrepância

Para quantificar a qualidade de um candidato, definiremos o conceito de *discrepância*, uma medida de quão diferentes são os dois segmentos — na forma, ou em alguma outra característica derivada.

6.2.1 Discrepância com correspondência perfeita

Sejam $a(t)$ e $b(t)$, $t \in [0 \,\_\_\, T]$, os dois segmentos de contorno que queremos comparar, com parametrização natural. Vamos supor inicialmente que há uma *correspondência perfeita* entre os dois segmentos, isto é, $a(t)$ e $b(t)$ correspondem ao mesmo ponto da fratura ideal, para todo t. Seja $\|a(t), b(t)\|^2$ uma medida da diferença entre dois valores do contorno $a(t)$ e $b(t)$, que chamaremos de *discrepância quadrática pontual.* Por exemplo, se os valores são pontos no plano podemos usar a distância euclidiana $|a(t) - b(t)|$. A *discrepância quadrática total* do candidato é, por definição,

$$C^2(a, b) = \int_0^T \|a(t), b(t)\|^2 \, dt \tag{6.1}$$

Na prática, os contornos dos segmentos do candidato (a,b) são dados por amostras $a = (a_1 .. a_n)$ e $b = (b_1 .. b_n)$, igualmente espaçadas ao longo dos segmentos. Nesta seção, vamos supor que as amostras estão centradas em n subintervalos iguais de $[0 \,\_\_\, T]$, ou seja

$$a_i = a\left(\frac{i - 1/2}{n} T\right) \qquad b_i = b\left(\frac{i - 1/2}{n} T\right)$$

Neste caso, a integral pode ser aproximada pela somatória das amostras:

$$C^2(a, b) \approx \delta S^2(a, b)$$

onde

$$S^2(a,b) = \sum_{i=1}^{n} \|a_i, b_i\|^2 \tag{6.2}$$

Intuitivamente, quanto menor o valor de $C^2(a,b)$ (ou $S^2(a,b)$) maior a probabilidade dos segmentos a, b corresponderem ao mesmo trecho de uma linha de fratura.

6.2.2 Justificativa da fórmula

Para justificar a utilização da função $S^2(a,b)$ na classificação dos candidatos, vamos estudar analiticamente o comportamento da mesma para uma situação bastante simplificada, em que as amostras são valores reais. Vamos supor primeiramente que o candidato é verdadeiro. Seja então c_i a amostra da curva ideal correspondente às amostras observadas a_i e b_i, e sejam $\alpha_i = a_i - c_i$, e $\beta_i = b_i - c_i$ as perturbações de c_i que resultaram nessas amostras. Vamos supor que as amostras ideais (c_i) e os ruídos (α_i e β_i) são variáveis aleatórias independentes, com distribuições gaussianas de desvios padrão ω e σ ($\sigma \ll \omega$), respectivamente. Neste caso, a diferença $a_i - b_i$ é uma variável aleatória com distribuição gaussiana de variância $\gamma^2 = 2\sigma^2$. Ou seja,

$$\Pr(a_i - b_i \approx z \mid (a,b) \in \mathcal{V}) \;=\; G_{\sigma\sqrt{2}}(z) \;=\; \frac{1}{2\sigma\sqrt{\pi}} \exp\left(-\frac{z^2}{4\sigma^2}\right)$$

Suponha agora que o candidato é falso. Neste caso a_i e b_i são duas amostras não relacionadas — isto é, $a_i = c'_i + \alpha_i$ e $b_i = c''_i + \beta_i$ — e portanto a diferença $a_i - b_i$ é uma variável aleatória com distribuição gaussiana de variância $\phi^2 = 2(\omega^2 + \sigma^2)$. Ou seja,

$$\Pr(a_i - b_i \approx z \mid (a,b) \in \mathcal{F}) \;=\; \frac{1}{2\sqrt{\omega^2+\sigma^2}\sqrt{\pi}} \exp\left(-\frac{1}{4}\frac{z^2}{\omega^2+\sigma^2}\right)$$

Em ambos os casos, prova-se que a função

$$S^2 = S^2(a,b) = \sum_{i=1}^{n} \|a_i, b_i\|^2$$

é uma variável aleatória com distribuição χ^2 de n graus de liberdade [25]. A média de cada termo $\|a_i, b_i\|^2$ é a variância de $a_i - b_i$; ou seja, $\gamma^2 = 2\sigma^2$, se os contornos forem correspondentes, e $\phi^2 = 2(\omega^2 + \sigma^2)$, se não forem. Portanto, o valor médio de $S^2(a,b)$ é $n\gamma^2 = 2n\sigma^2$ no primeiro caso, e $n\phi^2 = 2n(\omega^2+\sigma^2)$ no segundo caso. Isto é,

$$\Pr(S^2 \approx z \mid (a,b) \in \mathcal{V}) \;=\; \frac{1}{\gamma^2}\frac{(z/\gamma^2)^{(n-2)/2}\,\exp\left((-z/\gamma^2)/2\right)}{2^{n/2}\,\Gamma(n/2)} \tag{6.3}$$

$$\;=\; \frac{z^{(n-2)/2}\,\exp(-z/(2\gamma^2))}{\gamma^n\,2^{n/2}\,\Gamma(n/2)} \tag{6.4}$$

$$\Pr(S^2 \approx z \mid (a,b) \in \mathcal{F}) = \frac{1}{\phi^2}\frac{(z/\phi^2)^{(n-2)/2}\,\exp((-z/\phi^2)/2)}{2^{n/2}\,\Gamma(n/2)} \tag{6.5}$$

$$= \frac{z^{(n-2)/2}\,\exp(-z/(2\phi^2))}{\phi^n\,2^{n/2}\,\Gamma(n/2)} \tag{6.6}$$

Veja a figura 6.2.

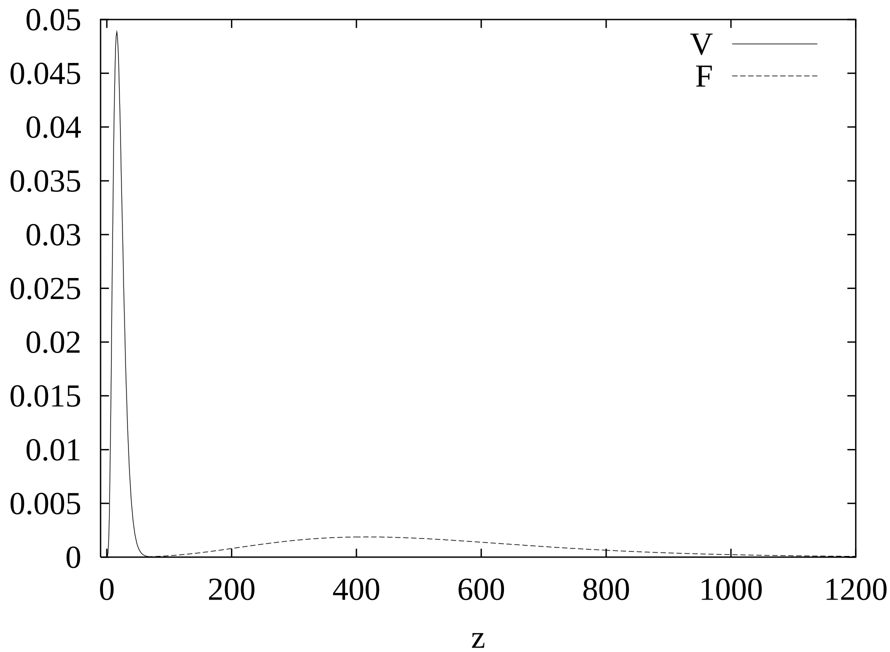


Figura 6.2: Distribuições de probabilidade da medida $S^2(a,b)$, para candidatos verdadeiros e falsos, com $n = 10$ amostras, $\sigma = 1$ e $\omega = 5$.

A figura 6.2 sugere que se $S^2(a,b) \ll 50$ o candidato (a,b) é provavelmente verdadeiro; enquanto que se $S^2(a,b) \gg 120$ ele é provavelmente falso.

O teorema de Bayes [25] permite tornar esta intuição mais precisa, e calcular a probabilidade de um candidato ser verdadeiro ($\mathcal{V}$), ou falso ($\mathcal{F}$), dado o valor de $S^2(a,b)$:

$$\Pr((a,b) \in \mathcal{V} \mid S^2 \approx z) = \frac{\Pr(S^2 \approx z \mid (a,b) \in \mathcal{V})\,\Pr((a,b) \in \mathcal{V})}{\sum_{\mathcal{X}=\mathcal{V},\mathcal{F}} \Pr(S^2 \approx z \mid (a,b) \in \mathcal{X})\,\Pr((a,b) \in \mathcal{X})}$$

ou, mais sucintamente,

$$\Pr(\mathcal{V} \mid S^2 \approx z) = \frac{\Pr(S^2 \approx z \mid \mathcal{V})\,\Pr(\mathcal{V})}{\sum_{\mathcal{X}=\mathcal{V},\mathcal{F}} \Pr(S^2 \approx z \mid \mathcal{X})\,\Pr(\mathcal{X})}$$

As probabilidades "a priori" $\Pr(\mathcal{V})$ e $\Pr(\mathcal{F})$ dependem do número de fragmentos e de seus comprimentos. Note que $\Pr(\mathcal{V})$ é a probabilidade da amostra b_1 ser a parceira correta da amostra a_1; caso seja, os demais pares a_i e b_i também serão corretos, pelas suposições iniciais. Portanto, se o número total de amostras considerando todos os fragmentos juntos (excetuando-se o que contém o segmento a) for M, então $\Pr(\mathcal{V}) = 1/M$. Logo,

$$\Pr(\mathcal{V} \mid S^2 \approx z) = \frac{\Pr(S^2 \approx z \mid \mathcal{V})\frac{1}{M}}{\Pr(S^2 \approx z \mid \mathcal{V})\frac{1}{M} + \Pr(S^2 \approx z \mid \mathcal{F})(1 - \frac{1}{M})} \tag{6.7}$$

$$= \frac{\Pr(S^2 \approx z \mid \mathcal{V})}{\Pr(S^2 \approx z \mid \mathcal{V}) + (M-1)\Pr(S^2 \approx z \mid \mathcal{F})} \tag{6.8}$$

Portanto pelas fórmulas (6.4) e (6.6),

$$\Pr(\mathcal{V} \mid S^2 \approx z) = \frac{\dfrac{z^{(n-2)/2}\exp(-z/(2\gamma^2))}{\gamma^n\, 2^{n/2}\,\Gamma(n/2)}}{\dfrac{z^{(n-2)/2}\exp(-z/2\gamma^2)}{\gamma^n\, 2^{n/2}\,\Gamma(n/2)} + (M-1)\dfrac{z^{(n-2)/2}\exp(-z/(2\phi^2))}{\phi^n\, 2^{n/2}\,\Gamma(n/2)}}$$

Simplificando,

$$\Pr(\mathcal{V} \mid S^2 \approx z) = \frac{1}{1 + (M-1)\left(\frac{\gamma}{\phi}\right)^n \exp\left(\frac{-z}{2}\left(\frac{\phi^2-\gamma^2}{\phi^2\gamma^2}\right)\right)} \tag{6.9}$$

Esta distribuição e seu complemento $\Pr(\mathcal{F} \mid S^2 \approx z) = 1 - \Pr(\mathcal{V} \mid S^2 \approx z)$ estão ilustradas na figura 6.3.

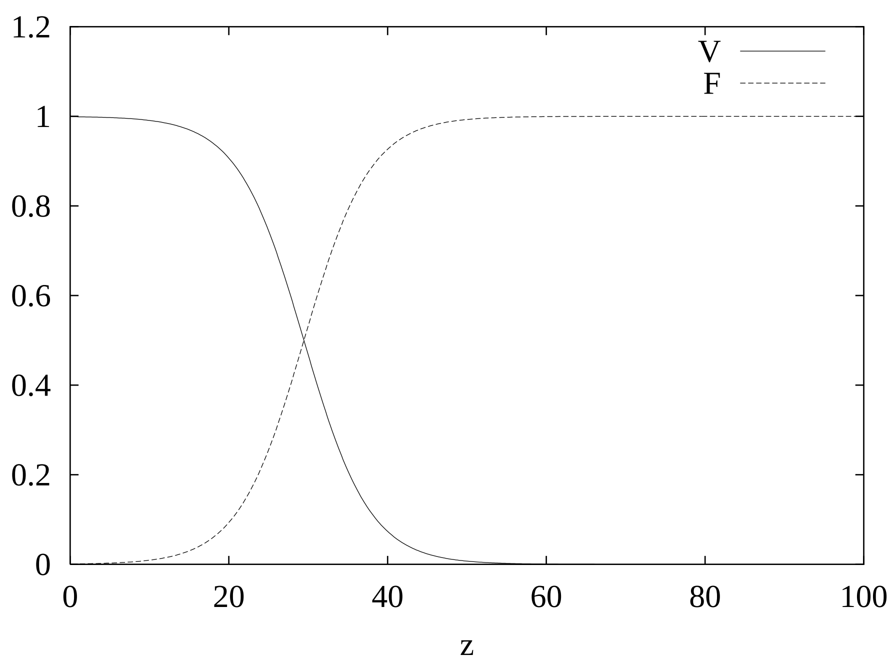


Figura 6.3: $\Pr(\mathcal{V} \mid S^2 \approx z)$ e $\Pr(\mathcal{F} \mid S^2 \approx z)$, para $n = 10$, $\sigma = 1$, $\omega = 5$ e $M = 10000$.

6.2.3 Discrepância de corte

A *discrepância quadrática crítica* é o valor Ξ^2 de $S^2(a,b)$ para o qual o candidato pode ser tanto verdadeiro quanto falso, ou seja, $\Pr(\mathcal{V} \mid S^2 \approx \Xi^2) = 1/2$. Pode-se verificar que

$$\Xi^2 = 2\left[n \ln \frac{\phi}{\gamma} - \ln(M-1)\right] \frac{\phi^2\gamma^2}{\phi^2 - \gamma^2} \tag{6.10}$$

Como pode ser visto na figura 6.3, quando $S^2 \ll \Xi^2$ é quase certo que o candidato é verdadeiro; e quando $S^2 \gg \Xi^2$ é quase certo que ele é falso.

Segundo a fórmula (6.10), Ξ^2 é negativo quando n é menor que um certo número mínimo

$$n_{\min} = \frac{\ln(M-1)}{\ln(\phi/\gamma)} \tag{6.11}$$

A interpretação deste fato é que para candidatos com menos de n_{min} amostras, a hipótese do candidato ser falso é mais provável que a oposta, por menor que seja sua discrepância $S^2(a,b)$.

Podemos escrever Ξ^2 na forma

$$\begin{aligned} \Xi^2 &= n\left(n - \frac{\ln(M-1)}{\ln(\phi/\gamma)}\right) \frac{\phi^2\gamma^2}{\phi^2-\gamma^2} \ \ln\frac{\phi}{\gamma} && (6.12) \\ &= (n - n_{min})\xi^2 && (6.13) \end{aligned}$$

onde

$$\xi^2 = \frac{\phi^2\gamma^2}{\phi^2-\gamma^2} \ \ln\frac{\phi}{\gamma}$$

Chamamos ξ^2 de *discrepância quadrática crítica amostral.*

Informalmente, ξ^2 é o valor médio $\|a_i, b_i\|^2$ que separa candidatos verdadeiros de falsos quando os candidatos são suficientemente longos ($n \gg 1$).

Observe que a discrepância quadrática crítica Ξ^2 depende de n e de M. Entretanto, n_{min} que depende de M é constante, e portanto desprezível à medida que n aumenta.

Se as amostras a_i e b_i são pontos de $\mathbf{R}^d$, em vez de números reais, a variável $S^2(a,b)$ tem distribuição χ^2 com nd graus de liberdade em vez de n. Nesse caso, a análise deve ser modificada de acordo.

O propósito desta análise é mostrar que existem constantes ξ^2 (que depende apenas da variância do "sinal" e do "ruído") e n_{min} (que depende do número de contornos) que nos permitem decidir se um candidato é verdadeiro ou falso, com bastante confiança, a partir do sinal de

$$\Delta(a,b) \ = \ C^2(a,b) - \delta\Xi^2 \ = \ \delta[S^2(a,b) - (n - n_{min})\xi^2] \tag{6.14}$$

Se o sinal de $\Delta(a,b)$ é negativo, o candidato é provavelmente verdadeiro; caso contrário, o candidato é provavelmente falso.

Porém, na prática, não podemos utilizar as fórmulas (6.11) e (6.13) para calcular os valores de ξ^2 e n_{min}, pois não conhecemos os valores de ϕ e γ. Além disso, as fórmulas (6.9), (6.10) e (6.14) foram deduzidas a partir de premissas pouco realistas. Na prática, não existe correspondência um-para-um entre as amostras dos segmentos. Além disso, como usamos mais amostras do que o necessário pelo critério de Nyquist, as amostras não são independentes; portanto, o número de graus de liberdade na distribuição da função D é significantemente menor que n.

Finalmente, se as amostras são pontos do plano ou do espaço, é necessário rodar e transladar um dos segmentos antes da comparação; e este ajuste remove três ou seis graus de liberdade, respectivamente, da distribuição da variável $S^2(a, b)$.

Em vista dessas dificuldades, optamos por determinar os parâmetros ξ^2 e n_{min} empiricamente, como será descrito na seção 9.2.

6.2.4 Saturação da discrepância

Outro fator que deve ser levado em conta no cálculo da discrepância $C^2(a, b)$ é a possibilidade de perda de migalhas, que pode fazer com que a distribuição dos erros $a_i - b_i$ não seja adequadamente modelada por uma gaussiana. Se a diferença de duas amostras é significativamente maior do que o limiar ξ, a hipótese mais provável é que as amostras não correspondem ao mesmo ponto da fratura ideal —- provavelmente por causa de material perdido, rebarbas etc. Neste caso, o valor exato da diferença não é muito significativo. Portanto, para preservar, na medida do possível, as conclusões da seção anterior precisamos definir a discrepância $\|a_i, b_i\|$ como uma função não-linear da diferença $|a_i - b_i|$, tal que $\|a_i, b_i\|^2$ tenha uma distribuição razoavelmente próxima de χ^2 com d graus de liberdade, onde d é a dimensão das amostras.

Na prática, isto significa que $\|a_i, b_i\|$ deve ser definida de modo a não ultrapassar um múltiplo fixo $k\xi$ de ξ. Em nossa experiência, uma escolha adequada é $k = 3$.

Capítulo 7

Correspondência irregular entre segmentos

No capítulo anterior, discutimos a classificação de candidatos supondo que havia uma correspondência um-para-um entre as amostras dos dois segmentos. Entretanto, os erros na aquisição dos contornos podem mudar o comprimento das linhas de fratura. Uma vez que as amostras são escolhidas a intervalos iguais sobre o contorno observado, independentemente, para cada contorno, verifica-se que, em geral, num candidato (a, b), uma amostra a_i de um segmento corresponde a algum ponto entre duas amostras b_j e b_{j+1} do outro segmento. Além disso, como os dois segmentos tem formas diferentes, a amostra a_{i+k} não corresponde necessariamente a um ponto de b entre b_{j+k} e b_{j+1+k}. Veja a figura 7.1.

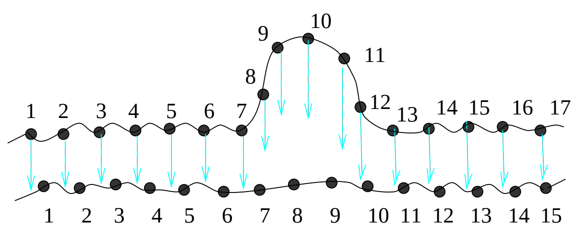


Figura 7.1: um casamento genérico.

Portanto, quando queremos medir a qualidade de um candidato, não podemos considerar uma correspondência um-para-um entre os dois segmentos como feito na fórmula (6.2). Ou seja, precisaremos generalizar a fórmula (6.2) para correspondências que permitem pular amostras em qualquer dos dois segmentos.

7.1 Correspondência perfeita com velocidade variável

Vamos supor primeiramente que os dois segmentos são descritos por funções contínuas $a(t)$ e $b(t)$, para $t \in [0 \_\_ T]$, com uma parametrização de velocidade variável tal que $a(t)$

e $b(t)$ são pontos correspondentes dos dois contornos, para todo t, como na seção 6.2. A fórmula da discrepância entre estas duas curvas (6.1) deve então ser modificada conforme segue:

$$C^2(a,b) = \int_{t=0}^{T} \|a(t), b(t)\|^2 \, \frac{|a^\bullet(t)| + |b^\bullet(t)|}{2} dt \tag{7.1}$$

O fator $(|a^\bullet(t)| + |b^\bullet(t)|)/2\, dt$ representa uma média da distância percorrida pelas duas curvas no intervalo dt. Note que, no caso particular em que há casamento perfeito entre as duas curvas na sua parametrização natural, a fórmula (7.1) reduz-se à fórmula (6.1).

7.2 Casamento contínuo

Na prática, cada segmento é discretizado separadamente com velocidade constante; de modo que, para obter a correspondência correta é necessário aplicar uma reparametrização a ambas as curvas. Definimos portanto um *casamento contínuo* de duas curvas $a(r)$, $r \in [0 \_\_ R]$ e $b(s)$,$s \in [0 \_\_ S]$, como consistindo de duas funções $r(t)$ e $s(t)$, contínuas e diferenciáveis, definidas em algum intervalo $[0 \_\_ T]$, tais que:

$r(0) = s(0) = 0;$

$r(T) = R$ e $s(T) = S;$

$r^\bullet(t) \geq 0$ e $s^\bullet(t) \geq 0,\ \forall\, t \in [0 \_\_ T];$

$(r^\bullet(t), s^\bullet(t)) \neq (0,0),\ \forall\, t \in [0 \_\_ T].$

Diremos que, segundo este casamento, o ponto $a(r(t))$ corresponde ao ponto $b(s(t))$, para todo $t \in [0 \_\_ T]$. Veja a figura 7.2.

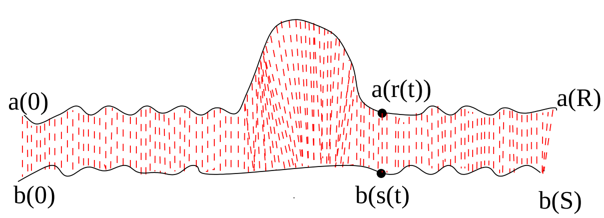


Figura 7.2: Um casamento contínuo entre dois segmentos

7.3 Discrepância com casamentos contínuos

Vamos agora adaptar a fórmula 6.1 para o caso em que os segmentos são descritos por duas funções $a(r)$, $r \in [0 \,\_\_\, R]$ e $b(s)$, $s \in [0 \,\_\_\, S]$, na parametrização natural; ou seja

$$v = \left|\frac{da}{dr}(r)\right| = \left|\frac{db}{ds}(s)\right| = 1$$

para todo r e s; e que a correspondência entre as duas curvas é dada por um casamento contínuo $(r(t), s(t))$, como definido na seção 7.2. A fórmula (7.1) fica então:

$$C^2(a, b; r, s) = \int_0^T \|a(r(t)), b(s(t))\|^2 \, \frac{r^\bullet(t) + s^\bullet(t)}{2} dt \tag{7.2}$$

7.4 Casamento discreto

Para fins computacionais, vamos aproximar um casamento contínuo por um casamento discreto. Suponha novamente que o candidato (a, b) é dado por duas seqüências de amostras $(a_0 .. \, a_m)$, $(b_0 .. \, b_n)$. Nesta seção convém supor que as amostras estão nas extremidades de intervalos de amostragem de largura uniforme δ, isto é

$$a_i = a(i\delta),\ i \in \{0, .. \, m\} \qquad\qquad b_j = b(j\delta),\ j \in \{0, .. \, n\}$$

Um casamento discreto entre esses dois segmentos consiste de uma seqüência de pares de índices (r_k, s_k), $k \in \{1, .. \, p\}$ satisfazendo as seguintes condições:

- $r_0 = 0$, $s_0 = 0$;
- $r_p = m$, $s_p = n$;
- $r_k \leq r_{k+1} \leq r_k + 1$;
- $s_k \leq s_{k+1} \leq s_k + 1$;
- $(r_k, s_k) \neq (r_{k+1}, s_{k+1})$.

para todo k. Diremos que, segundo este casamento, a amostra a_{r_k} corresponde a amostra b_{s_k}, para todo $k \in \{0, .. \, p\}$. Vamos definir o *passo* k do casamento como sendo o par de pares consecutivos $((r_k, s_k), (r_{k+1}, s_{k+1}))$ do mesmo.

Os pares (r_k, s_k) podem ser visualizados como uma "ferrovia" cujos trilhos representam os dois segmentos, e cujos dormentes são os pares (a_{r_k}, b_{s_k}). As condições acima dizem que o espaço entre dois dormentes — ou seja, um passo do casamento — pode ser um quadrilátero, ou degenerar num triângulo. Veja a figura 7.3.

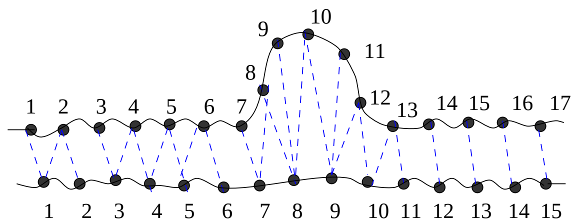


Figura 7.3: Um casamento discreto entre dois segmentos

Observe que um casamento discreto não permite que uma amostra do segmento a corresponda a um ponto entre duas amostras de b. Portanto, a correspondência representável pelo melhor casamento discreto possível pode diferir da correspondência verdadeira (contínua) por, no máximo, metade do passo de amostragem. Em conseqüência, se v_{max} é o valor máximo de $c^{\bullet}(t)$, a discrepância calculada entre as amostras discretas pode diferir de até $\delta v_{max}/2$ em relação a discrepância verdadeira. Este valor é outro motivo pelo qual o valor da discrepância crítica não pode ser facilmente calculado, e precisa ser determinado empiricamente.

7.5 Discrepância com casamento discreto

Para calcular a discrepância $C^2(a, b; r, s)$ nestas condições, podemos interpretar os índices r_k, s_k como amostras de duas funções contínuas $r(t), s(t)$ de $[0 \,\_\_\, p]$ para $[0 \,\_\_\, m\delta]$ e $[0 \,\_\_\, p]$ para $[0 \,\_\_\, n\delta]$, respectivamente, tomadas nos instantes $0, .. p$, e divididas por δ; isto é

$$r_k = \frac{1}{\delta} r(k), \qquad s_k = \frac{1}{\delta} s(k), \qquad k \in \{0, .. p\}$$

Nesse caso, a integral (7.2) pode ser aproximada pelo método do trapézio [7], resultando na fórmula

$$\begin{aligned} C^2(a, b; r, s) = \Big[\quad & \frac{1}{2} \|a_0, b_0\|^2 \, \frac{r^{\bullet}(0) + s^{\bullet}(0)}{2} \; + \\ & \sum_{k=1}^{p-1} \|a_{r_k}, b_{s_k}\|^2 \, \frac{r^{\bullet}(k) + s^{\bullet}(k)}{2} \; + \\ & \frac{1}{2} \|a_m, b_n\|^2 \, \frac{r^{\bullet}(p) + s^{\bullet}(p)}{2} \quad \Big] \end{aligned} \tag{7.3}$$

As derivadas $r^{\bullet}(k)$ e $s^{\bullet}(k)$ podem ser estimadas por diferenças finitas. Para o primeiro e o último termo da somatória, só podemos usar diferenças unilaterais

$$r^{\bullet}(0) = \delta(r_1 - r_0) \tag{7.4}$$

$$r^{\bullet}(p) = \delta(r_p - r_{p-1}) \tag{7.5}$$

Para os demais termos, é preferível usar as diferenças centradas nas amostras,

$$r^{\bullet}(k) = \frac{\delta}{2}(r_{k+1} - r_{k-1}) \qquad s^{\bullet}(k) = \frac{\delta}{2}(s_{k+1} - s_{k-1}) \tag{7.6}$$

Obtemos assim a fórmula

$$\begin{aligned} C^2(a,b;r,s) \ = \Big[\quad & \frac{1}{2}\left\|a_0, b_0\right\|^2 \frac{\delta(r_1 - r_0) + \delta(s_1 - s_0)}{2} \ + \\ & \sum_{k=1}^{p-1} \left\|a_{r_k}, b_{s_k}\right\|^2 \frac{\frac{\delta}{2}(r_{k+1} - r_{k-1}) + \frac{\delta}{2}(s_{k+1} - s_{k-1})}{2} \ + \\ & \frac{1}{2}\left\|a_m, b_n\right\|^2 \frac{\delta(r_p - r_{p-1}) + \delta(s_p - s_{p-1})}{2} \quad \Big] \end{aligned} \tag{7.7}$$

Simplificando,

$$\begin{aligned} C^2(a,b;r,s) \ = \ \frac{\delta}{4}\Big[\quad & \left\|a_0, b_0\right\|^2 (r_1 - r_0 + s_1 - s_0) \ + \\ & \sum_{k=1}^{p-1} \left\|a_{r_k}, b_{s_k}\right\|^2 (r_{k+1} - r_{k-1} + s_{k+1} - s_{k-1}) \ + \\ & \left\|a_m, b_n\right\|^2 (r_p - r_{p-1} + s_p - s_{p-1}) \quad \Big] \end{aligned} \tag{7.8}$$

Para simplificar ainda mais esta fórmula, vamos definir $\epsilon_k = \left\|a_{r_k}, b_{s_k}\right\|^2$, e $\tau_k = r_k + s_k$. Temos então

$$\begin{aligned} C^2(a,b;r,s) &= \frac{\delta}{4}\left[\epsilon_0(\tau_1 - \tau_0) + \sum_{k=1}^{p-1} \epsilon_k(\tau_{k+1} - \tau_{k-1}) + \epsilon_p(\tau_p - \tau_{p-1})\right] \\ &= \frac{\delta}{4}\left[\epsilon_0(\tau_1 - \tau_0) + \sum_{k=1}^{p-1} \epsilon_k(\tau_{k+1} - \tau_k) + \sum_{k=1}^{p-1} \epsilon_k(\tau_k - \tau_{k-1}) + \epsilon_p(\tau_p - \tau_{p-1})\right] \\ &= \frac{\delta}{4}\left[\sum_{k=0}^{p-1} \epsilon_k(\tau_{k+1} - \tau_k) + \sum_{k=1}^{p} \epsilon_k(\tau_k - \tau_{k-1})\right] \\ &= \frac{\delta}{4}\left[\sum_{k=0}^{p-1} \epsilon_k(\tau_{k+1} - \tau_k) + \sum_{k=0}^{p-1} \epsilon_{k+1}(\tau_{k+1} - \tau_k)\right] \\ &= \frac{\delta}{4}\sum_{k=0}^{p-1} (\epsilon_k + \epsilon_{k+1})(\tau_{k+1} - \tau_k) \\ &= \frac{\delta}{4}\sum_{k=0}^{p-1} \Big(\left\|a_{r_k}, b_{s_k}\right\|^2 + \left\|a_{r_{k+1}}, b_{s_{k+1}}\right\|^2 \Big)(r_{k+1} - r_k + s_{k+1} - s_k) \end{aligned} \tag{7.9}$$

O valor

$$c^2(a,b;i,j,i',j') = \frac{\delta}{4}\Big(\left\|a_i, b_j\right\|^2 + \left\|a_{i'}, b_{j'}\right\|^2 \Big)(i' - i + j' - j) \tag{7.10}$$

é dito *discrepância quadrática do passo* $((i,j),(i',j'))$. Note que a discrepância total $C^2(a,b;r,s)$ é a soma de $c^2(a,b;r_k,s_k,r_{k+1},s_{k+1})$ para todos os passos do casamento (r,s).

A discrepância quadrática $C^2(a,b;r,s)$ pode ser usada para discriminar candidatos verdadeiros de candidatos falsos. Como no capítulo 6, pode-se justificar que existem valores ξ^2 e n_{min}, a *discrepância quadrática crítica amostral* e o *comprimento mínimo*, tal que $C^2(a,b;r,s) \ll (L-\delta n_{min})\xi^2$ para candidatos verdadeiros, e $C^2(a,b;r,s) \gg (L-\delta n_{min})\xi^2$ para candidatos falsos; onde L é o comprimento médio dos dois segmentos que compõe o candidato, isto é $L = \dfrac{m+n}{2}\delta$.

Como observado no capítulo 6, para candidatos muito curtos ($L < \delta n_{min}$) o lado direito destas fórmulas é negativo, significando que a hipótese do candidato ser falso é mais possível que a oposta, por menor que seja a discrepância $C^2(a,b;r,s)$.

7.6 Penalidade para casamento imperfeito

A fórmula de discrepância total (7.8) não leva em conta a irregularidade do casamento (r,s); ela retrata apenas a diferença entre as amostras $\|a_i,b_j\|$. Esta insensibilidade pode ser problemática, dependendo da natureza das amostras. Por exemplo, suponha que cada amostra é um único número real, e $\|a_i,b_j\|$ foi definida como sendo $|a_i - b_i|$. Nesse caso, as seqüências a e b abaixo

$$a = (1,0,1,1,1,1,1,0)$$
$$b = (1,0,0,0,0,0,1,0)$$

têm discrepância quadrática zero para o casamento

$$r = (1,2,2,2,2,2,3,4,5,6,7,8) \quad s = (1,2,3,4,5,6,7,7,7,7,7,8)$$

Uma solução para este problema é acrescentar à fórmula da discrepância quadrática total $C^2(a,b;r,s)$ um termo que penaliza explicitamente casamentos imperfeitos, como o do exemplo acima. Em particular, nós adicionamos a $C^2(a,b;r,s)$ o termo de penalidade

$$W^2(r,s) = \delta\zeta^2 \sum_{k=0}^{p-1} \frac{|r_{k+1} - r_k - s_{k+1} + s_k|}{2}$$

onde ζ^2 é um parâmetro que depende da natureza das amostras. Ou seja

$$W^2(r,s) = \sum_{k=0}^{p-1} w^2(r_k,s_k,r_{k+1},s_{k+1}) \tag{7.11}$$

onde

$$w^2(i,j,i',j') \;=\; \frac{|i'-i-j'+j|}{2}\delta\zeta^2 \;=\; \begin{cases} 0 & \text{se } \; i'-i = j'-j \\ \delta\zeta^2/2 & \text{caso contrário} \end{cases} \tag{7.12}$$

O valor de ζ^2 é determinado empiricamente, mas deve ser menor que a discrepância quadrática crítica amostral ξ^2.

Em resumo, para discriminar os candidatos verdadeiros dos falsos, verificamos o sinal de

$$\Delta(a,b;r,s) = C^2(a,b;r,s) + W^2(r,s) - \delta\xi^2\left[\frac{m+n}{2} - n_{min}\right] \tag{7.13}$$

Note que podemos escrever

$$\Delta(a,b;r,s) = \sum_{k=0}^{p-1} d^2(a,b;r_k,s_k,r_{k+1},s_{k+1}) + \delta n_{min}\xi^2 \tag{7.14}$$

onde

$$d^2(a,b;i,j,i',j') = c^2(a,b;i,j,i',j') + w^2(i,j,i',j') - h^2(i,j,i',j') \tag{7.15}$$

com

$$h^2(i,j,i',j') = \frac{i'-i+j'-j}{2}\delta\xi^2 \;=\; \begin{cases} \delta\xi^2 & \text{se } \; i'-i = j'-j \\ \delta\xi^2/2 & \text{caso contrário} \end{cases} \tag{7.16}$$

O termo $W^2(r,s)$ não é necessário quando cada amostra é um ponto do plano (ou do espaço tridimensional), porque neste caso a discrepância $C^2(a,b;r,s)$ é mínima somente se os dois segmentos realmente tem a mesma forma.

Capítulo 8

Programação dinâmica

Neste capítulo, descreveremos como escolher o melhor alinhamento de dois segmentos a e b de um candidato, de modo que sua função discrepância seja mínima. Ou seja, precisamos encontrar um casamento (r, s) dos dois segmentos a e b que minimiza a função $\Delta(a, b; r, s)$ definida pela fórmula (7.13). Para resolver este problema usamos a técnica de *programação dinâmica* (PD).

8.1 Casamento com extremos fixos

Em primeiro lugar, consideraremos o problema de encontrar o melhor casamento de dois segmentos $(a_0, .. a_m)$, $(b_0, .. b_n)$ que começa com o par (a_0, b_0) e termina com o par (a_m, b_n). Como descrito por Meidanis e Setubal [19] este problema pode ser formulado como o problema de achar um caminho de custo mínimo num certo grafo $\vec{\mathcal{G}}$. Os vértices de $\vec{\mathcal{G}}$ são todos os pares $(i, j) \in \{0.. m\} \times \{0.. n\}$. Cada aresta corresponde a dois pares de índices que podem aparecer consecutivamente num casamento válido:

$$\begin{array}{l}(i, j) \to (i+1, j),\\(i, j) \to (i, j+1),\\(i, j) \to (i+1, j+1)\end{array}$$

para $0 \leq i \leq m-1$ e $0 \leq j \leq n-1$. Veja a figura 8.1.

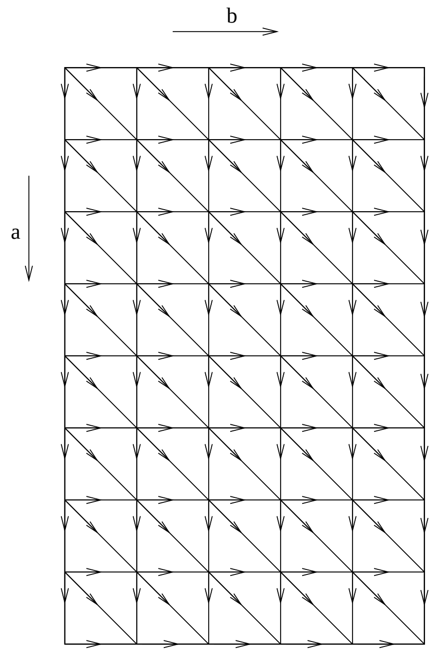


Figura 8.1: Grafo $\vec{\mathcal{G}}$.

A cada aresta deste grafo, com extremos $(i, j) \to (i', j')$, associamos um custo $d^2(a, b; i, j, i', j')$, calculado pela fórmula (7.15).

Podemos ver que um casamento discreto (r, s) de a e b, como definido na seção 7.5, corresponde a um caminho no grafo $\vec{\mathcal{G}}$ do vértice $(0, 0)$ ao vértice (m, n). Os vértices do caminho são os pares (r_k, s_k), e o custo total deste caminho é

$$\sum_{k=0}^{p-1} d^2(a, b; r_k, s_k, r_{k+1}, s_{k+1}) \; = \; \Delta(a, b; r, s) - \delta n_{min} \xi^2 \qquad (8.1)$$

Vide fórmula (7.14). Note que $\delta n_{min} \xi^2$ é constante, de modo que minimizar o custo de um caminho em $\vec{\mathcal{G}}$ corresponde a minimizar o indicador $\Delta(a, b; r, s)$, ou seja, determinar o casamento (r, s) que tem maior probabilidade de ser verdadeiro.

8.2 O algoritmo de programação dinâmica

No algoritmo de PD, o grafo $\vec{\mathcal{G}}$ é representado por uma matriz $(M_{i,j}, \;\; i \in 0, .. \, m, \;\; j \in 0, .. \, n)$ cujas linhas e colunas correspondem às amostras de a e b, respectivamente. Cada elemento $M_{i,j}$ da matriz corresponde ao vértice (i, j) de $\vec{\mathcal{G}}$. O valor de $M_{i,j}$ é o custo do caminho mais barato em $\mathcal{G}$ do vértice $(0, 0)$ ao vértice (i, j); Ou seja, $M_{i,j}$ é o valor mínimo para Δ para qualquer casamento do segmento (a_0, a_i) com o segmento (b_0, b_j).

O algoritmo de PD constrói esta matriz recursivamente pela fórmula

$$M_{i,j} \leftarrow \min \begin{cases} d^2(a,b;i-1,j-1,i,j) & + \quad M_{i-1,j-1}; \\ d^2(a,b;i-1,j,i,j) & + \quad M_{i-1,j}; \\ d^2(a,b;i,j-1,i,j) & + \quad M_{i,j-1}; \end{cases} \tag{8.2}$$

onde i varia de 1 até m, e j varia de 1 até n.

A coluna $j = 0$ é inicializada com $M_{0,0} \leftarrow 0$ e $M_{i,0} \leftarrow d^2(a,b;i-1,0,i,0) + M_{i-1,0}$, para $i = 1..\,m$. A linha $i = 0$ é inicializada de maneira análoga.

Uma vez calculados os custos mínimos de todos os pares (i,j) pela recorrência (8.2), é trivial encontrar o caminho de menor custo custo do vértice $(0,0)$ até o vértice (m,n), invertendo os cálculos que resultaram em $M_{m,n}$. Ou seja, começamos em $i' = m$, $j' = n$, e repetidamente passamos do vértice (i',j') para o vértice (i,j), com $i'-1 \leq i \leq i'$, $j'-1 \leq j \leq j'$ e $(i,j) \neq (i',j')$, cujo custo é $M_{i,j} = M_{i',j'} - d^2(a,b;i,j,i',j')$.

8.3 Casamento com extremos livres

O algoritmo básico da PD supõe que o ínicio dos segmentos a casar é conhecido. Entretanto, na nossa aplicação, isto não é verdade. Como veremos na seção 10.1, o problema central que precisamos resolver é na verdade o seguinte: dados dois segmentos $a = (a_0, ..\, a_m)$ e $b = (b_0, ..\, b_n)$ que são *aproximadamente* correspondentes, encontrar dois sub-segmentos $a' = (a_{u_a}, ..\, a_{v_a})$ e $b' = (b_{u_b}, ..\, b_{v_b})$ e um casamento (r,s) entre a' e b', que minimizem o valor $\Delta(a',b';r,s)$.

Conforme discutido na seção 6.2.3, para candidatos verdadeiros o valor de $\Delta(a',b';r,s)$ é mínimo quando o comprimento do sub-candidato (a',b') é o maior possível; enquanto que, para casamentos falsos, $\Delta(a',b';r,s)$ é mínimo quando o comprimento é menor que o comprimento mínimo.

8.3.1 Casamento com um extremo livre

Existem modificações da PD [19] que resolvem este problema na suposição que pelo menos um dos sub-segmentos a' e b' começa (ou termina) no início do respectivo super-segmento a, b. Infelizmente, esta restrição não pode ser imposta no nosso caso.

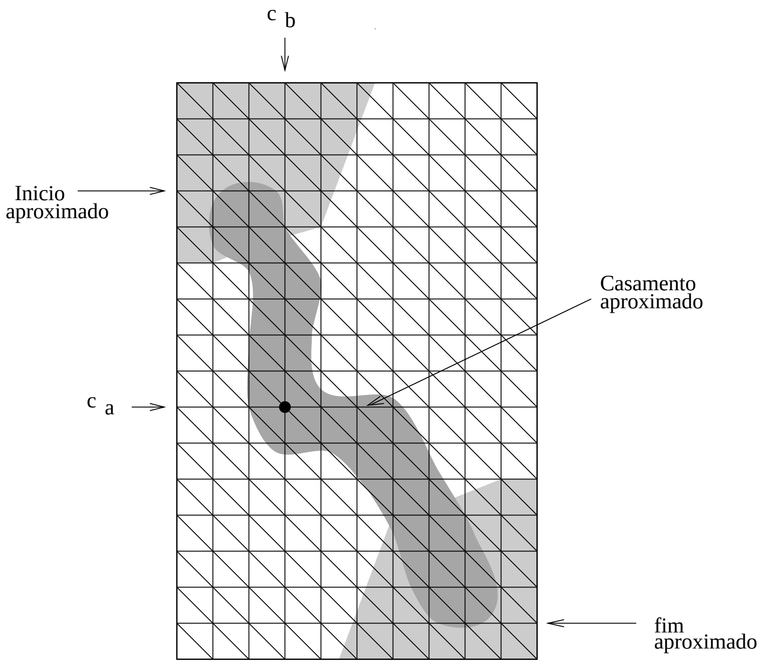


Figura 8.2: Caminho mínimo aproximado.

8.3.2 Casamento centrado

Por outro lado, na nossa aplicação, sabemos que o casamento procurado, se for interessante, deve cobrir uma fração significativa dos segmentos dados a e b. Ou seja, em termos do grafo $\vec{\mathcal{G}}$, sabemos que o caminho procurado começa "perto" do vértice $(0,0)$ e termina "perto" do vértice (m,n). Além disso, conhecemos um casamento aproximado entre os segmentos originais a e b, que dá uma idéia aproximada do caminho mínimo procurado (veja a figura 8.2).

Em particular, temos uma idéia aproximada do ponto médio desse caminho: sabemos que ele passa próximo a um par dado (c_a, c_b), que chamamos *centro* do candidato dado.

Nossa estratégia é portanto examinar todos os caminhos de custo mínimo no grafo $\vec{\mathcal{G}}$ que passam perto do par (c_a, c_b). Mais precisamente, examinamos todo caminho que passa por algum vértice (k_a, k_b), tal que $|c_a - k_a| \leq q$ e $|c_b - k_b| \leq q$, onde q é um parâmetro dado.

Pode-se verificar que, para isso, basta considerar os vértices (s_a, s_b) tais que

$$\begin{cases} s_a + s_b = c_a + c_b \\ |s_a - s_b| \leq q \end{cases}$$

ou

$$\begin{cases} s_a + s_b = c_a + c_b + 1 \\ |s_a - s_b| \leq q - 1 \end{cases}$$

A figura 8.3 mostra a posição destes vértices no grafo:

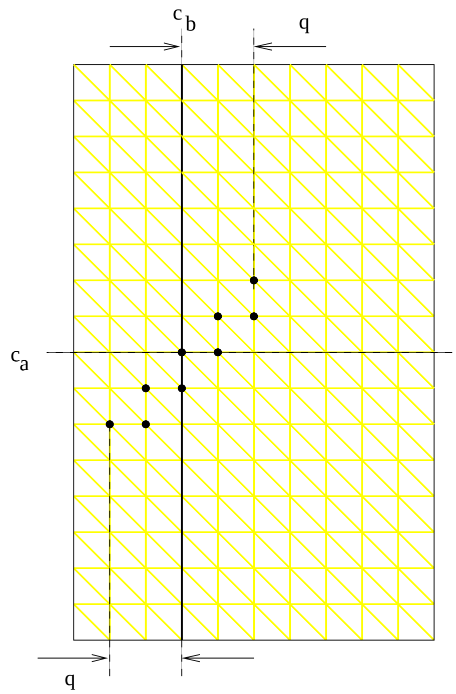


Figura 8.3: Centros prováveis.

É obvio que todo caminho que começa "perto" do ponto $(0, 0)$, termina "perto" de (m, n) e passa a uma certa distância $\leq q$ de (c_a, c_b) forçosamente passa por um dos vértices (s_a, s_b) definidos acima. Portanto, reduzimos nosso problema a $4q+1$ instâncias do seguinte subproblema: encontrar um caminho de custo mínimo no grafo $\vec{\mathcal{G}}$ que passa por um vértice (s_a, s_b) dado.

8.3.3 Programação dinâmica bidirecional

Para resolver este problema, partindo de cada possível centro (s_a, s_b), aplicamos o algoritmo de programação dinâmica duas vezes: primeiro na direção decrescente dos índices (trabalhando implicitamente com o grafo $\overleftarrow{\mathcal{G}}$, obtido invertendo-se a direção das arestas de $\vec{\mathcal{G}}$), e depois na direção crescente dos índices.

O resultado imediato dessas duas aplicações é preencher uma parte M' da matriz M (sombreada na figura 8.4) com os custos mínimos de caminhos que levam do par (s_a, s_b) a cada par (i_a, i_b) de M e vice-versa.

Seja (u_a, u_b) o menor elemento da submatriz $[0.. s_a] \times [0.. s_b]$ de M. Analogamente, seja (v_a, v_b) o menor elemento da submatriz $[s_a.. m] \times [s_b.. n]$ de M.

A concatenação dos caminhos P (invertido) e Q é o caminho de custo mínimo que passa pelo vértice (s_a, s_b), ou seja o casamento de dos dois segmentos a, b que contém o par (s_a, s_b) e minimiza a somatória (8.1). Como o termo $\delta\phi^2$ independe dos segmentos e do casamento, concluímos que este caminho também minimiza $\Delta(a, b; r, s)$.

Seja P o caminho de custo total mínimo em $\overleftarrow{\mathcal{G}}$ do centro (s_a, s_b) até (u_a, u_b). Seja Q o caminho de custo total mínimo em $\overrightarrow{\mathcal{G}}$ do centro (s_a, s_b) até (v_a, v_b). Para construir o caminho Q, basta partir de (v_a, v_b) e ir voltando até (s_a, s_b). De maneira análoga se obtém o caminho P.

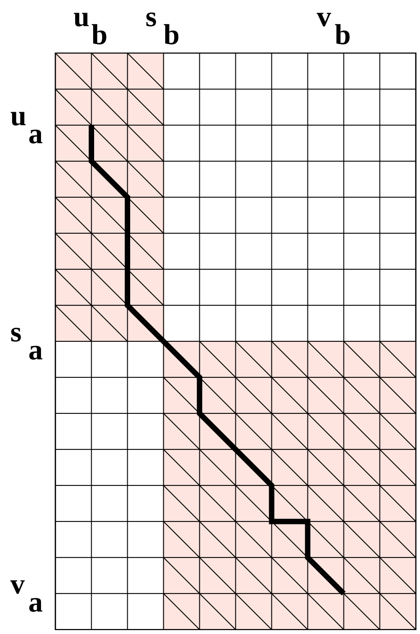


Figura 8.4: Caminho de menor custo contruído a partir de (s_a, s_b).

8.4 Custo da comparação

É fácil ver que o custo do algoritmo de programação dinâmica bidirecional, para um vértice determinado de partida (s_a, s_b) dado, é proporcional a $\Theta(mn)$, pois o número células da matriz M que precisam se calculadas é no mínimo $mn/2$ e no máximo mn. Portanto o custo total do algoritmo de programação dinâmica bidirecional para todos os vértice de partida (s_a, s_b) é $\Theta(qmn)$.

8.4.1 Um algoritmo mais eficiente

É possível diminuir ainda mais o custo da comparação usando toda informação disponível no casamento encontrado na escala mais grosseira.

Esta otimização supõe que o casamento na escala fina não será muito diferente do anterior uma vez levada em conta a correspondência entre as duas escalas. Ela significa que o caminho ótimo no grafo $\overrightarrow{\mathcal{G}}$ deve ser procurado dentro de uma faixa relativamente estreita. Mais precisamente, seja $(r, s) = (r_0, s_0)..(r_{p-1}, s_{p-1})$ um casamento inicial para o candidato (a, b) (Note que (r, s) pode cobrir apenas parte dos segmentos a e b).

Definimos o conjunto V dos *pares iniciais* como sendo todos os pares (r_k, s_k) desse casamento, mais todos os pares (i, j) com $0 \leq i \leq r_0$, e $0 \leq j \leq s_0$, mais todos os pares (i, j) com $r_{p-1} \leq i \leq m$ e $s_{p-1} \leq j \leq n$.

Ou seja,

$$\begin{aligned} V &= \{(r_k, s_k) : 0 \leq k \leq p-1\} \\ &\quad \cup \{(i,j) : 0 \leq i \leq r_0, \text{ e } 0 \leq j \leq s_0\} \\ &\quad \cup \{(i,j) : r_{p-1} \leq i \leq m, \text{ e } s_{p-1} \leq j \leq n\} \end{aligned} \tag{8.3}$$

Definimos a partir de V, o conjunto dos *pares válidos* $V^\star$ como sendo

$$V^\star = \{(i + \alpha, j + \beta) : (i, j) \in V, \text{ e } \alpha, \beta \in [-q..+q]\} \tag{8.4}$$

Veja a figura 8.5.

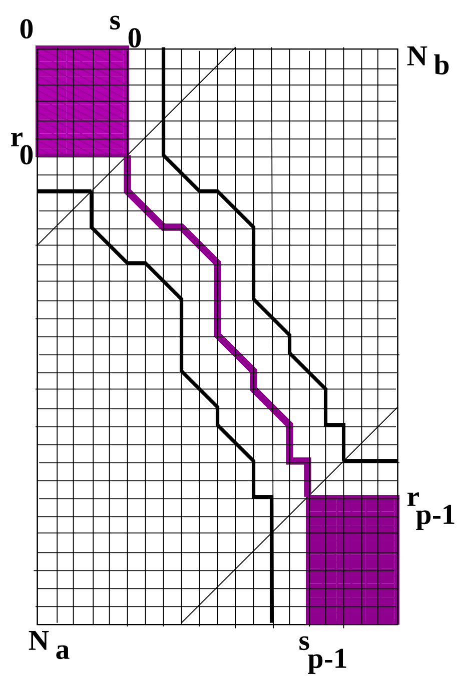


Figura 8.5: Caminho aproximado e faixa de interesse.

Se o intervalo $[-q..+q]$ for suficientemente grande, e o casamento (r, s) for suficientemente correto, o casamento entre a e b estará contido no conjunto de pares $V^\star$. Pode-se observar que a fronteira do conjunto $V^\star$ consiste basicamente de duas cópias do casamento inicial (r, s), deslocadas de $+q$ e $-q$ na direção diagonal. Este fato permite representar e enumerar o conjunto $V^\star$ de maneira simples e eficiente.

Normalmente o parâmetro q é uma constante pequena (2 a 5) independentemente do tamanho dos segmentos a, b. Da mesma forma, o número de amostras de a e b que não são cobertas pelo casamento inicial é também uma constante relativamente pequena. Portanto, o conjunto $V^\star$ tem $O((m+n)q)$ pares em vez de $O(mnq)$. Isto significa que o custo total do algoritmo de programação dinâmica bidirecional para todos os vértices de partida será $O((m+n)q^2)$.

Capítulo 9

Codificação dos contornos por curvatura

9.1 Invariantes Geométricos

Uma vez que as orientações e as posições em que os fragmentos foram digitalizados são arbitrárias, não é possível encontrar pares semelhantes comparando diretamente as coordenadas dos pontos. Para evitar o custo elevado de testar todas as rotações e translações possíveis, precisamos extrair dos contornos algumas características que não se alterem quando o fragmento é rodado ou transladado.

Exemplos de tais invariantes geométricos são a área, o perímetro e o momento de inércia do contorno. Infelizmente estes invariantes são inúteis para o nosso problema, pois dependem do contorno inteiro.

A função de forma definida no capítulo 5 também é invariante sob rotação e translação. Entretanto, ela depende do início e fim do segmento para o qual ela é calculada. Esta discrepância não é aceitável na nossa aplicação, pois o início e o fim dos segmentos são parte da informação a determinar.

Os invariantes que precisamos devem ser locais, isto é, devem depender apenas de um pequeno trecho do contorno.

9.1.1 Curvatura

O invariante local mais simples é a *curvatura*. A curvatura num instante t é dada pela seguinte expressão:

$$\kappa(t) = \frac{|c^{\bullet}(t) \ \times \ c^{\bullet\bullet}(t)|}{|c^{\bullet}(t)|^3} \tag{9.1}$$

onde $c^{\bullet}(t)$ é a primeira derivada da curva, e $c^{\bullet\bullet}(t)$ é a segunda derivada.

A curvatura é bastante utilizada no reconhecimento de formas, como por exemplo nos trabalhos de H.J.Wolfson[38, 39], P.Rosin[30], e A.Kalvin e outros[13].

Uma desvantagem do uso da curvatura é que curvas muito distintas podem ter gráficos de curvatura parecidos. Em geral, a discrepância calculada a partir das curvaturas não tem nenhuma relação rígida com a diferença de forma. Entretanto, a análise do capítulo 6 continua válida: existe um valor ξ da discrepância média amostral (diferença de curvatura) que distingue os candidatos verdadeiros dos falsos; e esta separação é bastante confiável para candidatos de comprimento suficiente.

9.1.2 Complementação das curvaturas

Note-se que, ao compararmos segmentos de curvaturas codificadas, um dos segmentos deve ser complementado além de ser invertido (como explicado na seção 6.1), para que as curvaturas dos trechos se tornem comparáveis.

9.1.3 Estimativa numérica da curvatura

Um obstáculo ao uso da curvatura como invariante de forma é que ela é muito sensível a ruído [39]. Ou seja, pequenas mudanças na forma do contorno (como erros de quantização, rebarbas ou perdas de migalhas do fragmento) podem acarretar diferenças grandes de curvatura. Além disso, o cálculo das derivadas $c^{\bullet}(t)$ e $c^{\bullet\bullet}(t)$ por diferenciação numérica é instável.

Entretanto, podemos reduzir a severidade destes problemas trabalhando com uma versão filtrada da curva e usando um número de amostras superior ao critério de Nyquist. O cálculo da velocidade e da aceleração nos pontos amostrais, que é o que nos interessa é realizado como descrito abaixo.

Dados os pontos $c_i = c(t_i)$, amostrados nos instantes t_i, $i = 0, .. m-1$, as velocidades $v_i = c^{\bullet}(t)$ nesses instantes podem ser estimadas pela fórmula

$$v_i = \frac{c_{i+1} - c_{i-1}}{2\delta}$$

onde δ é o tamanho do passo, já que estamos considerando que após a filtragem a distância entre duas amostras consecutivas é constante.

As acelerações nesses instantes podem então ser estimadas pela fórmula

$$a_i = \frac{c_{i-1} - 2c_i + c_{i+1}}{\delta^2}$$

9.2 Determinação da discrepância crítica

Como observamos na seção 6.2.1, a discrepância crítica pontual ξ^2, que distingue candidatos verdadeiros de falsos, deve ser determinada por testes empíricos. Especificamente, nós usamos o seguinte método:

1. Filtramos os contornos na escala desejada λ, e amostramos com passo uniforme $\delta = \lambda/4$.

2. Calculamos a curvatura de cada amostra, pela fórmula 9.1.

3. Escolhemos aleatoriamente um conjunto aproximado de candidatos (a, b), verdadeiros e falsos, de comprimento suficiente, e extraímos os trechos correspondentes das cadeias de curvatura.

4. Para cada candidato (a, b), escolhemos um trecho (a', b') de comprimento fixo, centrado em (a, b), e calculamos para o mesmo um casamento discreto (r, s) que minimiza o valor dos termos da discrepância $G^2(a', b'; r, s) = C^2(a', b'; r, s) + W^2(r, s)$, segundo as fórmulas (7.8) e (7.11). Neste cálculo, a discrepância amostral $\|a_i, b_j\|$ é a diferença $|a_i - b_j|$ das curvaturas estimadas nas duas amostras correspondentes. Seja $G^2_{\min}(a, b)$ o valor de $G^2(a', b'; r, s)$ para este casamento.

5. Para cada categoria de candidatos (verdadeiros e falsos) construímos um histograma dos valores $G^2_{\min}(a, b)/((L_a + L_b)/2)$, onde L_a e L_b são os comprimentos dos dois segmentos.

6. Determinamos um valor ξ^2 que separa estes dois histogramas.

A figura 9.1 mostra uma instância destes histogramas, obtidos a partir dos contornos do exemplo 11.7 filtrados com escala característica $\lambda = 32$.

9.2.1 Influência do comprimento

A figura 9.1 mostra a influência do comprimento dos candidatos na sua classificação. Este gráfico foi obtido a partir de um conjunto de 10 candidatos verdadeiros e 10 falsos. Para cada candidato (a, b), e cada comprimento $(L_a + L_b)/(2\delta)$, encontramos o casamento (r', s') que cobre um trecho central (a', b') de (a, b) de comprimento $n' = (L'_a + L'_b)/(2\delta)$ e tem discrepância total $G^2(a, b; r, s)$ mínima; seja $G^2_{\min}(a, b; n')$ esta discrepância. O gráfico mostra a evolução do quociente $G^2_{\min}(a, b; h)/n'$ em função de n', para cada candidato (a, b).

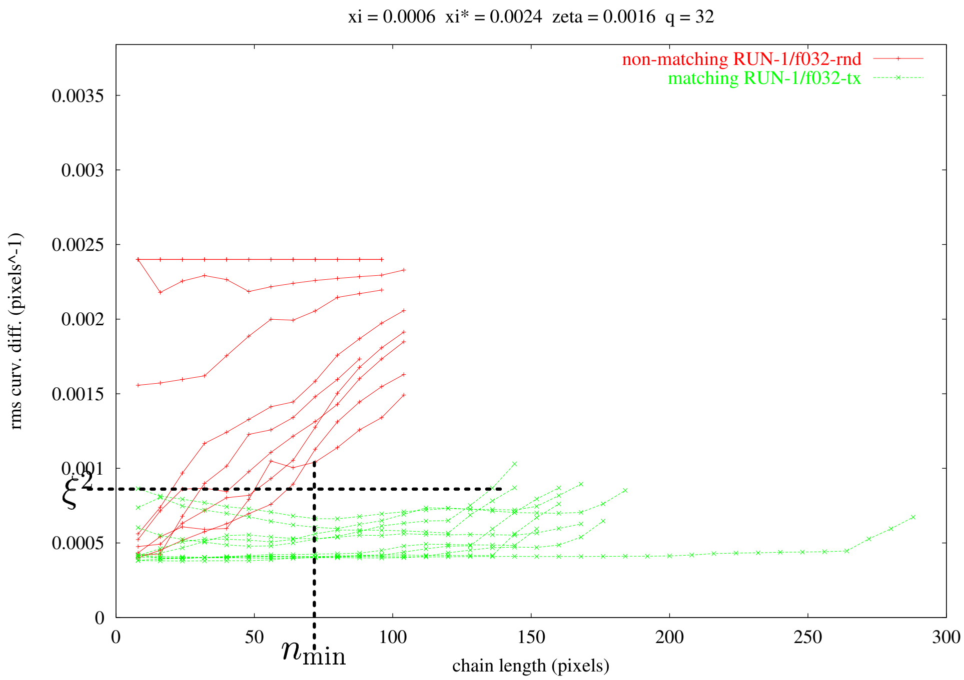


Figura 9.1: Gráfico de $G^2_{\min}(a,b;n')/h$ em função de n' para alguns candidatos, verdadeiros $(\times)$ and falsos $(+)$.

Podemos ver neste gráfico que a ordenada $\xi^2 \approx 0.0008$ separa os candidatos verdadeiros $(\times)$ dos falsos $(+)$, pelo menos para $h > 60$ pixels. Portanto, se usarmos este valor de ξ^2 na fórmula $\Delta(a,b) = G^2_{\min}(a,b) - n\xi^2$, teremos um valor negativo para os primeiros e positivo para os últimos.

9.2.2 Amostragem da curva

No cálculo da curvatura, supomos que a curva é dada por amostras igualmente espaçadas no tempo, com freqüência suficiente para satisfazer o teorema de Nyquist. Como visto na seção 3.8.7, para curvas que foram filtradas com filtro gaussiano de comprimento característico λ, o critério de Nyquist é praticamente satisfeito se o espaço entre as amostras for menor ou igual a $\lambda/4$.

Na verdade, como os algoritmos de comparação consideram apenas casamentos discretos (em que cada amostra de uma curva é comparada com uma amostra da outra) é necessário usar uma frequência de amostragem tal que o erro devido a esta limitação seja tolerável. Ou seja, se a derivada da curvatura $|d\kappa/dt|$ é limitada por uma constante β, e o intervalo de tempo entre amostras é ϵ, a restrição a casamentos discretos introduzirá um erro menor ou igual a $\frac{1}{2}\epsilon\beta$ na curvatura comparada. Portanto se ξ é a diferença significativa de curvatura, deveríamos usar passo $\delta < 2\xi/\beta$. Não levantamos dados sobre a distribuição dos valores de $|d\kappa/dt|$, mas os resultados satisfatórios obtidos $\delta = \lambda/4$ indicam que este passo satisfaz o critério acima.

9.3 Codificação das curvaturas

As curvaturas calculadas segundo a fórmula (9.1) são números reais. Como veremos, o processo de comparação exige acesso aos dados de curvatura de todas as curvas ao mesmo tempo, implicando em um alto custo de armazenamento. Entretanto, uma vez que não é necessário saber o valor exato da curvatura, podemos economizar espaço em disco e memória, quantizando cada amostra para um conjunto pequeno de valores discretos, $[-q\ ..\ +q]$, especificamente inteiros num intervalo limitado. Desta forma, cada amostra pode ser codificada de maneira compacta com $\lceil \log_2(2q+1) \rceil$ bits. Note-se entretanto que os níveis de quantização devem ser suficientemente finos para que as diferenças significativas de curvatura sejam preservadas na cadeia codificada.

Por conveniência e para facilitar a depuração dos programas, nas ilustrações a seguir utilizamos $q = 26$, e representamos cada amostra por uma letra minúscula para curvatura negativa (convexidades), uma letra maiúscula para curvatura positiva (concavidades), e '0' para curvatura nula (partes retas da curva). Veja as figuras 9.2 e 9.3. Numa implementação otimizada, entretanto, seria melhor utilizar $q = 127$ o que ainda permite representar cada amostra num único *byte*.

Definimos uma *cadeia* como uma seqüência de um ou mais valores quantizados de curvatura. Observe que há correspondência 1-1 entre os valores quantizados da cadeia e os pontos amostrados na curva correspondente.

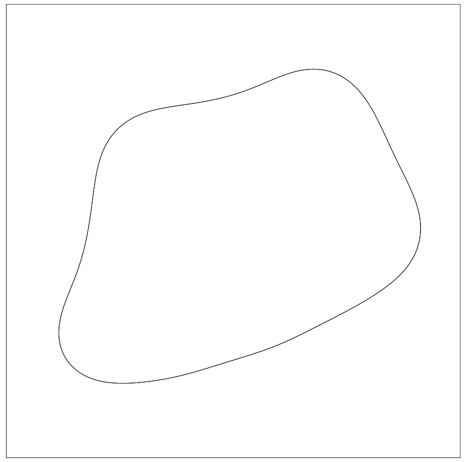
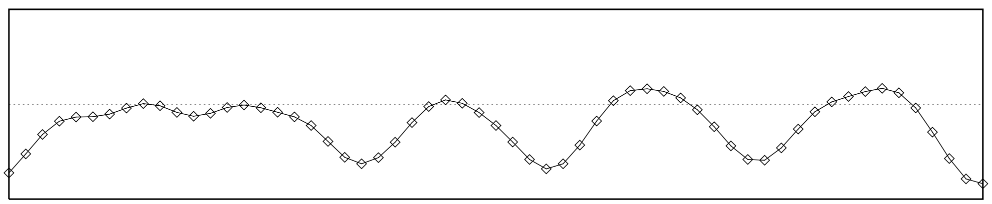


Figura 9.2: Contorno filtrado de um fragmento.



`TROLKKJEaBIKIEAEIKMPRSRQMCfaIMPRSSQLeklkgGNQRSQNHchkljEORTT`

Figura 9.3: Gráfico de curvatura e cadeia de curvatura codificada para o contorno da figura 9.2.

9.3.1 Função de quantização

Ao quantizar a curvatura, devemos utilizar um número suficiente de valores para que o erro de quantização seja desprezível em relação às diferenças significativas de curvatura; isto é da ordem do ruído inerente na curvatura.

Por outro lado, para maximizar a eficiência da codificação (isto é; maximizar o número de bits de informação útil na cadeia codificada) é desejável que a distribuição dos valores codificados seja aproximadamente uniforme.

Verifica-se experimentalmente que a distribuição de valores da curvatura κ, para materiais cerâmicos, é aproximadamente uma gaussiana com média $\bar{\kappa} \approx 0$. Veja por exemplo a figura 9.4 referente aos fragmentos cerâmicos mostrados nas figuras **??** e 11.7.

Portanto, antes de quantificar os valores da curvatura, aplicamos aos mesmos a função ϕ definida por

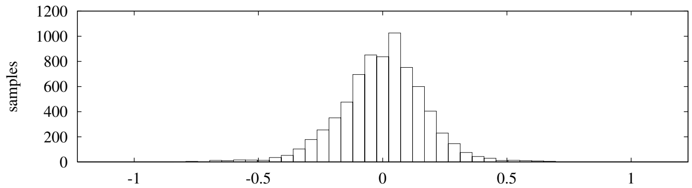
$$\phi(\kappa) = \operatorname{erf}(\kappa/(\hat{\kappa}\sqrt{2})) = \frac{2}{\sqrt{\pi}} \int_0^{\kappa/(\hat{\kappa}\sqrt{2})} \exp(-x^2)dx \qquad (9.2)$$

Esta função transforma valores reais em distribuição normal de média zero e desvio padrão $\hat{\kappa}$ em valores uniformemente distribuídos entre -1 e $+1$. A figura 9.5 mostra a distribuição dos valores $\phi(\kappa)$ para as mesmas curvas da figura 9.4.

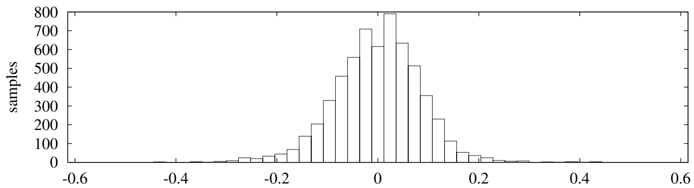
A cadeia de curvatura codificada consiste então em valores de $q\phi(\kappa)$ em cada ponto amostrado da curva, arredondados para o inteiro mais próximo.

9.3.2 Estimativa do parâmetro $\hat{\kappa}$

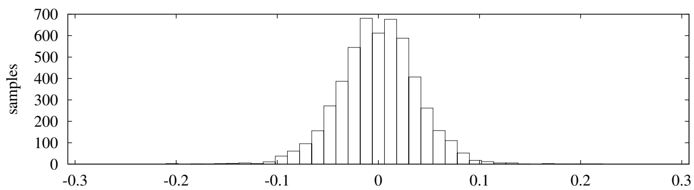
O parâmetro $\hat{\kappa}$ da função de quantização (9.2) deve em princípio ser determinado empiricamente, para cada instância do problema e cada escala de filtragem. Entretanto, se as linhas de fratura forem fractais com dimensão fixa, uma curva $\tilde{c}$ filtrada na escala λ será similar à curva c filtrada na escala λ^d. Portando a distribuição de curvatura da curva $\tilde{c}$ deve ser semelhante à curva c, comprimida horizontalmente por um fator de $1/\lambda^d$, ou seja uma gaussiana de desvio padrão σ_1/λ^d, para alguma constante σ_1. No capítulo 11 veremos que esta relação é verificada na prática.



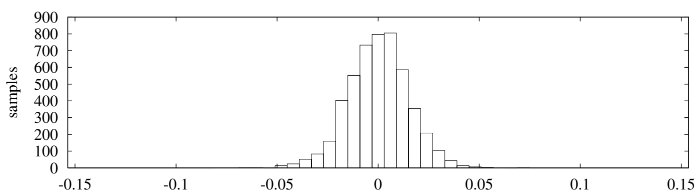
(a) escala de filtragem $\lambda = 1$ pixel, $\hat{\kappa} = 0.17$



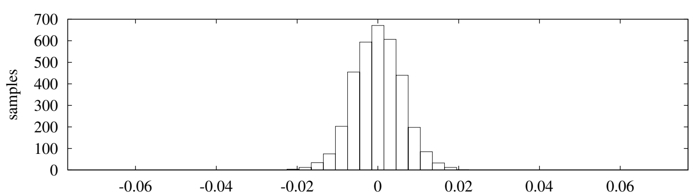
(b) escala de filtragem $\lambda = 2$ pixels, $\hat{\kappa} = 0.085$.



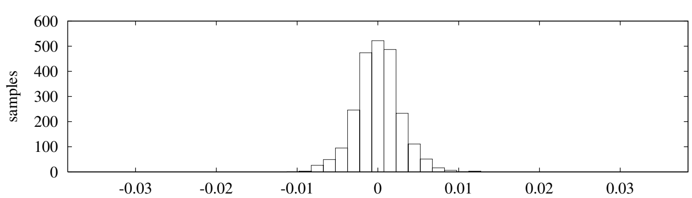
(c) escala de filtragem $\lambda = 4$ pixels, $\hat{\kappa} = 0.039$.



(d) escala de filtragem $\lambda = 8$ pixels, $\hat{\kappa} = 0.015$.

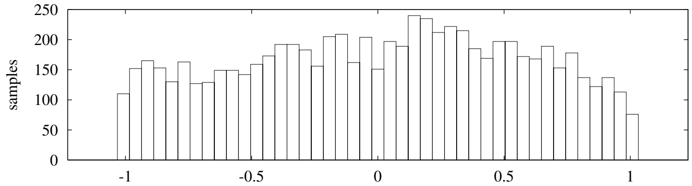


(e) escala de filtragem $\lambda = 16$ pixels, $\hat{\kappa} = 0.0060$.

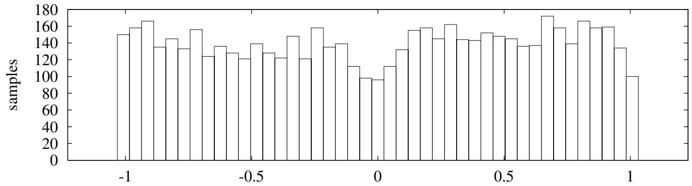


(f) escala de filtragem $\lambda = 32$ pixels, $\hat{\kappa} = 0.0028$.

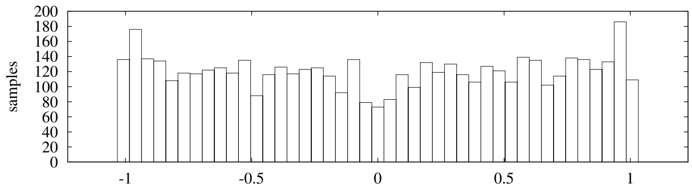
Figura 9.4: Histogramas de distribuição de curvatura κ e desvio padrão $\hat{\kappa}$ da mesma para diversas escalas de filtragem λ.



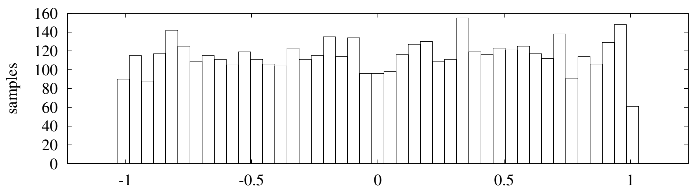
(a) escala de filtragem $\lambda = 1$ pixels.



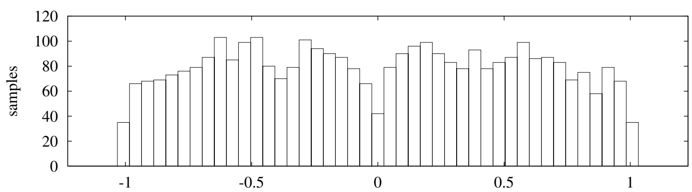
(b) escala de filtragem $\lambda = 2$ pixels.



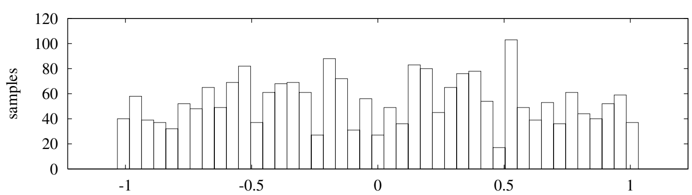
(c) escala de filtragem $\lambda = 4$ pixels.



(d) escala de filtragem $\lambda = 8$ pixels.



(e) escala de filtragem $\lambda = 16$ pixels.



(f) escala de filtragem $\lambda = 32$ pixels.

Figura 9.5: Histogramas de distribuição de $\phi(\kappa)$ para várias escalas de filtragem, usando $\hat{\kappa} = 0.18/\lambda^{1.2}$

Capítulo 10

Múltiplas escalas

Em princípio, para encontrar os pares de fragmentos adjacentes, bastaria comparar dois a dois todos os contornos observados, em todas as posições possíveis, com os métodos descritos nos capítulos 6-9. Entretanto, o alto custo do algoritmo de programação dinâmica torna inviável esta abordagem ingênua.

Mais precisamente, suponha que temos um total de N contornos observados com comprimento médio L, e que a distância entre duas amostras é δ; de modo que o número médio de amostras por contorno é $n = L/\delta$. Suponha que queremos detectar candidatos com comprimento maior ou igual a L_{min}. Na prática, podemos supor que $L_{min} = \beta L = \Omega(n^2)$, para alguma constante β não muito pequena. O número mínimo de amostras num casamento de tamanho L_{min} é então $n_{min} = L_{min}/\delta = \beta n$.

Para dois contornos dados de n amostras cada, existem em princípio $\Theta(n^2)$ candidatos a verificar, pois cada contorno tem n segmentos de comprimento n_{min}. O custo de comparar dois segmentos com n_{min} amostras cada, com o algoritmo de programação dinâmica (PD), é proporcional a n_{min}^2. Como temos N contornos, que devem ser testados dois a dois, o tempo total seria $\Omega(N^2 n^4)$.

Com um pouco de trabalho adicional, é possível chegar a reduzir o número de candidatos de cada par para $\Theta(n)$, pois pares que tem um mesmo alinhamento e uma sobreposição razoável são de certa forma redundantes, e podem ser omitidos. Com essa otimização, o tempo total seria $\Omega(N^2 n^3)$.

Lembrando que queremos resolver o problema com milhares de fragmentos, cada um deles com milhares de amostras, é obvio que o custo dessa abordagem é excessivo.

Neste capítulo, veremos como usar a *técnica de múltiplas escalas* [36] para resolver o problema do alto custo da comparação dos contornos dos fragmentos.

10.1 Técnica de múltiplas escalas

De modo geral, a técnica de múltiplas escalas consiste em resolver várias versões aproximadas de um mesmo problema, com precisão crescente, começando com a aproximação mais grosseira possível e terminando com a versão que se deseja resolver; sendo que a solução encontrada em cada etapa é usada como estimativa inicial para a resolução da etapa seguinte.

A técnica de múltiplas escalas é usada em diversos trabalhos de processamento de contornos [20, 29, 27, 40]. Porém, nestes trabalhos a técnica é geralmente usada para localizar pontos "característicos" dos contornos, como cantos ou máximos e mínimos de curvatura. Como discutido na seção 1.6, no caso dos contornos de fragmentos, a determinação de tais pontos não é viável nem muito útil. Portanto, no nosso trabalho utilizamos a técnica de múltiplas escalas diretamente na comparação dos valores da curvatura, sem a identificação prévia de pontos característicos.

10.1.1 Borramento de cantos

Quando queremos resolver o problema com contornos filtrados devemos levar em conta o "borramento" dos cantos do fragmento causado pela filtragem, e estender ou reduzir os segmentos conforme for o caso. Veja a figura 10.1.

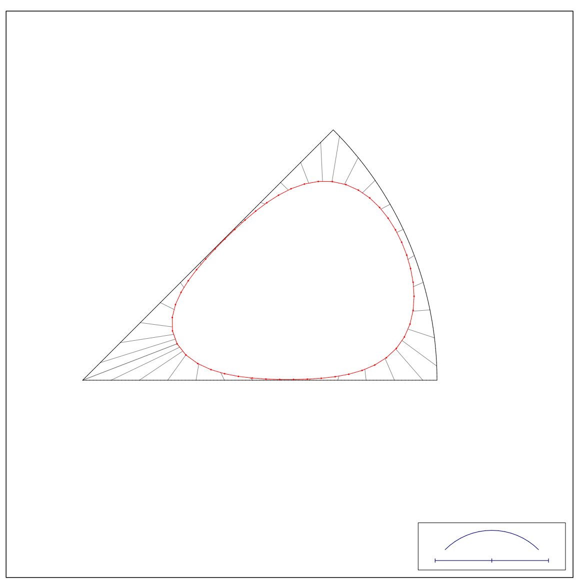


Figura 10.1: Borramento dos cantos

Mais precisamente, supõe-se que a presença de um canto num ponto $c(t)$ do contorno bruto tem um efeito significativo nos pontos $\tilde{c}(t')$ do contorno filtrado com $|t' - t| \leq \alpha\lambda$. A constante α depende da discrepância crítica ξ e da geometria do canto. Porém a dependência em ξ é logarítmica, e portanto podemos supor que α é constante. No nosso caso, após testes empíricos, adotamos $\alpha = 3.0$.

Devido a este fenômeno, um candidato reconhecível — como $(AB,\ CD)$ na figura 10.2 terá seu comprimento reduzido em $2\alpha\lambda$ quando os contornos forem filtrados na escala λ, resultando num candidato $(UV,\ XY)$.

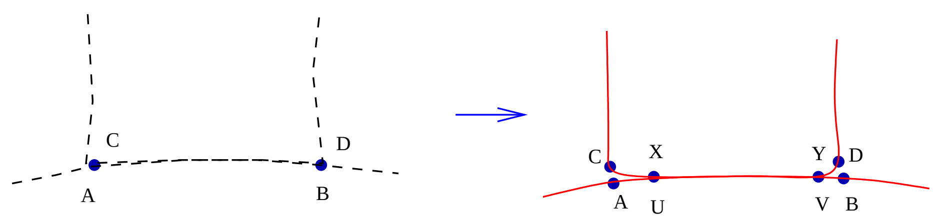


Figura 10.2: Encolhimento dos candidatos devido ao borramento dos cantos.

10.1.2 O algoritmo de casamento em múltiplas escalas

Especificamente, nós começamos resolvendo o problema da identificação de fragmentos adjacentes com uma versão grosseira dos contornos filtrados e amostrados com maior passo δ possível. Esta escala é aquela em que os segmentos procurados ainda não foram completamente obscurecidos pelo borramento dos cantos. Isto é, $\lambda_R = \lambda_1 2^R$ é a maior escala tal que o comprimento mínimo L_{min} (dos segmentos a encontrar), menos o borramento de canto $2\alpha\lambda_R$, ainda é maior que o passo de amostragem $\delta_r = \lambda_R/4$. Ou seja, $\lambda_R \leq \dfrac{L_{min}}{1/4 + 2\alpha}$, portanto

$$R = \log\left(\frac{\dfrac{L_{min}}{1/4+2\alpha}}{\lambda_1}\right)$$

Nesta escala, o número médio de amostras por contorno é $n_{min} = L_{min}/\delta_{max} = O(1)$. Portanto, a comparação exaustiva de todos os fragmentos em todas as posições possíveis pode ser feita em tempo $O(N^2)$. A desvantagem é que os candidatos encontrados nesta etapa também têm $O(1)$ amostras, portanto a comparação não é muito conclusiva. Em conseqüência, obtemos em geral um número muito grande de candidatos, da ordem de N^2.

Em seguida, resolvemos de novo o problema com um passo menor $\delta_{max}/2$, mas *testando apenas os candidatos encontrados no processamento anterior*. Repetimos este processo para passos δ cada vez menores, em progressão geométrica, terminando com um passo δ_{min} compatível com a precisão dos contornos observados.

Mais precisamente, utilizamos o seguinte procedimento:

Algoritmo 3 *Identificação multi-escala de fragmentos adjacentes*
Entradas:

- *contornos brutos* $\mathcal{C}^{(0)} = \{c_0, c_1, .. c_{N-1}\}$*;*
- *escala inicial de filtragem* λ_1 *(mais fina);*
- *comprimento mínimo* L_{min} *dos candidatos a procurar;*
- *o fator de borramento de cantos* α*.*

Saída:

- *conjunto de candidatos* $\mathcal{P}$ *provavelmente corretos.*

1. $r \leftarrow 1$; $\delta_1 \leftarrow \lambda_1/4$.
2. Enquanto $\delta_r + 2\alpha\lambda_r \leq L_{min}$
   - 2.1. Filtre os contornos $\mathcal{C}^{(r-1)}$ com um filtro geométrico de comprimento característico λ_r, obtendo contornos $\mathcal{C}^{(r)}$;
   - 2.2. $\lambda_{(r+1)} \leftarrow 2\lambda_r$; $\delta_{r+1} \leftarrow 2\delta_{r+1}$; $r \leftarrow r + 1$;
3. $r \leftarrow r - 1$;
4. Determine um conjunto de candidatos iniciais $\mathcal{P}^{(r)}$ a partir dos contornos $\mathcal{C}^{(r)}$.
5. Enquanto $r > 1$
   - 5.1. Mapeie os candidatos $\mathcal{P}^{(r)}$ das curvas $\mathcal{C}^{(r)}$ para as curvas $\mathcal{C}^{(r-1)}$ da escala $r-1$, obtendo os candidatos brutos $\mathcal{Q}^{(r-1)}$.
   - 5.2. $r \leftarrow r - 1$
   - 5.3. Refine os candidatos brutos $\mathcal{Q}^{(r)}$ usando programação dinâmica, obtendo os candidatos refinados $\mathcal{P}^{(r)}$.
   - 5.4. Elimine do conjunto $\mathcal{P}^{(r)}$ os candidatos que tiverem discrepância positiva.
   - 5.5. Dentre os candidatos restantes, em $\mathcal{P}^{(r)}$, combine os candidatos sobrepostos.
6. Mapeie os candidatos $\mathcal{P}^{(1)}$ das curvas $\mathcal{C}^{(1)}$ para as curvas $\mathcal{C}^{(0)}$, obtendo os candidatos finais $\mathcal{P}^{(0)}$.
7. Retorne os candidatos $\mathcal{P}^{(0)}$

10.1.3 Análise do algoritmo multi-escala

Observe que, a cada etapa, o número de amostras dobra, e portanto o custo de comparar dois segmentos de um candidato com o algoritmo de programação dinâmica aproximadamente quadruplica. Mesmo com a otimização descrita na seção 8.4.1, que usa o casamento anterior, o custo no mínimo dobra. Por outro lado, a cada etapa a forma dos segmentos é examinada em maior detalhe do que na etapa anterior, de modo que os candidatos falsos vão sendo progressivamente eliminados. Na última etapa, trabalhamos com o máximo de detalhes; mas, se o método funciona como esperado, os candidatos que restam são quase todos verdadeiros, e portanto são uma fração bem pequena dos candidatos iniciais.

Vamos agora tornar esta análise mais precisa. Como veremos na seção 10.1.5, a geração dos candidatos iniciais leva tempo $O(N^2)$ e normalmente gera $O(N^2)$ candidatos. Para analisar o resto do algoritmo, precisamos adotar algumas hipóteses sobre a eficácia do mesmo. Vamos supor que a detecção de candidatos verdadeiros funciona pelo menos na escala mais fina, ou seja, que o número de candidatos devolvidos na última etapa $\left|\mathcal{P}^{(1)}\right|$ é $O(M)$, onde M é o número de candidatos reconhecíveis. Pelo teorema de Euler para grafos planares [4], podemos supor que $M = O(N)$.

Vamos supor também que o número $P_r = \left|\mathcal{P}^{(r)}\right|$ de candidatos devolvidos na etapa r decai exponencialmente, com razão $1/\eta$. Ou seja, se R é o valor máximo de r no algoritmo, vamos supor que $P_r \approx P_R(1/\eta)^{R-r} = P_1\eta^{r-1}$. Uma vez que $P_1 = O(N)$ e $P_R = \Theta(N^2)$, o custo total da etapa r é proporcional a $P_r n_r^2 = P_R \eta^{r-1}$, e portanto, o custo total será dominado pelo custo da etapa $r = R$, ou seja $O(P_R) = O(N^2)$.

Para evitar problemas de "aliasing" ao amostrar os contornos, temos que respeitar o critério de Nyquist, e portanto precisamos a cada etapa trabalhar com os contornos adequadamente filtrados. Como observamos na seção 3.4, a escala λ de filtragem deve satisfazer $\lambda \geq 4\delta$.

10.1.4 Mapeamento dos candidatos

Após mapear um candidato da escala λ_r para a escala λ_{r-1} precisamos expandir os segmentos por $\alpha(\lambda_r - \lambda_{r-1})$ em cada direção. Este ajuste é ilustrado na figura 10.3: o candidato $(UV,\ XY)$, encontrado na escala mais grosseira (esquerda) deve ser estendido para $(CD,\ AB)$ após ser mapeado para a escala mais fina (direita).

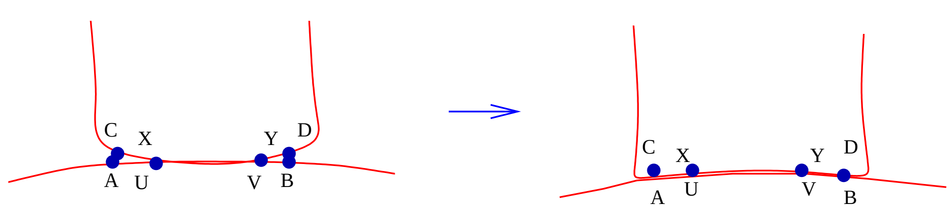


Figura 10.3: Mapeamento de um segmento de uma escala mais grosseira (a esquerda) para outra mais fina (direita).

10.1.5 Geração de candidatos iniciais

Para iniciar o processo multi-escala, temos que resolver o problema na escala mais grosseira, onde $n_{\min} \approx 1$. Uma maneira ingênua de fazer isto é aplicar o algoritmo de PD para todos os pares de segmentos possíveis com $n_{\min}$ amostras de cada lado. Porém, esta abordagem ainda teria um custo muito alto.

Em vez disso, aplicamos um método que dispensa PD e procura apenas candidatos com casamentos perfeitos (ou correspondência perfeita, seção 6.2.1). Este método é descrito a seguir:

Dadas duas cadeias circulares $a = a_0, .. a_{n_a-1}$ e $b = b_0, .. b_{n_b-1}$, onde cada uma descreve o contorno de um fragmento (circular) distinto, consideramos o conjunto de pares (i, j), onde i é um índice em a e j é um índice em b. Estes pares podem ser interpretados como os vértices de uma grade toroidal com períodos n_a e n_b. Um casamento perfeito é então uma sequência de pares (i, j) ao longo de uma mesma diagonal desta grade.

O número de diagonais d da grade toroidal é o máximo divisor comum de n_a e n_b. Cada diagonal é uma seqüência circular de tamanho $\gamma = n_a n_b / d$. Numeramos as diagonais de 0 a $d - 1$, e indexamos os elementos de cada diagonal de 0 a $\gamma - 1$. O elemento de índice r da diagonal de número t corresponde ao par de amostras (a_i, b_j) onde $i = k$ e $j = (t - k)$ módulo n_b.

Cada diagonal t é processada em separado. Para cada elemento da diagonal de índice k, calculamos um valor booleano $\mathcal{T}_k$ que informa se os pares a_i e b_j correspondentes aos n_{min} elementos consecutivos da diagonal, a partir do elemento k, têm discriminante Δ (fórmula (7.14)) negativo. Finalmente, percorremos o vetor $\mathcal{T}$, devolvendo como candidatos iniciais todos os trechos maximais de elementos verdadeiros.

Uma visualização gráfica do que ocorre é apresentada na figura 10.4. A imagem (uma matriz de tons de cinza) tem uma linha para cada amostra a_i e uma coluna b_j; a claridade do tom de cinza do pixel $m_{i,j}$ é proporcional à diferença absoluta das amostras $|a_i - b_j|$. Portanto, casamentos perfeitos promissores dão origem a linhas diagonais claras.

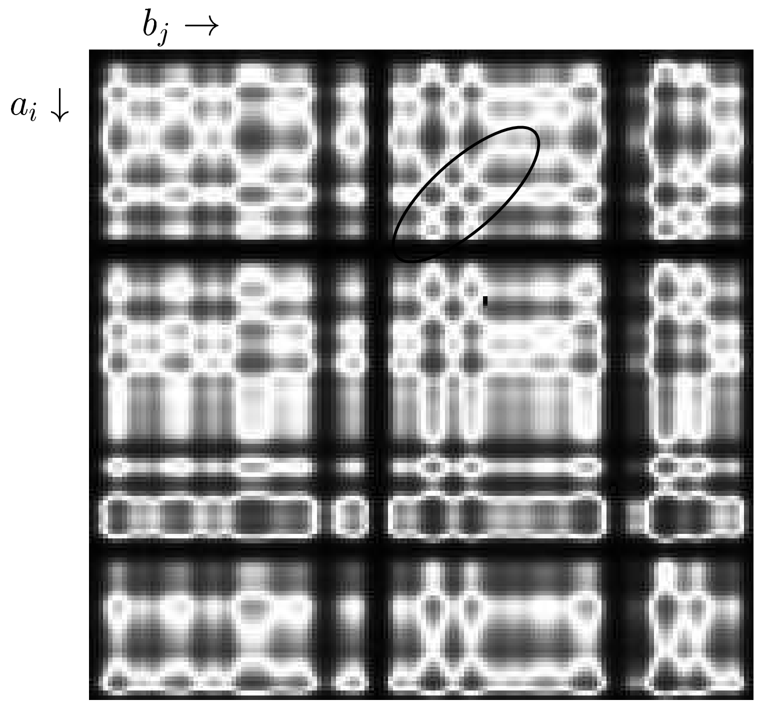


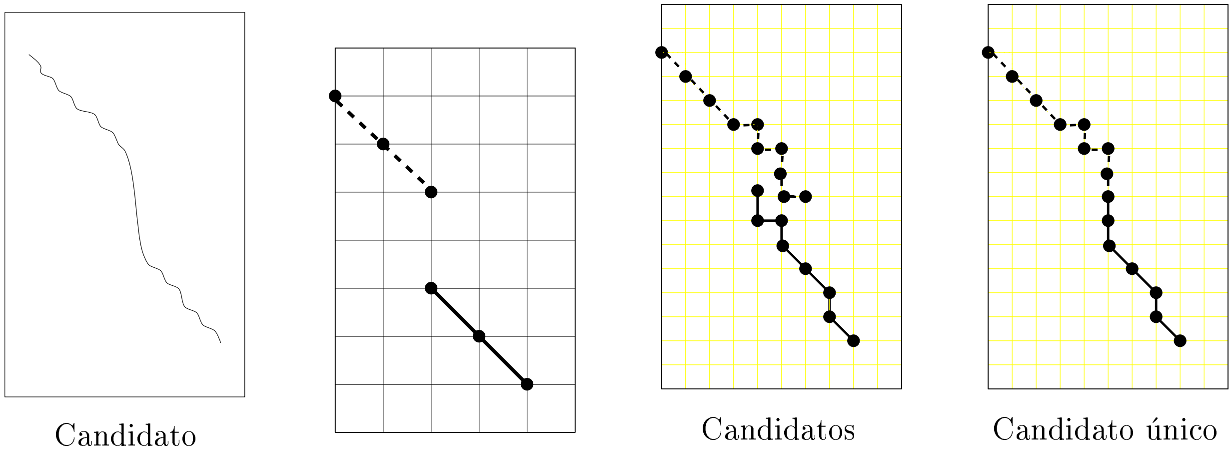
Figura 10.4: Casamentos perfeitos mais promissores. Note parte superior, mais a direita, a presença de uma diagonal mais longa e clara

Como observado acima, nesta etapa podemos supor que o tamanho dos contornos é $O(1)$ e portanto o custo total é $O(N^2)$.

10.1.6 União de candidatos sobrepostos

Uma vez que os candidatos iniciais encontrados no passo (4) estão restritos a casamentos perfeitos, é comum que um mesmo candidato reconhecível (a, b) na escala mais fina dê origem a dois ou mais candidatos parciais na escala inicial. Veja a figura 10.5(a,b). À medida que esses candidatos são mapeados e refinados pelo algoritmo de PD, eles tendem a convergir para trechos do candidato (a, b), com seções em comum. Veja a figura 10.5(c).

O objetivo do passo (5.5) é descobrir tais candidatos e combiná-los em candidatos maiores, como mostrado na figura 10.5(d).



Candidato reconhecível, na escala mais fina.

Candidatos iniciais obtidos no passo 4.

Candidatos refinados pelo passo 5.4.

Candidato único condensado no passo 5.5.

Figura 10.5: Sobreposição de candidatos.

Na implementação atual, este teste é feito verificando-se se existem pelo menos m_{min} pares no emparelhamento de cada candidato que diferem em no máximo q_{max} pares do outro candidato. Mais precisamente, se cada candidato, visto como um caminho no grafo $\vec{\mathcal{G}}$, tem pelo menos m_{min} vértices dentro da região

$$U = \{(r_k + \delta,\ s_k + \delta)\ :\ 0 \leq k \leq p-1 \text{ e } \delta \in [-q..+q]\}$$

onde $(r_k,\ s_k)$, $k = 0..p-1$ são os vértices que definem o casamento do outro candidato. Veja a figura 10.6.

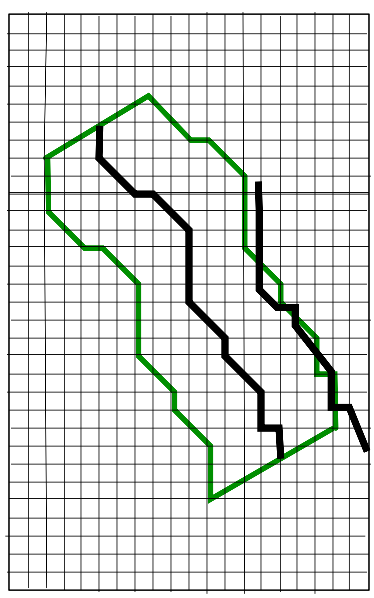


Figura 10.6: A região U usada no critério de sobreposição de candidatos.

Capítulo 11

Análise de alguns exemplos

Apresentaremos neste capítulo os resultados de alguns testes dos nossos programas. Os dados foram obtidos de fragmentos reais (se bem que especificamente gerados para este fim).

11.1 Avaliação dos resultados

Uma *solução* para o problema da reconstrução é um conjunto de candidatos supostamente verdadeiros, isto é, candidatos que correspondem a linhas de fratura entre pares de fragmentos adjacentes no objeto original. Para avaliar o desempenho do programa, introduziremos duas medidas quantitativas da qualidade dessa solução.

11.1.1 Relevância

Idealmente, uma solução S devolvida pelo programa deve conter apenas candidatos verdadeiros. Portanto, um critério para medir a qualidade de uma solução encontrada S é sua *relevância* definida por

$$\rho(S) = \frac{|S \ \cap \ T|}{|S|}$$

onde T é conjunto dos candidatos verdadeiros envolvendo os contornos dados. Este critério mede a capacidade do programa rejeitar candidatos não verdadeiros.

11.1.2 Sensitividade

O outro critério de medida do desempenho do programa é sua *sensitividade*, isto é, a capacidade do programa de reconhecer os candidatos verdadeiros.

Entretanto, o conceito de candidato verdadeiro não é adequado para este fim, porque é baseado em informações que podem não estar presentes nos dados. Se a perda de material ou os erros de aquisição são suficientemente grandes, os dois segmentos de um candidato verdadeiro podem ter formas completamente diferentes. Além disso, os candidatos verdadeiros, tal como definidos, incluem também pares de segmentos cuja linha de fratura ideal é pequena demais para que o par seja identificado sem ambigüidade (veja no capítulo 5).

Sendo assim, para fim de cálculo de sensitividade do programa é interessante definir o conceito de *candidato reconhecível*, que é o candidato verdadeiro que poderia ser identificado, por uma pessoa cuidadosa e paciente que comparasse somente os contornos digitalizados dos fragmentos, sem limite de tempo. A sensitividade de uma solução S pode ser definida então pela fórmula

$$\nu(S) = \frac{|S\ \cap\ R|}{|R|}$$

onde R é o conjunto dos candidatos reconhecíveis. Note que $\nu(S)$ varia entre 0 e 1, e que $\nu(S) = 1$ significa que o programa conseguiu encontrar todos os candidatos reconhecíveis.

Em geral, existe um conflito entre os critérios de relevância e sensitividade. Se relaxarmos os critérios de seleção dos candidatos (por exemplo, se aumentarmos a discrepância crítica ξ, ou o número máximo m_r de candidatos considerados para cada par de curvas), obteremos em geral soluções com maior sensitividade, porém menor relevância.

11.1.3 Apresentação dos resultados

Para cada teste, e cada escala apresentaremos os seguinte parâmetros:

- a discrepância crítica amostral ξ usada no refinamento,
- a penalidade para casamento imperfeito ζ,
- o número máximo permitido de candidatos por par m_r,
- o número de candidatos n_{cands} obtidos em cada escala,
- o custo em segundos da etapa de refinamento t_{ref},
- o número $|S \cap T|$ de candidatos verdadeiros no conjunto refinado,
- a relevância ρ desse conjunto, e
- a sensitividade ν.

11.2 Teste 1 - fragmentos de papel

Para este primeiro teste, o objeto original era um pedaço de papel retangular (veja a figura 11.1), de cerca de 13.0 cm por 8.0 cm, o qual foi levemente chamuscado para ficar quebradiço, e então rasgado em 20 pedaços.

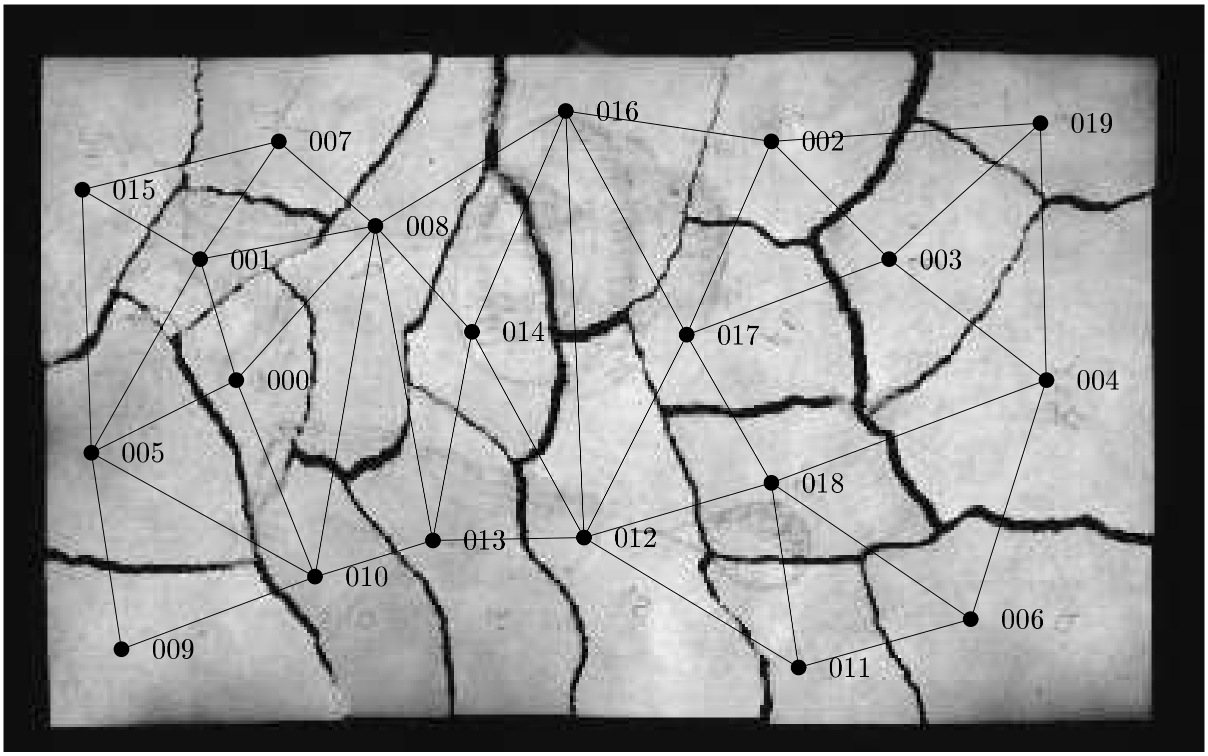


Figura 11.1: Fragmentos do teste 1, montados manualmente, e seu grafo de adjacências.

Os dados de entrada para este teste eram o conjunto dos 20 fragmentos mostrados na figura 11.2. Os fragmentos foram digitalizados diretamente com um scanner de mesa padrão (modelo UMAX UC630 Maxcolor) com 300 pontos por polegada. O comprimento médio L dos contornos extraídos era aproximadamente 1200 pixels.

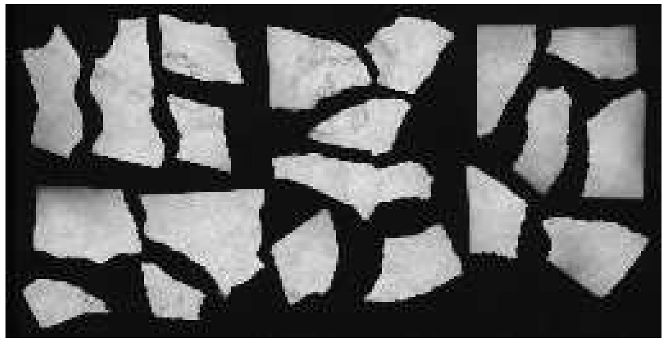


Figura 11.2: Imagens de entrada para o teste 1.

Os contornos foram filtrados e convertidos para cadeias de curvaturas codificadas, como explicado na seção 9.3. O algoritmo de alinhamento em multiplas escalas foi então aplicado a estes contornos codificados, começando na escala $\lambda_4 = 32$ pixels e terminando na escala $\lambda_0 = 2$ pixels. O comprimento mínimo alvo L_{min} dos candidatos a procurar foi fixado em 210 pixels (17.8 mm). Note-se que na escala $\lambda = 32$ o borramento dos cantos reduz tais candidatos a $210 - 6 * 32 = 18$ pixels. O conjunto de candidatos reconhecíveis com comprimento maior ou igual a L_{min} tinha 15 candidatos. A figura abaixo mostra o grafo de adjacências para este conjunto de candidatos.

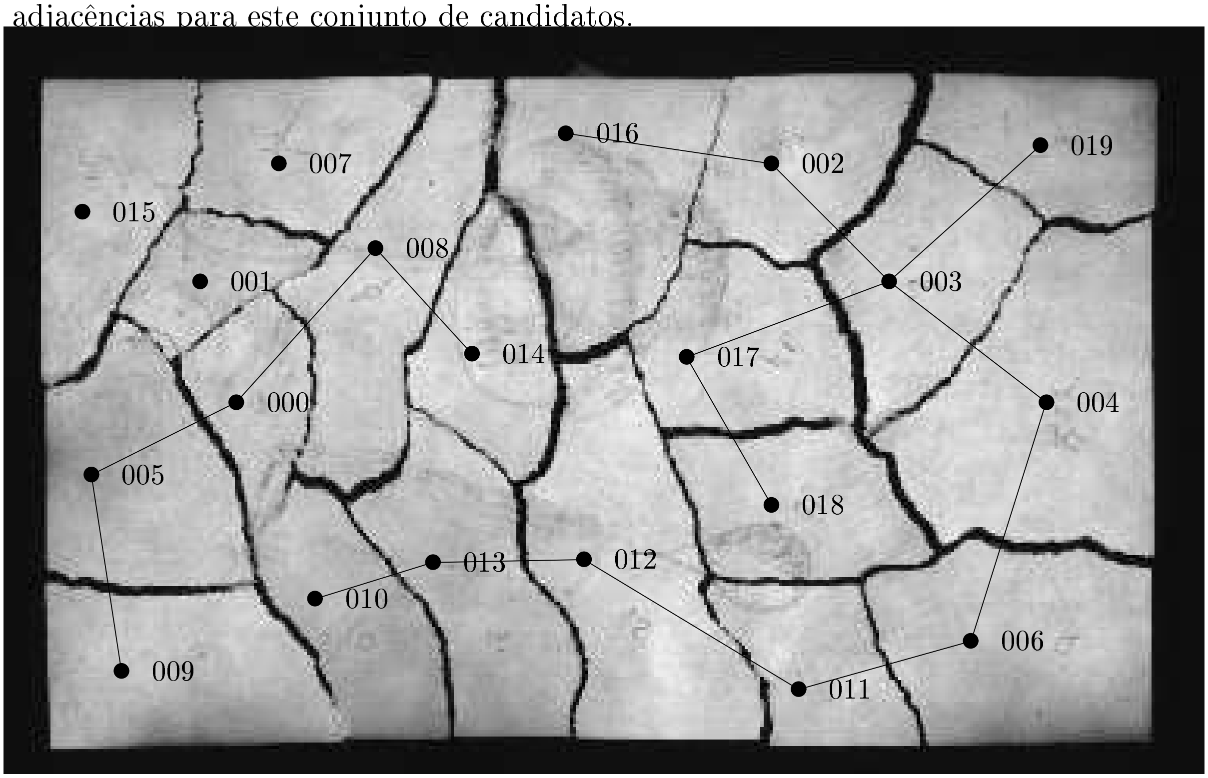


Figura 11.3: Fragmentos com o grafo de adjacências para os candidatos reconhecíveis com $L > 17.8$ mm.

A tabela abaixo mostra os parâmetros usados no algoritmo de multi-escala:

λ	ξ^2	ζ	m_r	n_{cands}	t_{ref}	$\|S \cap T\|$	ρ	ν
32	0.00085	0.00200	16	2300	3066	15	0.0065	1.00
16	0.00340	0.00840	8	212	2098	15	0.0708	1.00
8	0.01000	0.01800	4	88	349	15	0.1705	1.00
4	0.02300	0.03500	2	28	305	11	0.3929	0.73
2	0.10000	0.30000	1	14	244	9	0.6429	0.60

As figuras 11.5 e 11.4 abaixo mostram os candidatos resultantes da penúltima etapa de refinamento ($\lambda = 4$).

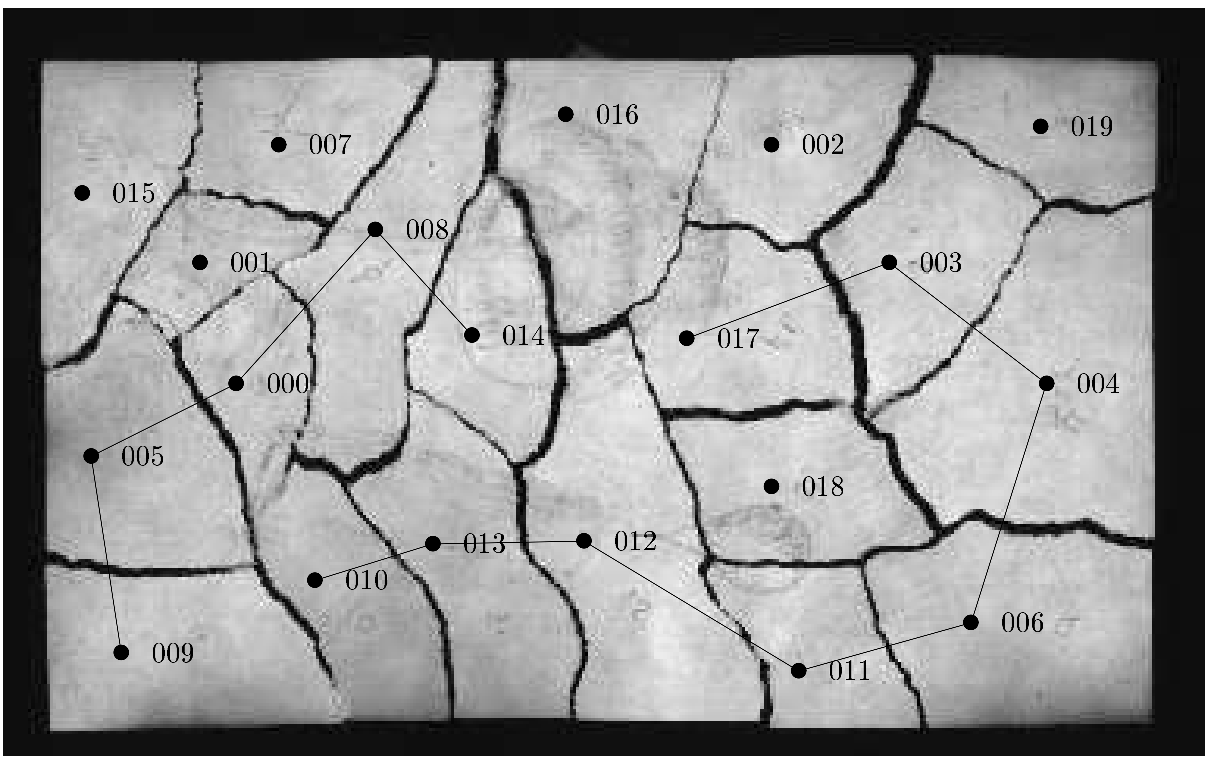


Figura 11.4: Grafo de adjacências obtido pelo programa para fragmentos do teste 1, na escala $\lambda = 4$

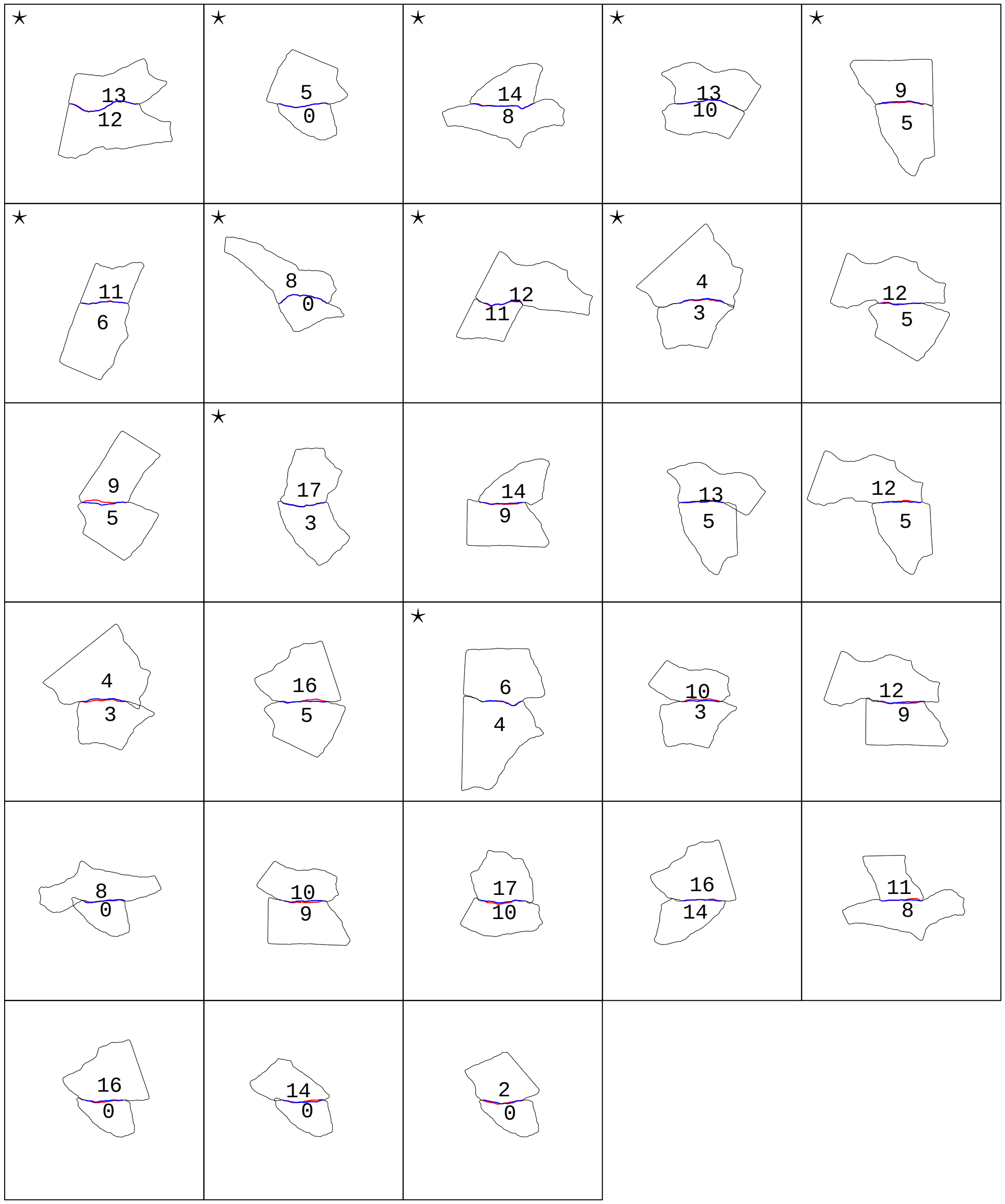


Figura 11.5: Candidatos devolvidos pelo algoritmo de alinhamento em múltiplas escalas para o teste 1, na escala $\lambda = 4$. As estrelinhas ($\star$) identificam os candidatos verdadeiros.

11.3 Teste 2 - fragmentos de cerâmica

Para o teste 2, os objetos originais eram 5 ladrilhos retangulares de cerâmica não vitrificada, medindo cerca de 25.0 cm por 6.0 cm cada um, que foram quebrados em 112 pedaços.

Veja a figura 11.6.

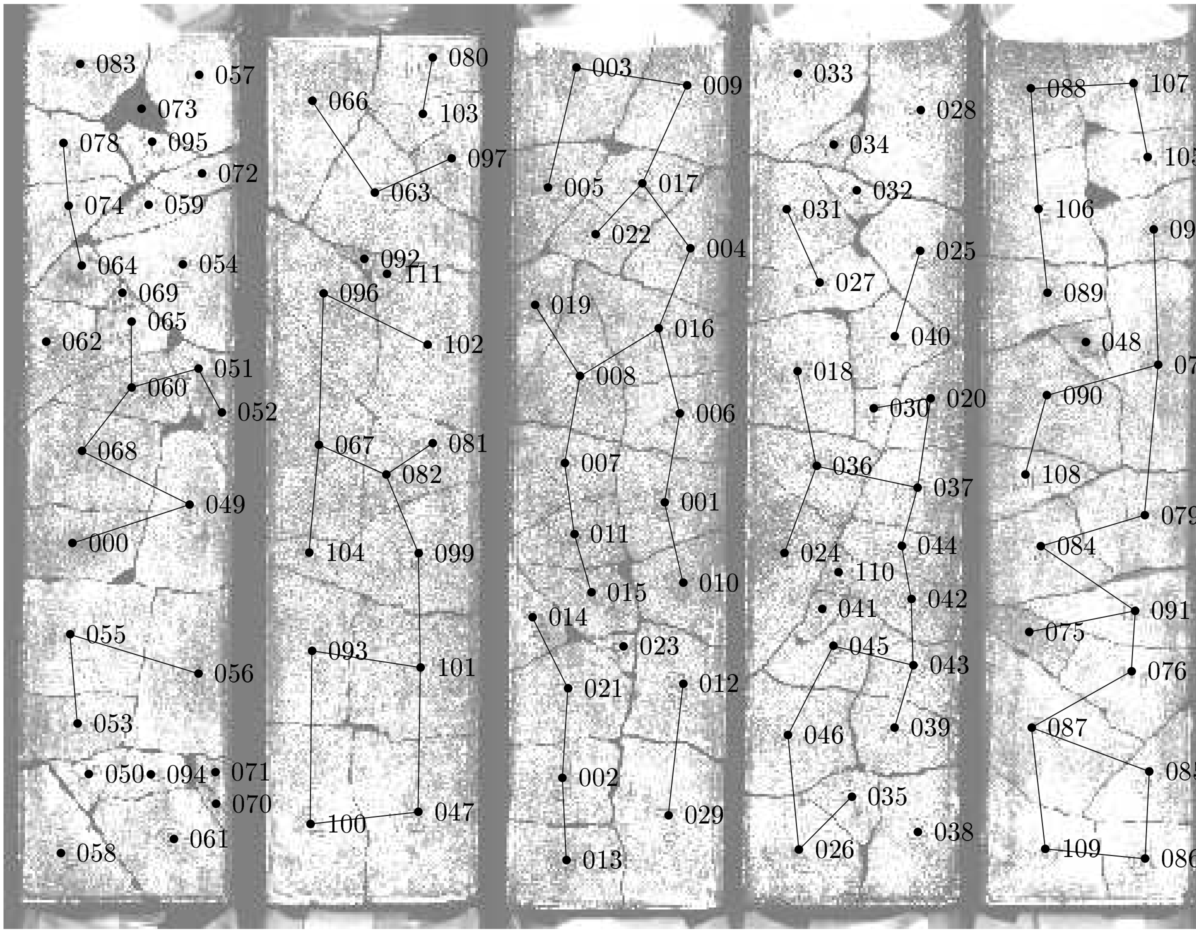
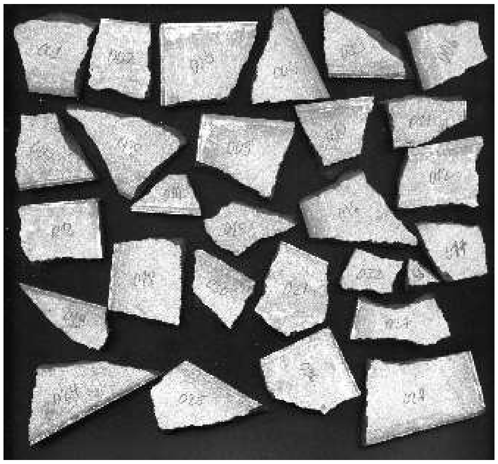
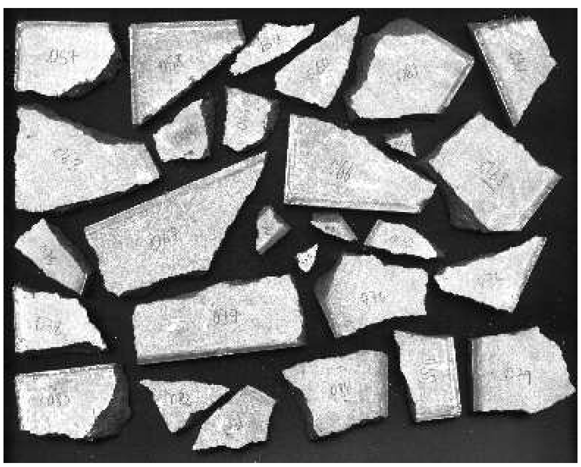
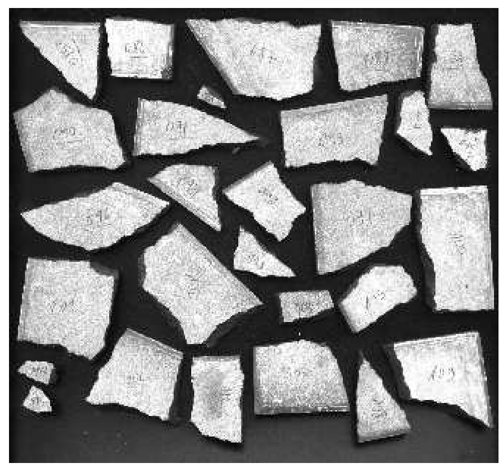


Figura 11.6: Fragmentos do teste 2, montados manualmente, e seu grafo de adjacências

Os fragmentos foram digitalizados diretamente usando um scanner de mesa padrão (modelo UMAX UC630 Maxcolor) com 300 pixels por polegada. O lado plano de cada fragmento foi esfregado levemente com giz para realçar o contraste. A figura 11.7 mostra as imagens obtidas. O comprimento médio L de cada contorno extraído destas imagens era aproximadamente 1500 pixels.




(a) (b)

(c) (d)

Figura 11.7: Imagens de entrada para o teste 2.

Os contornos extraídos foram filtrados e convertidos para representação por curvatura, conforme explicado na seção 9.3. A figura 11.8 mostra o gráfico do desvio padrão $\hat{\kappa}_r$ dos valores da curvatura, para cada escala de filtragem λ_r. Pode-se verificar que estes valores obedecem razoavelmente a fórmula

$$\hat{\kappa}_r = 0.75/\lambda_r^{1.21} \tag{11.1}$$

o que pode ser interpretado como indicação de que a dimensão fractal das linhas de fratura é próxima a 1.2 nas escalas de 1 a 16 pixels.

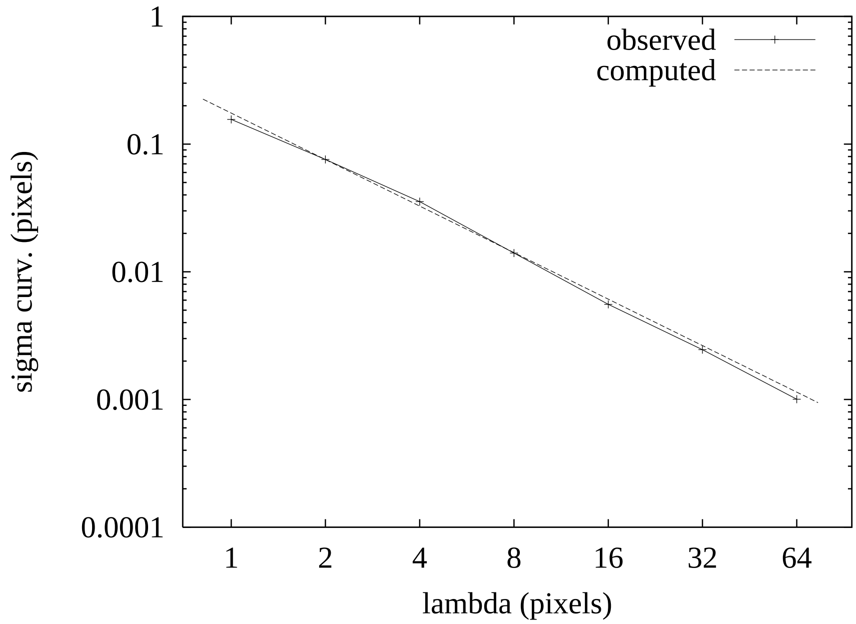


Figura 11.8: Gráfico do desvio padrão $\hat{\kappa}$ dos valores da curvatura em função da escala de filtragem λ (escala logarítmica). A linha pontilhada é o gráfico da fórmula (11.1).

O algoritmo de alinhamento em múltiplas escalas foi aplicado então para estes contornos codificados, começando na escala $\lambda_4 = 32$ pixels e terminando na escala $\lambda_0 = 2$ pixels. Foram feitas duas execuções do programa, indicadas a seguir por $2A$ e $2B$, com parâmetros ligeiramente diferentes.

11.3.1 Teste 2A

Nesta rodada o comprimento mínimo L_{min} dos candidatos a procurar foi fixado em 250 pixels (20.8 mm). Note que na escala $\lambda = 32$ estes candidatos são reduzidos a $250 - 2 * 32 = 58$ pixels. Na figura 11.9 mostramos o grafo de adjacências dos 53 candidatos reconhecíveis com comprimento maior ou igual a L_{min}.

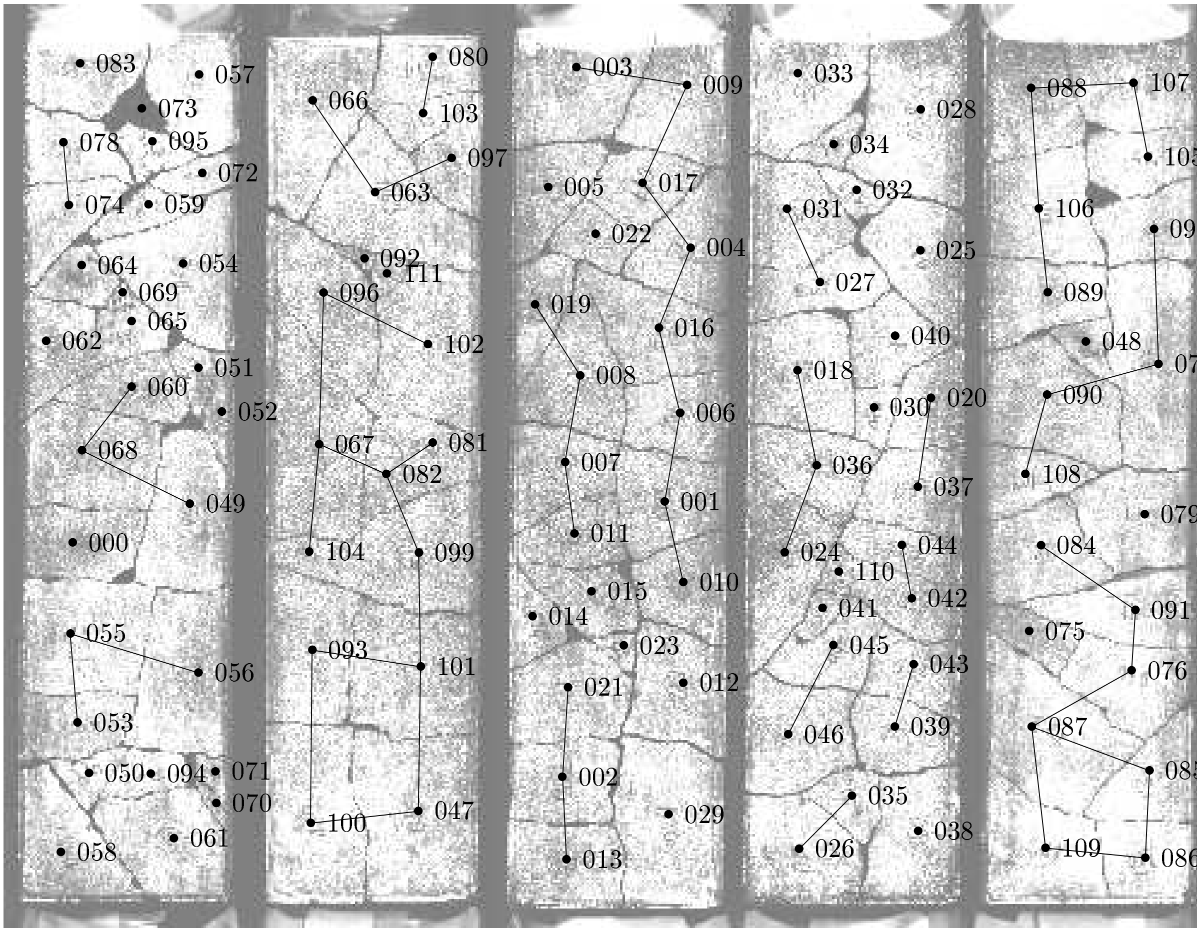


Figura 11.9: Grafo de adjacências dos candidatos reconhecíveis com $L_{min} > 250$ pixels.

O número de candidatos iniciais gerado pelo programa foi 87773. A tabela abaixo mostra os parâmetros usados no algoritmo de multi-escala:

λ	ξ^2	ζ	m_r	n_{cands}	t_{ref}	$\lvert S \cap T \rvert$	ρ	ν
32	0.00070	0.00021	16	12950	14200	45	0.0035	0.8491
16	0.00220	0.00440	8	342	7484	35	0.1023	0.6604
8	0.00800	0.01400	4	67	668	27	0.4030	0.5094
4	0.02400	0.04000	2	28	321	23	0.8214	0.4340
2	0.06500	0.10000	1	22	419	19	0.8636	0.3585

A figura 11.10 abaixo mostra os 22 candidatos sobreviventes na escala $\lambda = 2.0$ pixels.

Figura 11.10: Os 22 candidatos devolvidos pelo algoritmo de alinhamento em múltiplas escalas para o teste 2A na escala $\lambda = 2$ pixels. As estrelinhas ($\star$) identificam candidatos verdadeiros.

A figura 11.11 mostra o subconjunto das arestas do grafo de adjacencia que correspondem aos pares verdadeiros encontrados pelo algoritmo.

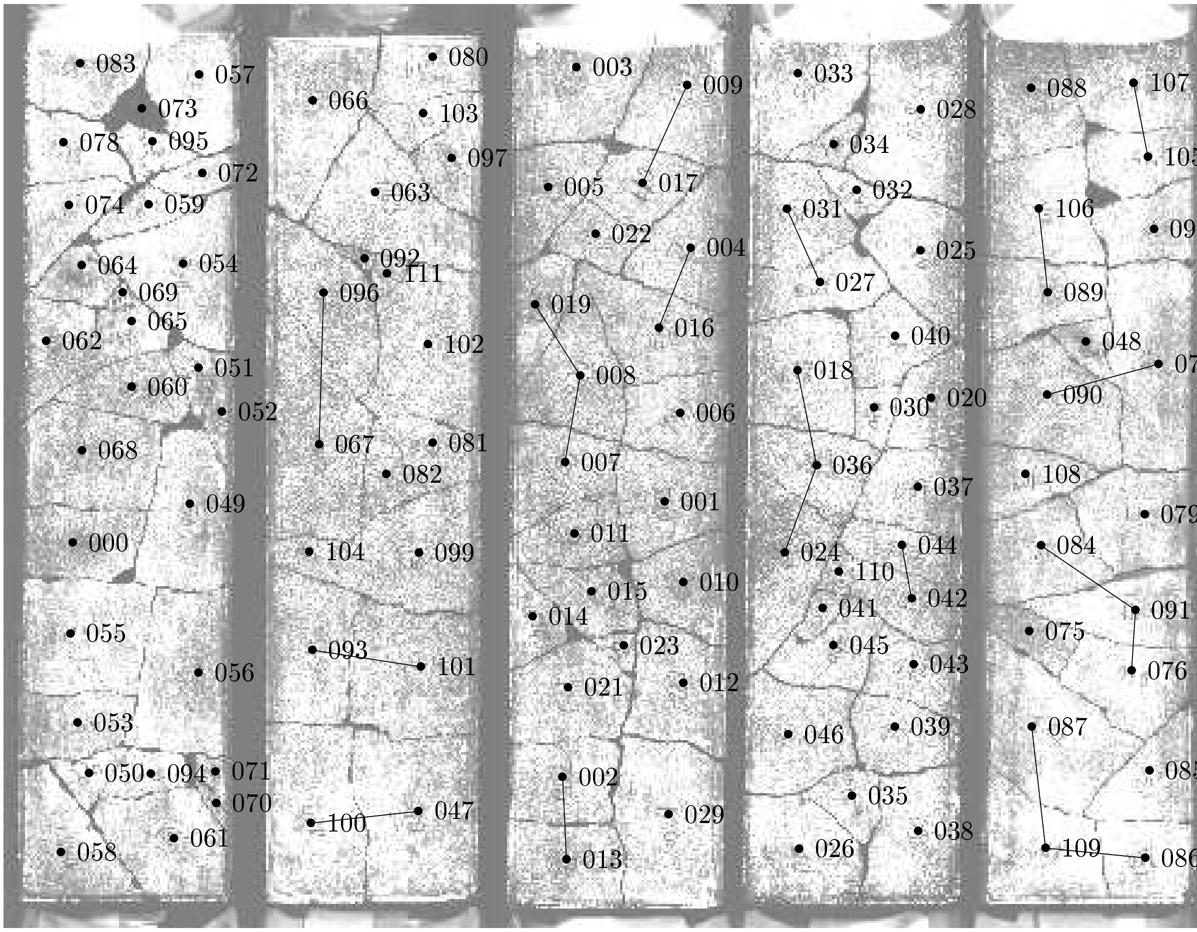


Figura 11.11: Fragmentos do teste 2A, montados manualmente, e o subgrafo de adjacências encontrado pelo programa na escala $\lambda = 2$ pixels.

11.3.2 Teste 2B

A tabela abaixo resume os resultados de uma segunda execução do procedimento multi-escala para os fragmentos do teste 2, com critérios mais relaxados (valores maiores para ξ) e um comprimento menor para L_{min} (210 pixels). (Na escala $\lambda = 32$. estes candidatos ficam reduzidosa $210 - 6 * 32 = 18$ pixels.) O número de candidatos iniciais gerado pelo programa foi 166626.

λ	ξ	ζ	m_r	n_{cands}	t_{ref}	$\lvert S \cap T \rvert$	ρ	ν
32	0.00080	0.00210	16	64743	31091	57	0.0009	0.7703
16	0.00260	0.00440	8	7797	52551	70	0.0090	0.9459
8	0.00900	0.01400	4	1814	10095	64	0.0353	0.8649
4	0.02600	0.04000	2	442	4058	50	0.1131	0.6756
2	0.07050	0.10000	1	277	2542	46	0.1661	0.6217

Note-se que a sensitividade ν aumentou (de 0.3585 para 0.6217) em comparação com o teste anterior, enquanto que a relevância ρ diminuiu (de 0.8636 para 0.1661). As figuras 11.13 e 11.14 mostram os 60 primeiros candidatos devolvidos na última etapa, e a figura 11.12 apresenta o grafo de adjacências correspondente.

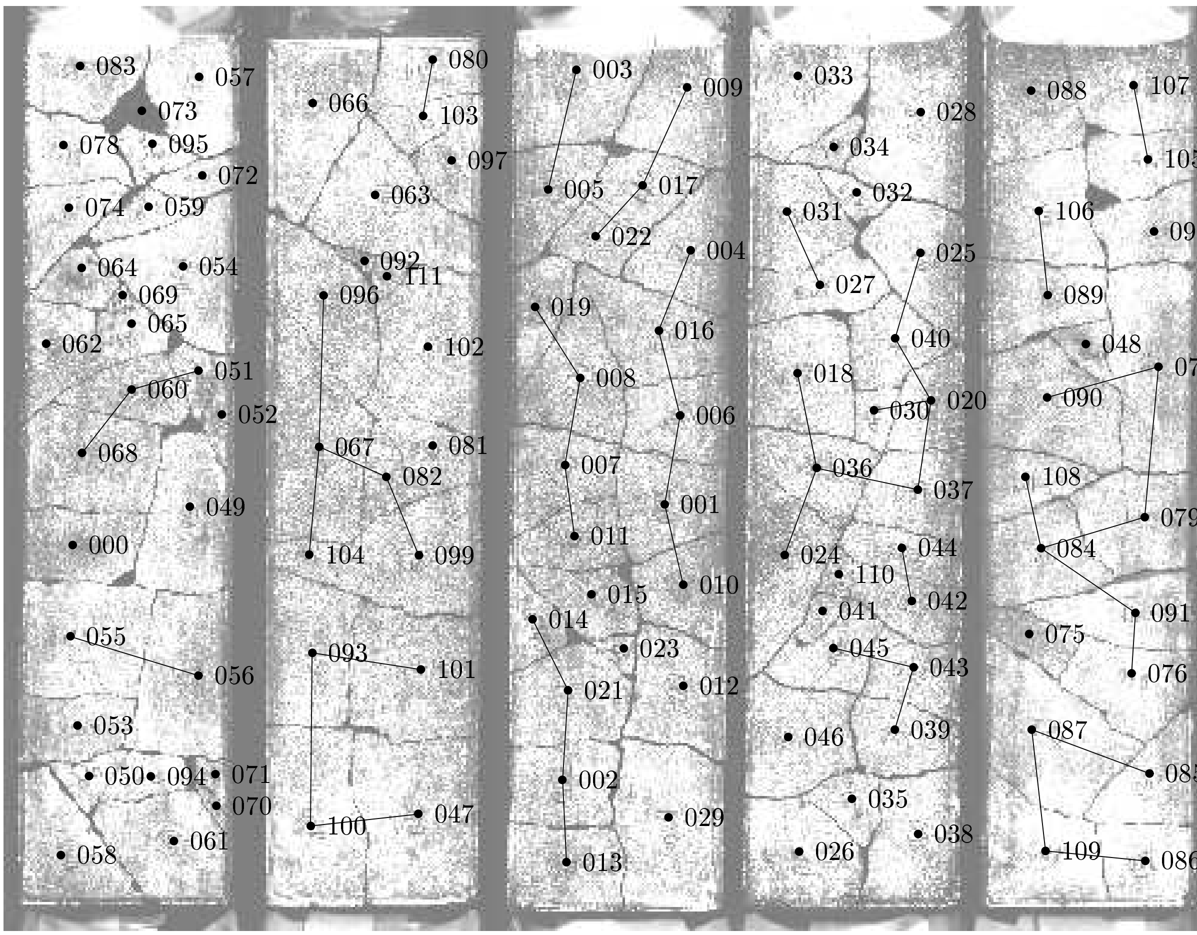


Figura 11.12: Fragmentos do teste 2B, montados manualmente, e o subgrafo de adjacências na escala $\lambda = 2$ pixels.

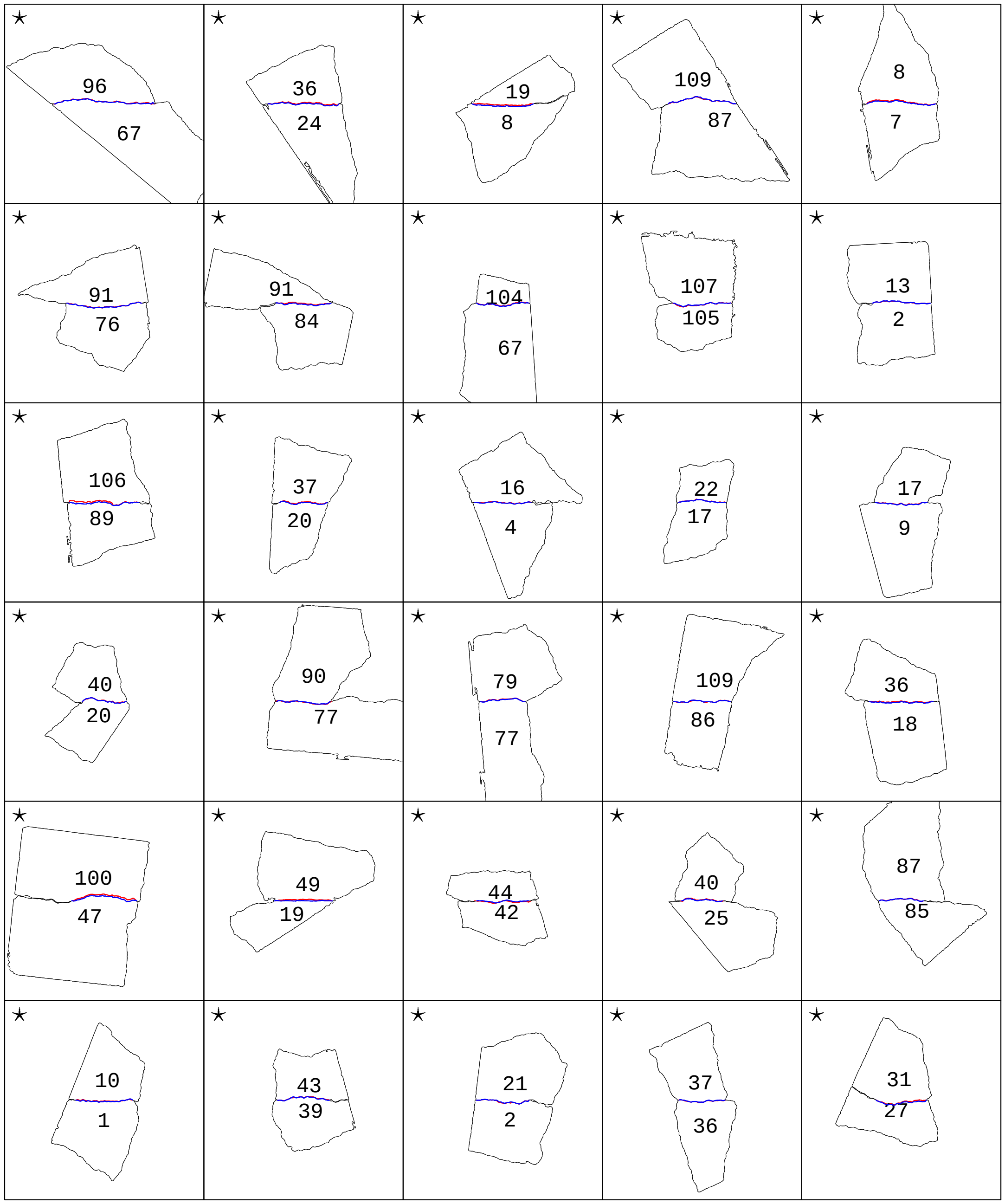


Figura 11.13: Os 30 primeiros candidatos devolvidos pelo algoritmo de alinhamento em múltiplas escalas para o teste 2B na escala $\lambda = 2$ pixels. As estrelinhas ($\star$) identificam os candidatos verdadeiros.

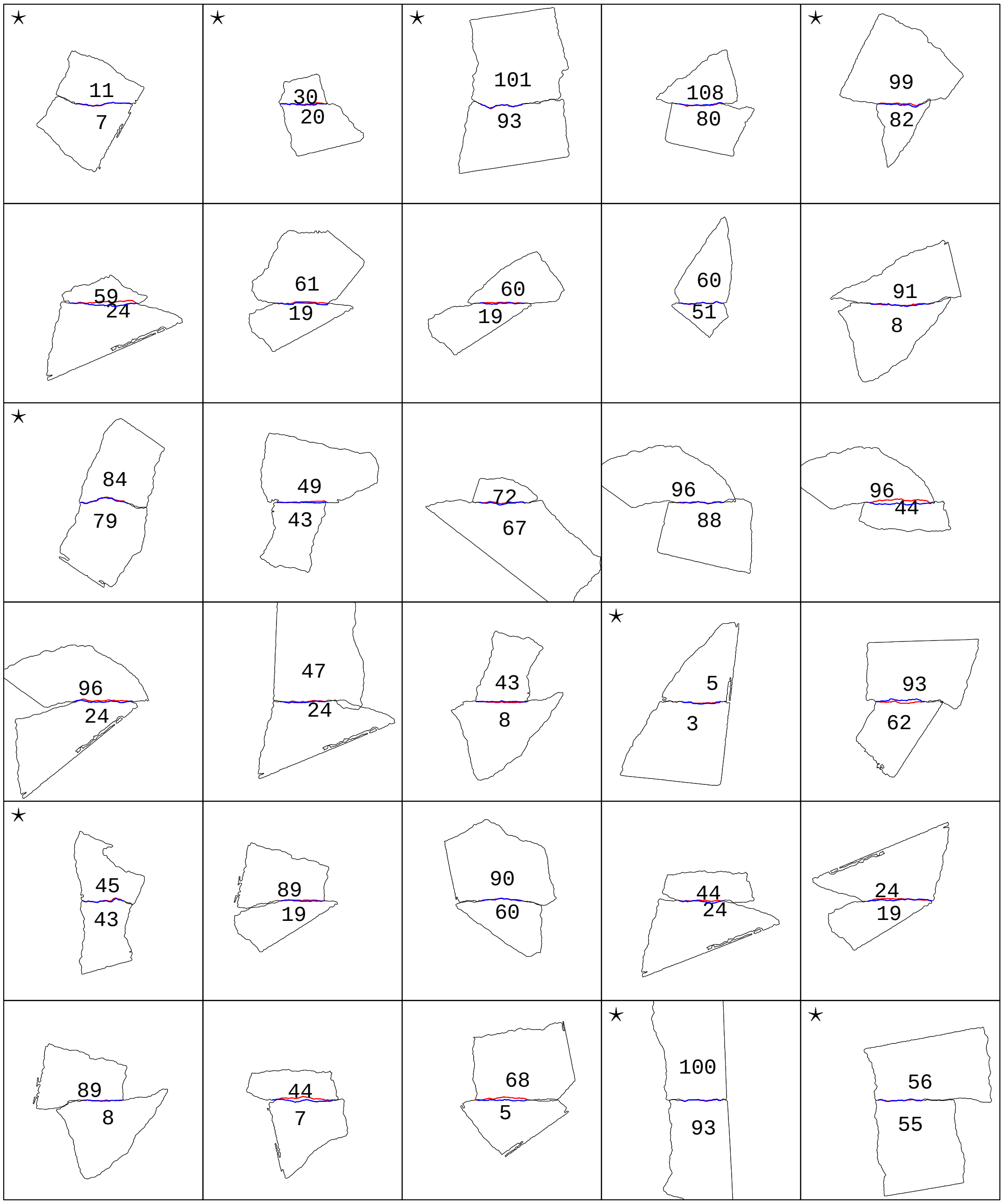


Figura 11.14: Os próximos 30 candidatos devolvidos pelo algoritmo de alinhamento em múltiplas escalas para o teste 2B na escala $\lambda = 2$ pixels. As estrelinhas ($\star$) identificam os candidatos verdadeiros.

Capítulo 12

Conclusões e sugestões para trabalhos futuros

Os experimentos apresentados no capítulo 11, apesar de seus tamanhos modestos, demonstram a possibilidade de mecanicamente identificar segmentos adjacentes a partir de suas linhas de contorno. Eles também validam a premissa básica do método de casamento em múltiplas escalas, em que o número de candidatos sobreviventes diminui muito rapidamente a cada refinamento com aumento de resolução. O custo computacional de nosso programa de casamento em múltiplas escalas ainda é grande (cerca de 2 horas para os 112 fragmentos do teste 2), mas será certamente fácil reduzir seu custo com pequeno esforço de programação.

12.1 Contribuições para a área

Para identificar pares adjacentes, num conjunto grande segmentos, tivemos que resolver vários problemas, alguns dos quais não têm uma solução ótima conhecida. Antes de mais nada tivemos que descobrir se seria possível atingir nosso objetivo apenas a partir dos contornos dos fragmentos. Não encontramos na literatura nada que garantisse este ser um problema possível de ser resolvido ou não, de modo que tivemos que desenvolver uma técnica de estimação da quantidade de informação contida em linhas de fratura (capítulo 5).

Uma vez mostrado que o problema pode ser resolvido a princípio, enfrentamos o problema da filtragem dos contornos. As técnicas usualmente empregadas, baseadas em filtragem paramétrica, são afetadas pelo efeito do ruído no comprimento da curva, e produzem resultados diferentes dependendo da orientação do segmento. Para solucionar este problema desenvolvemos o algoritmo de filtragem geométrica com parametrização "a posteriori" (capítulo 4).

Outros problemas que tivemos que resolver foram a geração dos candidatos iniciais, o alinhamento de segmentos correspondentes, e mapeamento dos segmentos em várias escalas de filtragem que também demandaram soluções originais.

12.2 Sugestões e trabalhos futuros

Uma necessidade óbvia é estender estas técnicas para fragmentos com superfícies curvas (vasos, estátuas, etc) que constituem a maioria dos fragmentos encontrados em escavações arqueológicas.

Esta extensão exigirá o uso de técnicas de visão estereoscópica para extração das linhas de fratura, e o estudo e representação de invariantes locais para curvas no $\mathbf{R}^3$ (incluindo por exemplo torsão, além de curvatura).

Também é necessário melhorar a implementação para torná-la mais eficiente e mais fácil de usar, incluindo a escolha automática de valores para os vários parâmetros, que atualmente devem ser fornecidos pelo usuário.

Outra direção para trabalho futuro é adaptar os métodos apresentados aqui para outras áreas tais como: medicina legal (para reconstrução de ossadas) e biologia computacional (na identificação de cadeias de DNA).

Apêndice A

Descrição dos programas

Nós implementamos os métodos e algoritmos apresentados neste trabalho usando a linguagem Modula-3 [12], [24]. A programação totalizou cerca de 20.000 linhas de código. Neste capítulo vamos descrever os principais componentes desta implementação. Maiores detalhes podem ser obtidos nas interfaces dos módulos das bibliotecas e nos comentários dos programas.

A.1 Programas principais:

Nesta seção, descreveremos os programas necessários para processar uma instância do problema, desde as imagens iniciais até a lista dos pares de fragmentos adjacentes encontrados.

`PZSplit`

Este programa separa uma imagem de N fragmentos em N imagens individuais, usando uma variante do algoritmo de propagação descrito na seção 2. O programa permite escolher o nível de cinza δ_{sep}, e a cor de fundo δ_{min} (suposta uniforme). As imagens são tons de cinza, no formato pgm usado pelo pacote `PBMplus`.

`PZBoundary`

Este programa extrai o contorno de um único fragmento a partir de uma imagem do mesmo, segundo o algoritmo descrito na seção 2.2. O programa pode opcionalmente gravar o contorno externo (a_i) ou interno (b_i), em vez do contorno interpolado (p_i).

PZFilter

Este programa filtra uma curva em várias escalas λ_{min}, $\alpha\lambda_{min}$, $\alpha^2\lambda_{min}, .. \alpha^{n-1}\lambda_{min}$, usando o algoritmo de filtragem geométrica descrito na seção 4.6. Além dos contornos iniciais (produzidos pelo `PZBoundary`) este programa recebe a escala inicial λ_{min}, o passo α, e o número de escalas n. Além disso, devem ser fornecidos o número r de estágios intermediários do algoritmo de filtragem, e a tolerância ϵ para determinação de sua convergência. Uma saída colateral deste programa são os arquivos que fornecem a correspondência entre as parametrizações do contorno nas diferentes escalas de filtragem.

PZComputeVelAcc

Este programa parte de um contorno amostrado e calcula a velocidade e a aceleração em cada amostra na parametrização natural.

PZComputeCurvature

Este programa calcula a representação por curvatura de um contorno dado como descrito no capítulo 9 a partir dos arquivos produzidos por `PZComputeVelAcc`.

PZEncodeCurvature

Este programa codifica as curvaturas de cada contorno conforme descrito na seção 9.3. O parâmetro $\hat{\kappa}_r$ deve ser fornecido ao programa.

PZGetStraightSegs

Dado um conjunto de contornos, este programa produz um arquivo de segmentos dos mesmos que não devem ser considerados no processo de casamento, por terem curvaturas muito próximas a zero. Conforme observado na seção 1.6 estas partes são provavelmente bordas externas; e, de qualquer forma, elas não podem ser casadas com confiança, pois produziriam um número muito grande de candidatos.

Este programa deve ser aplicado a contornos filtrados numa escala intermediária, pois nas escalas mais finas os erros de aquisição do contorno podem impedir o reconhecimento de bordas externas.

PZMapSegs

Este programa mapeia um conjunto de segmentos de contornos, de uma escala de filtragem λ_i para outra escala de filtragem λ_j. Para isto, é necessário que os contornos correspondentes já tenham sido filtrados nas escalas λ_i e λ_j. Ele é usado no algoritmo de reconhecimento em múltiplas escalas, e também para mapear os segmentos retos obtidos por `PZGetStraightSegs` para outras escalas. Ele recebe como entrada, além dos segmentos numa escala λ_i, o fator de borramento (seção 10.1.1), e os arquivos que definem a correspondência entre os contornos na escala λ_i e λ_j (que são produzidos pelo `PZFilter`, como subproduto do processo de filtragem).

`PZGetInitialCands`

Este programa lê um conjunto de contornos e gera o conjunto dos candidatos iniciais dos mesmos. Ele encontra apenas candidatos definidos por alinhamentos perfeitos, conforme o algoritmo descrito na seção 10.1.5. Neste processo são desconsiderados os segmentos produzidos pelo programa `PZGetStraightSegs`.

`PZMapCands`

Este programa análogo ao `PZMapSegs`, mapeia o conjunto de candidatos de uma escala de filtragem λ_i para outra escala λ_j.

`PZRefineCands`

Este programa refina um conjunto de candidatos aproximados, usando o algoritmo de programação dinâmica a partir do centro ("centrífugo"), conforme descrito na seção 8.3.2, aplicado às representações por curvatura dos contornos. Além disso, ele classifica os candidatos refinados pelo valor da discrepância total, e elimina os mais discrepantes, como discutido na seção 9.2. Além dos candidatos aproximados e das cadeias de curvatura, o programa precisa dos seguintes parâmetros: o valor crítico de corte ξ^2; o tamanho mínimo de um candidato; o número máximo de candidatos; e o número máximo de candidatos por curva.

A.2 Utilização dos programas

Os programas são tipicamente usados segundo os fluxogramas abaixo. No início, os fragmentos digitalizados em conjunto são separados com o `PZFilter`, e cada fragmento tem seu contorno extraído com o `PZBoundary`. Feito isto, cada contorno é então filtrado em diferentes escalas de filtragem, com uma única execução do `PZFilter` (figura A.1).

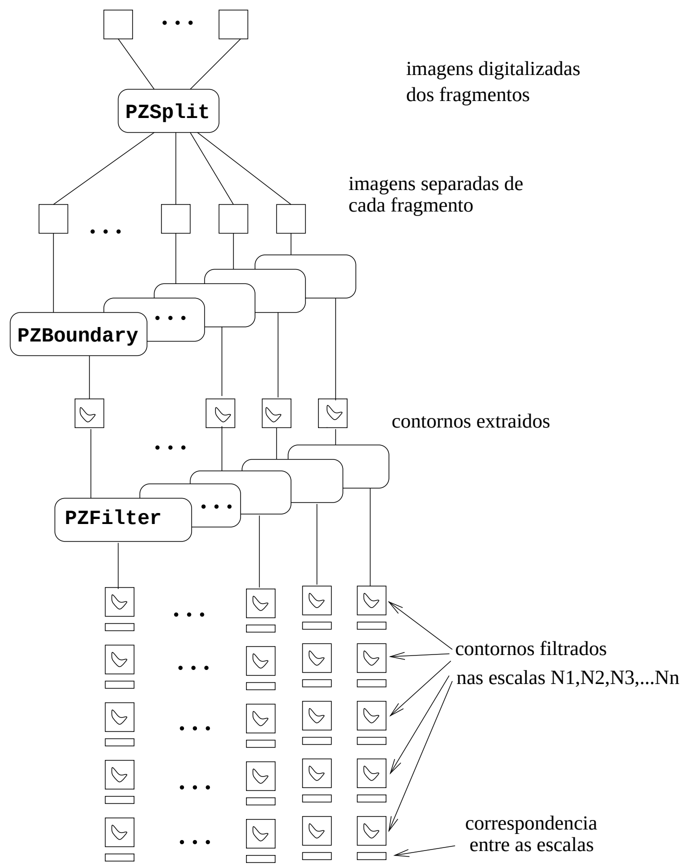


Figura A.1: Diagrama mostrando a extração dos contornos e filtragem em diferentes escalas.

Para cada contorno filtrado, em cada escala, é calculada a representação por curvatura do mesmo usando os programas `PZComputeVelAcc` e `PZComputeCurvature`. A seguir, o programa `PZCurvHist` é usado para calcular o valor de $\hat{\kappa}_r$, utilizado na função de quantização ϕ. As curvaturas são então codificadas pelo programa `PZEncodeCurvature`, conforme descrito na seção 9.3.

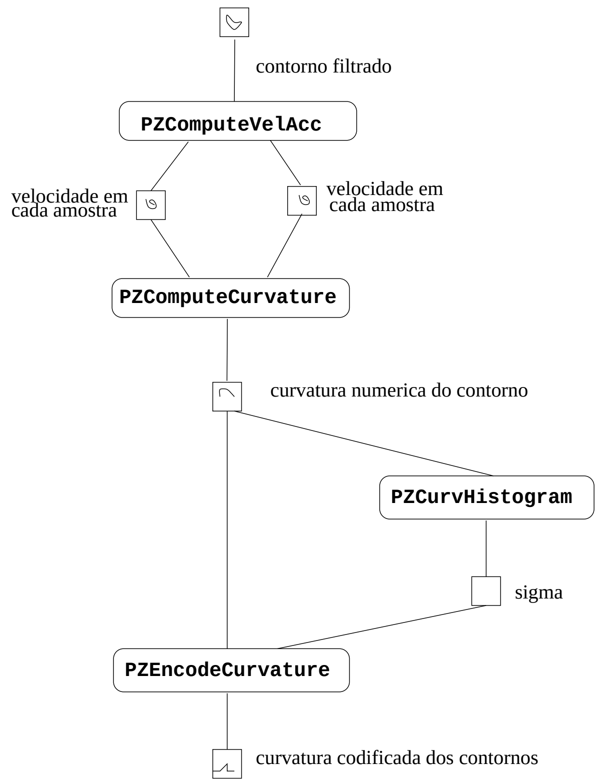


Figura A.2: Diagrama mostrando o cálculo e a codificação da curvatura para um contorno.

Feito isso, usamos o programa `PZGetStraigthSegs` para extrair segmentos do contorno, que provavelmente correspondem a bordas externas, na escala mais conveniente para esta identificação. Em seguida, estes segmentos são mapeados para escala mais grosseira com o `PZMapSegs`. Os candidatos iniciais são então gerados com o `PZGetInitialCands`, ignorando-se os segmentos retos.

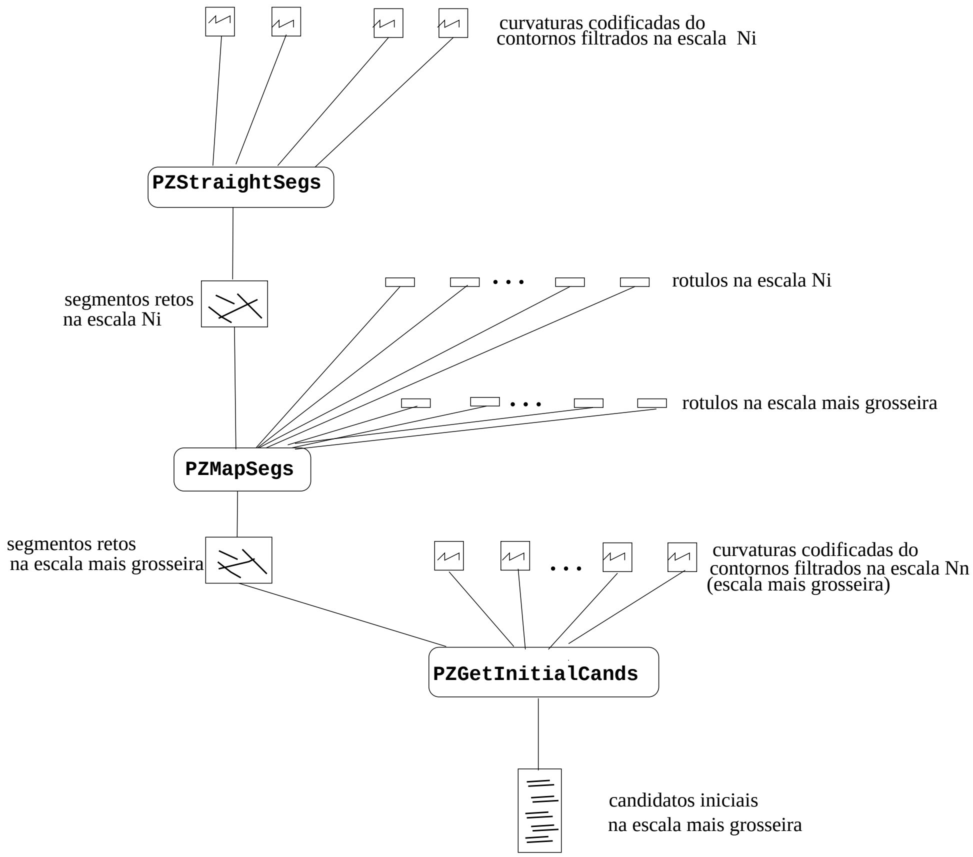


Figura A.3: Diagrama mostrando a extração dos segmentos de borda e a inicialização do arquivo de candidatos.

Uma vez obtido um conjunto inicial de candidatos, usamos alternadamente os programas `PZMapCands`, para "traduzir" os candidatos de uma escala para a escala mais fina seguinte, e `PZRefineCands` para encontrar o melhor casamento, e eliminar os candidatos que se revelam pouco promissores. A figura A.4 ilustra uma destas etapas.

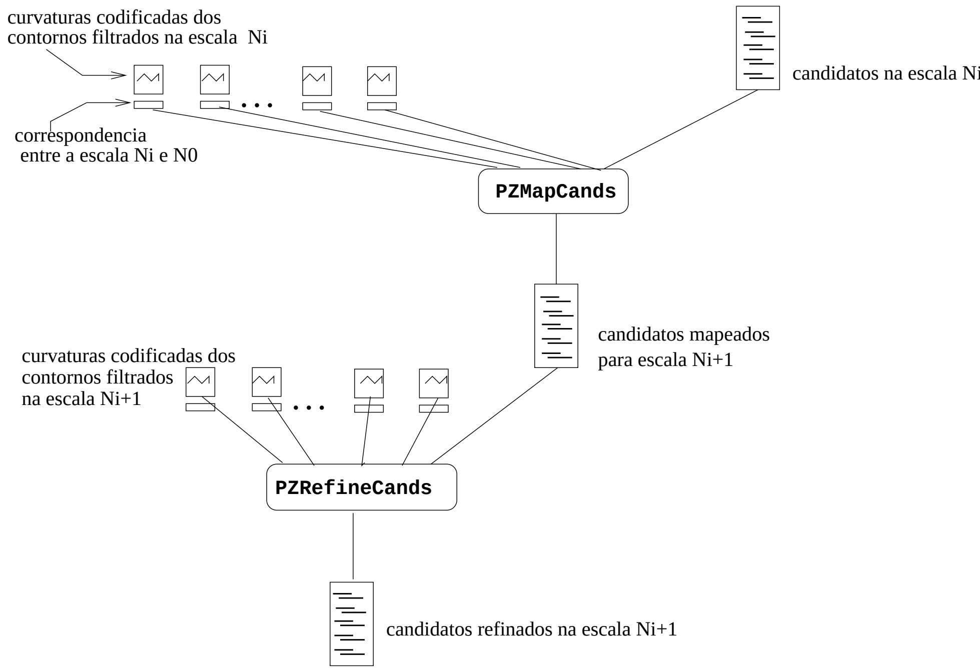


Figura A.4: Diagrama mostrando o refinamento dos candidatos.

Este processso de refinamento/mapeamento dos candidatos é repetido até que os candidatos que restam sejam na sua maioria corretos, ou até atingir a escala de detalhes a mais fina possível.

A.3 Programas de impressão e traçado

Estes programas não são estritamente necessários para a solução do problema, mas são bastante úteis para monitorar o processo. Os programas de desenho podem gerar saída no formato Postscript (.ps) ou Encapsulated Postscript (.eps) [1].

`PZDraw`

Este programa desenha o contorno de um fragmento e foi usado, por exemplo, para gerar a figura 9.2.

`PZDrawSegs`

Este programa desenha um ou mais segmentos de contorno.

`PZDrawCands`

Este programa é usado para desenhar um conjunto de candidatos, opcionalmente com seus casamentos. As figuras 11.5 e 11.10 foram geradas por este programa.

`PZPrintCands`

Este programa imprime um conjunto de candidatos num formato legível. Para cada candidato são impressos os números dos dois contornos, o índice de início e fim de cada segmento, a discrepância e o casamento (codificado de maneira compacta). Opcionalmente, o programa lê os contornos codificados por curvatura e imprime os segmentos dos mesmos correspondentes a cada candidato.

`PZDrawCandGrid`

Este programa desenha a matriz que representa o grafo $\vec{\mathcal{G}}$ (seção 8.2), e o caminho correspondente ao casamento de cada par de segmentos do conjunto de candidatos. A figura 10.6 é semelhante a gerada por este programa.

A.4 Programas de avaliação e estatística

Os programas a seguir são utilizados principalmente para ajudar a escolha dos parâmetros usados pelos programas principais. Na resolução de várias instâncias similares do problema, eles provavelmente só precisam ser executados uma vez.

`PZRefineCandsGeometry`

Este programa refina um conjunto de candidatos, utilizando as coordenadas cartesianas das amostras e não as curvaturas. O programa usa procedimentos de otimização não linear para encontrar a melhor rotação e o melhor deslocamento relativo dos dois segmentos. Ele é usado ocasionalmente para validar o casamento por curvatura codificada, ou para obter candidatos sabidamente corretos, para fins de teste.

`PZMatchImage`

Este programa produz uma imagem m em tons de cinza que mostra em forma gráfica todos os casamentos perfeitos de dois segmentos a, b na representação por curvatura, como a mostrada na figura 10.4.

PZRandomCands

Dado um conjunto de contornos observados, este programa extrai do mesmo um conjunto de candidatos aleatórios, constituído de pares de segmentos de mesmo comprimento escolhidos aleatóriamente. O programa opcionalmente lê uma lista de segmentos dentro dos quais os candidatos são escolhidos.

PZCurvHistogram

Este programa recebe um conjunto de gráficos de curvaturas de contornos, produzidos pelo `PZComputeCurvature`, e produz histogramas dos mesmos, como os da figura 9.4. O programa pode opcionalmente excluir dos histogramas um conjunto especificado de segmentos. Sua função principal é estimar o valor de $\hat{\kappa}_r$ usado na fórmula de quantização da curvatura (seção 9.3.1).

PZPlotMatchCost

Dado um conjunto de candidatos codificados por curvatura, este programa executa o algoritmo de programação dinâmica centrífuga para cada candidato, e determina para cada comprimento S o casamento de menor discrepância com esse comprimento. Esses valores de discrepância são plotados em função de S, produzindo um gráfico como o da figura 9.1. A principal função deste programa é determinar a discrepância crítica ξ a ser usada no casamento de segmentos e classificação dos candidatos.

PZSpectrum

Este programa calcula a transformada de Fourier de uma função periódica (por exemplo, a coordenada x ou y de um contorno) e grava o espectro de potência da mesma.

A.5 Programas auxiliares

PZConvertSyntheticCands

Este programa converte o formato do arquivo de um conjunto de candidatos obtidos manualmente (.txt) para o formato padrão de um conjunto de candidatos (.can)

PZMapChain

Este programa recebe um contorno (ou segmento) na forma de uma curva plana, e aplica à mesma uma rotação e/ou translação especificada.

A.6 Biblioteca `libm3pz`

Os programas utilizam várias bibliotecas de Modula 3, incluindo geometria projetiva (`libm3geo`), geração de arquivos Postscript (`libm3ps`), otimização (`libm3minu, libm3minn`), etc. Especificamente para este trabalho foi desenvolvida uma biblioteca especial, `libm3pz`, de operações com curvas, segmentos, candidatos, etc. A biblioteca `libm3pz` contém os seguintes módulos:

`PZGeo:`

Este módulo descreve vários procedimentos de manipulação geométrica. Ele inclui procedimentos para interpolar linearmente pontos (`LinearInterpolate`), interpolar cubicamente os pontos (`HermiteInterpolate`), estimar a velocidade a partir de três amostras sucessivas por vários métodos (`EstimateVelocityL, EstimateVelocityC, EstimateVelocityQ`), calcular a distância média quadrática entre dois segmentos de reta (`AvgSegDist`), calcular o comprimento da curva polinomial (`HermiteCurveLength`) e procedimentos `Rotation` e `Translation` para calcular as matrizes de rotação e translação que levam um segmento de reta em outro.

`PZProc:`

Este módulo apresenta uma miscelânea de funções e procedimentos auxiliares.

`PZLR3Chain:`

Este módulo define a representação de contornos amostrados onde as amostras são pontos do $\mathcal{R}^3$ (embora, no nosso trabalho, apenas as coordenadas x e y sejam utilizadas, pois nossas curvas são planas). Ele provê procedimentos para leitura e gravação de tais contornos.

`PZLRChain:`

Este módulo fornece um conjunto de procedimentos para manipulação de um vetor de números reais. Esta representação é usada tanto para representar os gráficos de curvaturas ao longo dos contornos, quanto os vetores que definem a correspondência entre as amostras de um contorno filtrado e as do contorno original (seção 4.7.5).

`PZSymbolChain:`

Este módulo define os principais procedimentos de manipulação de um conjunto de vetores de curvaturas codificadas. Ele também prove procedimentos para leitura e gravação de tais contornos.

`PZIntChain:`

Este módulo implementa um conjunto de procedimentos para manipulação de um vetor de inteiros.

`PZSmooth:`

Este módulo contém os procedimentos necessários para filtrar uma curva (`PZLR3Chain`) usando convolução.

`PZSegment:`

Este módulo define a representação de segmentos e os procedimentos para sua leitura (`Read`) e gravação (`Write`).

`PZCandidate:`

Este módulo define a representação de candidatos e os principais procedimentos para a sua manipulação. Em particular, inclui procedimentos para leitura e gravação (`Read`, `Write` e `Print`); procedimentos para detectar e juntar candidatos parcialmente sobrepostos ou repetidos (`MergeOverlaps`, `Overlap`) e para ordenar e eliminar candidatos pouco promissores (`Sort`, `Prune`).

`PZMatch:`

Este módulo provê os procedimentos o casamento genérico de contornos por meio de programação dinâmica tanto na forma padrão (`Match`), quanto na variante "centrífuga" (`OutMatch`). Estes procedimentos recebem como parâmetro uma função que determina o custo de um passo (i, j, i', j') de modo que eles podem ser usados para casamento de contornos quer eles estejam codificados por curvatura, quer na forma de coordenadas cartesianas.

`PZOptMatrix:`

Este módulo define o procedimento que cálcula o melhor alinhamento de dois segmentos a partir das coordenadas x e y das amostras. O procedimento retorna a matriz de translação e rotação que fornece o alinhamento ótimo.

`PZMismatch:`

Este módulo calcula a discrepância dados os valores da integral de $\|a - b\|$, a distância crítica, a distância de saturação e a penalidade por casamentos imperfeitos, conforme descrito no capítulo 7.

`PZPlot:`

Este módulo define procedimentos para desenhar contornos, segmentos, candidatos, e casamentos, gerando arquivos no formato de um arquivo Postscript (.ps) ou Encapsulated Postscript (.eps).

`PZFullPlot:`

Este módulo semelhante ao `PZPlot:`, mas também abre o arquivo para mudar escalas para ajustar as curvas.

`PZFourier:`

Este módulo define procedimentos para análise de Fourier de sequências reais (`PZLRChain`).

`PZImage`

Este módulo define procedimentos para representação e manipulação de imagens em tons de cinza, no formato ppm.

`PZMatrix`

Este módulo define a matriz de transformação geométrica, e procedimentos que criam matrizes de rotação e translação.

`PZShape`

Este módulo implementa o cálculo da função de forma de uma curva, como descrito no capítulo 5.

`PZSymbol`

Este módulo define a representação codificada das curvaturas e a a função ϕ (9.2) de quantização da curvatura.

A.7 Formatos dos arquivos

A.7.1 Contornos

Os contornos e arquivos associados (reparametrizações, curvaturas, etc) são armazenados em arquivos com formatos semelhantes. O formato é

`begin` $\langle$*tipo*$\rangle$ `(format of` $\langle$*data*$\rangle$`)`
`samples =` $\langle$*número*$\rangle$
`unit =` $\langle$*unidade*$\rangle$
$\langle$*amostra 0*$\rangle$
$\langle$*amostra 1*$\rangle$
$\vdots$
$\langle$*amostra* $n-1\rangle$
`end`

Para acelerar a leitura e gravação do arquivo gravamos os valores como inteiros que são multiplicados por um fator de correção real (`unit`) quando lidos do disco para a memória. A $\langle$*data*$\rangle$ é dada por ano-mês-dia, e é atualizada a cada modificação do formato do arquivo. O $\langle$*tipo*$\rangle$ pode ser

- `SymbolChain:` cada amostra é um caracter entre $z..a0A..Z$, representando um número entre -26 e -26, conforme descrito na seção 9.3. Em arquivos deste tipo, a linha `unit` é substituída por uma linha `sigma` $= \langle$ *numero* $\rangle$ que determina o desvio padrão $\hat{\kappa}$, usado na função de codificação ϕ.

- `LR3Chain:` cada amostra é um ponto do $\mathcal{R}^3$, dado por 3 números $x,\ y,\ z$. Este tipo de arquivo é usado para representar os contornos (atualmente planos, isto é $z = 0$).

- `LRChain:` cada amostra é um único número real. Este tipo de arquivo é usado para armazenar curvatura numérica em cada ponto de amostra, e o rótulo da mesma que define a correspondência entre diferentes escalas.

A.7.2 Segmentos

Este arquivo é usado para listar segmentos de contorno, por exemplo os segmentos de borda determinados pelo programa `PZStraightSegs`.

O formato é

`begin PZSegments (format of` $\langle data \rangle$`)`
`segments =` $\langle número \rangle$
$\langle segmento\ 0 \rangle$
$\langle segmento\ 1 \rangle$
$\vdots$
$\langle segmento\ (n-1) \rangle$
`end`

Onde cada $\langle segmento\ i \rangle$ tem as seguintes informações: o número do contorno , o número de amostras do mesmo, o índice da amostra inicial e o número de amostras do segmento.

A.7.3 Lista de candidatos

Este arquivo é usado para descrever um conjunto de candidatos numa dada escala.

`begin PZCandidate.List (format of` $\langle data \rangle$`)`
$\langle número\ de\ candidatos \rangle$
$\langle candidato\ 0 \rangle$
$\langle candidato\ 1 \rangle$
$\langle candidato\ 2 \rangle$
$\vdots$
$\langle candidato\ (n-1) \rangle$
`end`

Onde cada candidato tem as seguintes informações: os dois segmentos (como no arquivo de segmentos) cada qual seguido de um indicador do sentido de leitura, ("+" normal, "–" invertido); a discrepância, o comprimento total dos segmentos, o comprimento casado e opcionalmente um $\langle casamento \rangle$.

O significado da discrepância depende do programa que produz o arquivo. O casamento é dado pelo início do mesmo relativo a cada segmento, e uma seqüência codificada de passos (`/`, `|` ou `\`, dependento se apenas o primeiro segmento avança, se os dois segmentos avançam, ou se apenas o segundo segmento avança (vide capítulo 8)).

A.7.4 Lista de candidatos manuais

Para fins de teste, podemos gerar um conjunto de candidatos manualmente, bastando para isso produzir um arquivo no seguinte formato:

$n = \langle \textit{número de candidatos} \rangle$
$\langle \textit{descrição do candidato } 0 \rangle$
$\langle \textit{descrição do candidato } 1 \rangle$
$\langle \textit{descrição do candidato } 2 \rangle$
$\vdots$
$\langle \textit{descrição do candidato } (n-1) \rangle$

Cada $\langle \textit{descrição do candidato } i \rangle$ consiste das seguintes informações: o número do primeiro contorno, o início e o fim do segmento considerado; o número do segundo contorno, e o início e o fim do segmento considerado. O início e o fim são fornecidos na forma de frações decimais (entre 0 e 1) do comprimento total do contorno, a partir da amostra inicial. O programa `PZConvertCands` transforma este arquivo no formato padrão (seção A.7.3).

Bibliografia

[1] Adobe Systems Inc. *PostScript Language Reference Manual.* Addison-Wesley, Reading, Massachusetts, 1985.

[2] J. Babaud, A. Witkin, M. Baudin, and R. Duda. Uniquess of the Gaussian kernel for scale-space filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 8(1):26–33, 1986.

[3] Roger Bagnall. Advanced papyrological information system. File *papyrus/texts/-APISgrant.html* at *http://scriptorium.lib.duke.edu/*, 1994.

[4] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications.* The MacMillan Press LTD, London, 1976.

[5] F. Boussofiane and G. Bertrand. A new method for recognizing and locating objects by searching longest paths. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 26(12):445–448, 1993.

[6] Ronald N. Bracewell. *The Fourier Transform and its Applications.* McGraw-Hill, 1986.

[7] R. Bulirsch, R. Bartels, and W. Gautchi. *Introduction to Numerical Analysis.* Springer Verlag, Berlin, 1992.

[8] Grigore C. Burdea and Haim J. Wolfson. Solving jigsaw puzzles by a robot. *IEEE Transactions on Robotics and Automation*, 5(6):752–764, 1989.

[9] Roland T. Chin and Charles R. Dyer. Model-based recognition in robot vision. *ACM Computing Surveys*, 18(1):67–108, 86.

[10] K. Falconer. *Fractal Geometry: Mathematical Foundations and Applications.* John Wiley & Sons, 1990.

[11] K. T. Glowacki. Franchthi excavations: 17,000 years of Greek prehistory. *http://www.indiana.edu/∼archaeol/franchthi/franchthi/pot.html*, 1990.

[12] Sam P. Harbison. *Modula-3.* Prentice-Hall, 1992.

[13] Alan Kalvin, Edith Schonberg, Jacob T. Schwartz, and Micha Sharir. Two-dimensional, model-based, boundary matching using footprints. *The International Journal of Robotics Research*, 5(4):38–55, 1986.

[14] Alan D. Kalvin, Alfredo Remy, Orlando Ardito, Kim Morla, Eduardo Nolasco, Jorge Prado, Regulo Franco, Antonio Murga, and Guillermo Wiese. Using visualization in the archaeological excavations of a pre-Inca temple in Peru. Report RC 20518, IBM T. J. Watson Research Center, 1996.

[15] Joan Keller. The glass from Sepphoris: A preliminary report. *http://www.colby.edu/rel/Glass.html*, 1995.

[16] Bhagwandas P. Lathi. *Communications systems.* John Wiley & Sons, 1968.

[17] Raymond Legault and Ching Y. Suen. Optimal local weighted averaging methods in contour smoothing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(8):801–817, 1997.

[18] James W. McKee and J. K. Aggarwal. Computer recognition of partial views of curved objects. *IEEE Transactions on Computers*, c26(8):790–800, 1977.

[19] João Meidanis and João Carlos Setubal. *Introduction to Computational Molecular Biology.* PWS, 1997.

[20] Farzin Mokhtarian. Silhouette-based isolated object recognition through curvature scale space. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 17(5):539–544, 1995.

[21] Farzin Mokhtarian and Alan Mackworth. Scale-based description and recognition of planar curves and two-dimension shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 8(1):34–43, 1986.

[22] Farzin Mokhtarian and Alan K. Mackworth. A theory of multiscale, curvature-based shape representation for planar curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 14(8):789–805, 1992.

[23] Hochfeiler Multimedia. Gli archivi di Ebla. *http://www.sysin.it/ipertest/archi.htm*, 1997.

[24] Greg Nelson. *Systems Programming with Modula-3.* Prentice-Hall, 1991.

[25] Carl E. Pearson. *Handbook of Applied Mathematics — Selected Results and Methods.* Van Nostrand Reinhold - VNR - 2^a edição, 1983.

[26] Arthur R. Pope. Model based object recognition: A survey of recent research. Technical Report TR-94-04, Berkeley University, California, 1994.

[27] Anothai Rattarangsi and Roland T. Chin. Scale-based detection of corners of planar curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 14(4):430–449, 1992.

[28] Azriel Rosenfeld and Avinash C. Kac. *Digital Picture Processing.* Academic Press, 1982.

[29] Paul Rosin and Svetha Venkatesh. Extracting natural scales using the Fourier description. *Pattern Recogniton*, 26(9):1383–1393, 1993.

[30] Paul L. Rosin. Multiscale representation and matching of curves using codons. *CVGIP: Graphical Models and Image Processing.*, 55(4):286–310, 1993.

[31] B. Sandak, R. Nussinov, and Haim J. Wolfson. An automated computer vision and robotics based technique for 3-d flexible biomolecular docking and matching. *Computer Applications in the Biosciences*, 11(1):87–99, 1995.

[32] Guillermo Sapiro and Allen Tannenbaum. Area and lenght preserving geometric invariant scale-spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 17(1):67–72, 1995.

[33] Behzad Shahraray and David J. Anderson. Optimal estimation of contour properties by cross-valided regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 8(6):600–610, 1989.

[34] Göktürk Üçoluk and I. Hakki Toroslu. Reconstruction of 3-d surface object from its pieces. In 9_{th} *Canadian Conference on Computational Geometry*, volume 1, 1997.

[35] R. Vergnieux. Le fac-similé èlectronique ou les restitutions virtuelles. In X^e *CNRS Table Ronde de Informatique et Égyptologie*, Bordeaux, 1994.

[36] Andrew P. Witkin. Scale-space filtering. In *Proc. IJCAI Eighth Proc. International Joint Conference on Artificial Intelligence*, pages 1019–1021, 1983.

[37] H. Wolfson, E. Schonberg, A. Kalvin, and Y. Lamda. Solving jigsaw puzzles by computer vision. In *Annals of Operations Research*, volume 1986, pages 51–64, 1988.

[38] Haim J. Wolfson. Model-based object recognition by geometric hashing. In *Proc. First European Conference on Computer Vision*, volume 427, pages 526–536, 1990.

[39] Haim J. Wolfson. On curve matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 12(5):483–488, 1990.

[40] Daniel M. Wuescher and Kim L. Boyer. Robust contour decomposition using a constant curvature criterion. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*, 13(1):41–51, 1991.