Universidade Estadual de Campinas

Faculdade de Engenharia Elétrica e de Computação

Departamento de Semicondutores, Instrumentos e Fotônica

Um Método Para Granulometria de Imagens Topográficas Baseado na Teoria de Cálculo da Dimensão Fractal

Autor: Laerte Ferreira Morgado

Orientador: Prof. Dr. Vítor Baranauskas

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, como parte dos pré-requisitos para obtenção do Título de Mestre em Engenharia Elétrica.

Outubro de 1996

BIBLIOTECA CENTRAL
UNICAMP

ADE IBC
CHAMADA:
Unicamp
M82m
Ex.
TOMBO BC/ 31489
PROC. 281/97
C [ ] D [X]
PREÇO R$ 11,00
DATA 29/08/97
CPD

CM-00099889-1

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

M82m

Morgado, Laerte Ferreira
Um método para granulometria de imagens topográficas baseado na teoria de cálculo da dimensão fractal / Laerte Ferreira Morgado.--Campinas, SP: [s.n.], 1997.

Orientador: Vitor Baranauskas.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Fractais. 2. Processamento de imagens. 3. Programação orientada a objetos. 4. Hausdorff, Medidas de. 5. Teoria das medidas. 6. Mapas topográficos. I. Baranauskas, Vitor. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Resumo

Apresentamos um método baseado na Teoria dos Fractais que permite efetuar o cálculo do grau de irregularidade em cada ponto da superfície de uma imagem topográfica. O algoritmo proposto fornece valores que são função da escala de observação, de forma a ignorar irregularidades com tamanhos muito maiores ou muito menores que o valor da escala. Dessa forma, é possível implementar duas funcionalidades: calcular os graus de irregularidade para todos os *pixels* de uma imagem em uma determinada escala de observação e calcular os graus de irregularidade em diversas escalas para um determinado *pixel* da imagem. Com a primeira funcionalidade, podemos segmentar a imagem topográfica em regiões de maior ou menor irregularidade quando observada sob uma determinada escala. Com a segunda funcionalidade, podemos estudar a variação do grau de irregularidade de um ponto da imagem quando variamos a escala de observação. Mostramos que esse estudo permite identificar os tamanhos das irregularidades que ocorrem em superfícies topográficas como, por exemplo, os tamanhos médios dos grãos e as distâncias médias entre eles. Um ambiente gráfico foi desenvolvido com a concepção de Orientação a Objetos em C++ para estudo desse algoritmo e pode ser facilmente usado para análise de outros algoritmos similares.

Abstract

We describe a method based on the Theory of Fractals to calculate a measure of the degree of irregularity in each surface point of any topographic image. The proposed algorithm gives values that are dependent on the scale of observation so as to ignore irregularities which sizes are much greater or lower than the scale value. Therefore, it is possible to implement two features: calculation of the degrees of irregularity for all the pixels of an image in a given scale of observation and calculation of the degrees of irregularity in many scales of observation for a given image pixel. With the first feature we can segment the topographic image in regions of different degrees of irregularity in a given scale of observation. With the second feature we can study the variation of the degree of irregularity measured for an image pixel while we change the scale of observation. We show that the proposed method allows the identification of the sizes of irregularities that occur in topographic surfaces, such as the mean sizes of the grains.

Agradecimentos

Sem dúvida alguma, nenhum de meus projetos pessoais ou profissionais teria tido a menor possibilidade de sucesso sem o apoio e confiança que em mim depositaram meus pais, Jacinta da Silva Morgado e Lafaiete Ferreira Morgado. Ao meu pai devo inúmeros ensinamentos de vida os quais teve a coragem e humildade de aprender, coisa das mais raras e das mais difíceis de se conseguir.

Aos meus irmãos, Lenilson e Luciano, pelos momentos de alegria e aventura que tivemos, cujas lembranças fizeram-me seguir em frente, sem dúvidas. À minha irmã, Doralice, por sua infinita compreensão.

Muitos agradecimentos aos meus inúmeros familiares, em especial à Edilamar, por ter reconhecido muito cedo meu trabalho e aos meus tios Anastácio e Mariza pelo companheirismo e coragem.

Aos professores do CESO, em Brasília, pelo pioneirismo que tem ajudado gerações de estudantes da rede pública de ensino. Um agradecimento especial ao Prof. Clóvis, pelas oportunidades que me proporcionou.

Agredeço também ao Prof. Dr. Pedro Jussieu de Rezende, pelas ótimas oportunidades de trabalhos durante meus tempos de Iniciação Científica e ao Prof. Dr. Célio Guimarães por inúmeras oportunidades e confiança.

Um agradecimento especial ao meu orientador, Prof. Dr. Vítor Baranauskas e a sua esposa, Prof.ª Dr.ª Maria Cecília Baranauskas, por terem me dado apoio em situações muito difíceis.

Ao meu colega Marcelo Tosin, pela amizade e pelo excelente intercâmbio de idéias.

À Unicamp e à Faculdade de Engenharia Elétrica e Computação pelas instalações e equipamentos.

Ao CNPq pelo suporte finaceiro durante toda esta jornada.

Aos inúmeros amigos com os quais sempre pude contar e que tornaram-se imprescindíveis para o sucesso de meu trabalho.

A própria percepção do que você é, tal como é, no momento de ação num relacionamento, liberta daquilo "que é". Apenas em liberdade pode haver descoberta. Uma mente condicionada não pode descobrir a verdade. Liberdade não é abstração; ela se torna realidade com a virtude. Pois a exata natureza da virtude é livrar-nos das causas da confusão. Afinal de contas, a não-virtude significa desordem, conflito. Mas a virtude é liberdade, é a clareza da percepção que o entendimento produz. Você não pode se tornar virtuoso. O tornar-se é a ilusão da ambição, da cobiça. Virtude é a percepção imediata daquilo "que é". Dessa forma, o autoconhecimento é o princípio da sabedoria, e a sabedoria irá resolver os nossos problemas e, por conseguinte, os problemas do mundo.

J. Krishnamurti - Colombo, Dezembro de 1949

Conteúdo

RESUMO i

ABSTRACT ii

AGRADECIMENTOS iii

CONTEÚDO v

LISTA DE FIGURAS vi

1 Introdução 1

2 Motivação e Objetivo 4

3 Fundamentos Teóricos 9

4 O Método 16

5 O Ambiente Fraktal 33

5.1 A Implementação de Novos Algoritmos 40

5.2 A Implementação de Novos Objetos Gráficos 46

5.3 Usos e Limitações do Sistema Fraktal 47

5.4 Como Usar o Sistema Fraktal 48

5.5 A Opção Normalize/Detail Emphasis 53

6 Estudo de Imagens 59

6.1 O Uso do Espectro de Irregularidades 62

7 Conclusão 77

BIBLIOGRAFIA 78

Lista de Figuras

1.1 Construção da Curva de Koch. 2

2.1 Placas de um capacitor com $E_1 \approx E_2$. As irregularidades afetam significativamente o meio itermediário. 5

2.2 Placas de um capacitor com $E_1 \gg E_2$. As placas comportam-se como uma superfície plana. 5

2.3 A dependência da irregularidade com a escala de observação. 7

3.1 Uma δ-cobertura de um conjunto $F \subset \mathbf{R}^2$ 10

3.2 Gráfico de $\mathcal{H}^s(F)$ versus s. A dimensão de Hausdorff é o valor de s no qual ocorre o salto de ∞ para 0. 12

3.3 Cobertura de um retângulo por cubos de lado δ. 13

3.4 Duas distribuições homogêneas de massa. 14

4.1 Perfil de uma imagem topográfica e sua representação em computador. 17

4.2 A altura da imagem oscila imprevisivelmente entre $T - A/2$ e $T + A/2$ dentro de um quadrado de lado δ. 18

4.3 Segmentação da imagem por uma grade de quadrados. Dentro de cada quadrado, a altura da imagem localiza-se entre um valor mínimo e um máximo. 19

4.4 Segmentação da imagem por uma grade de quadrados. Cada quadrado possui lado δ e engloba M colunas e L linhas. 21

4.5 O conceito de massa em escala δ. 22

4.6 A massa em escala δ como medida de irregularidade (I). 22

4.7 A massa em escala δ como medida de irregularidade (II). 23

4.8 O cálculo de $\mu_P^\delta(d)$ (I). 24

4.9 O cálculo de $\mu_P^\delta(d)$ (II). 24

4.10 O cálculo de $\mu_P^\delta(d)$ (III). 25

4.11 Como aproveitar a adjacência de quadrados de *pixels* no cálculo da variância de seus valores (I). 29

4.12 Como aproveitar a adjacência de quadrados de *pixels* no cálculo da variância de seus valores (II). 30

5.1 Relacionamento entre as classes *FKTInterface*, *FKTIAlg* e *FKTIObj*. 36

5.2 Hierarquia de classes parcial do sistema Fraktal. 38

5.3 A janela aberta pela classe *FKTXVParms*. 44

5.4 Janela principal do sistema Fraktal. 48

5.5 A entrada do nome de um arquivo de imagem. 49

5.6 A janela de visualização de uma imagem. 50

5.7 A janela de operações para imagens. 51

5.8 A janela de execução do algoritmo que calcula o mapa de irregularidades de uma imagem. 52

5.9 Um mapa de irregularidades. 54

5.10 A janela de operações de um mapa de irregularidades. 55

5.11 A janela de execução do algoritmo que calcula o espectro de irregularidades de um ponto da imagem. 55

5.12 O gráfico que mostra os valores de irregularidades de um *pixel* para várias escalas. Os valores do eixo das abscissas estão em nanômetros. O eixo das ordenadas é adimensional. 56

5.13 A janela de operação sobre o espectro de irregularidades. 57

5.14 O processo de quantização do valores dos *pixels*. 58

6.1 O decréscimo da irregularidade para $\delta > \mathcal{A}$. 60

6.2 O espectro de irregularidades para uma imagem com $\mathcal{A} = 40$. O valores do eixo das abscissas representam quantidades de *pixels*. O eixo das ordenadas é adimensional. . 61

6.3 Uma imagem com pirâmides de altura 5, base de lado 30 e com espaçamentos horizontal e vertical iguais a 40. 63

6.4 Espectro de irregularidades com $n = 2$, para um *pixel* no topo de uma pirâmide da figura 6.3. Os valores do eixo das abscissas representam quantidade de *pixels*. O eixo das ordenadas é adimensional. 65

6.5 Espectro de irregularidades com $n = 2$, para um *pixel* no ponto médio entre quatro pirâmides da figura 6.3. Os valores do eixo das abscissas representam quantidade de *pixels*. O eixo das ordenadas é adimensional. 66

6.6 Imagem resultante da adição de uma senóide com um mapa de pirâmides. 70

6.7 O espectro de irregularidades para a imagem da figura 6.6. Os valores do eixo das abscissas representam quantidades de *pixels*. O eixo das ordenadas á adimensional. . 71

6.8 Imagem com duas regiões com diferentes níveis de oscilação vertical. 73

6.9 Mapa de irregularidades para a imagem da figura 6.8. 74

6.10 Imagem de um filme de diamante obtida com o sistema Fraktal. 75

6.11 O espectro de irregularidades para um *pixel* da imagem 6.10. Os valores do eixo das abscissas estão em nanômetros. O eixo das ordenadas é adimensional. 76

Capítulo 1

Introdução

Exitem vários aspectos instigantes envolvidos nos conceitos de definição da Geometria Fractal. Um dos mais importantes é, sem dúvida, a existência de objetos geométricos com dimensão fracionária, muito ao contrário do que ocorre com a Geometria Euclidiana que define objetos com dimensão topológica inteira. Nesse sentido, é bastante conhecida a chamada curva de Koch, obtida da seguinte maneira: pegue um segmento de reta L_0 e o divida em três sub-segmentos de mesmo tamanho. Remova o segmento do meio e construa, como mostra a figura 1.1, dois outros segmentos com o mesmo tamanho do segmento removido. Nesse instante, temos a curva L_1 com quatro segmentos de reta. Repita recursivamente o mesmo procedimento acima para cada um desses segmentos. A curva de Koch é o limite da seqüência de curvas L_i, quando i tende para infinito, limite este que iremos representar por L_∞. Seu aspecto lembra o de um floco de neve e pode ser provado que sua dimensão fractal (mais precisamente, sua dimensão de Hausdorff) é $\log 4/\log 3 \approx 1.26$. Analisando um pouco essa construção, podemos notar que, se o comprimento da curva L_0 é 1, o comprimento da curva L_i é $4^i/3^i$. Ou seja, a curva de Koch (L_∞) tem comprimento infinito, embora isso possa não parecer óbvio de imediato. Intuitivamente, podemos pensar que a curva fica tão densa, de forma que seus pontos conseguem preencher um parcela da vizinhança bidimensional do espaço no qual está inserida. Mas não o preenche totalmente, de forma que sua dimensão é um número real maior que 1 e menor que 2.

Além da definição da dimensão fractal, a própria definição dos objetos fractais envolve características inicialmente pouco intuitivas. Uma delas é a presença de detalhes em todas as escalas de observação. Ou seja, por menor que seja a vizinhança de um ponto do objeto, sempre encontraremos estruturas do objeto de comprimento ainda menor. Isso pode ser observado facilmente na curva de Koch. Se dela extrairmos um porção diminuta dentro de uma bola de raio r centrada em um de seus pontos, veremos que existirá um passo de construção n grande o suficiente de forma que os segmentos de reta da parte extraída serão menores do que r. Outra característica comum dos objetos fractais é a auto-similaridade em todas as escalas de observação. Observe que na seqüência de construção da curva de Koch, a curva L_i é composta de quatro partes similares à curva L_{i-1} com comprimento reduzido a um terço. Ou seja, cada uma dessas partes dá origem

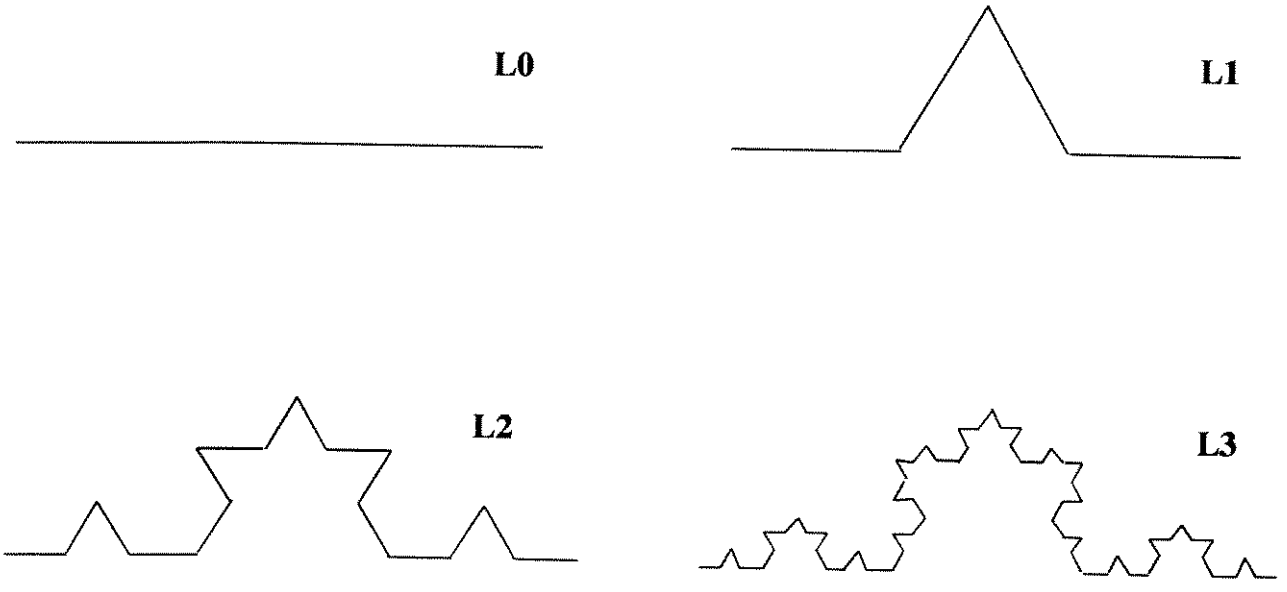

Figura 1.1: Construção da Curva de Koch.

a uma seqüência de curvas que, no limite, dão origem a curvas que são similares à curva Koch como um todo, apenas em escala menor. Em resumo, a curva de Koch pode ser olhada como sendo composta de 4, 16, 64, 256, etc. curvas de Koch em escalas cada vez menores.

Várias outras características são usadas na identificação dos objetos fractais. Mandelbrot [Man82] define um fractal como sendo um conjunto cuja dimensão de Hausdorff é estritamente maior do que sua dimensão topológica. Além disso, os objetos definidos como fractais são, em geral, muito irregulares para serem descritos em linguagem geométrica tradicional.

Talvez o mais importante é compreender que os fractais não possuem uma definição matemática precisa, conforme observado em [Fal90]. São caracterizados a partir de uma série de propriedades, as quais permitem distinguir claramente os objetos não-fractais, mas que podem ser violadas em muitos casos específicos. Assim, de acordo com [Fal90], os fractais:

- Possuem detalhes em escalas arbitrariamente pequenas;
- São demasiadamente irregulares para serem descritos em linguagem geométrica tradicional;
- Possuem alguma forma se auto-similaridade, muitas vezes aproximada ou estatística;
- Geralmente possuem dimensão fractal maior que sua dimensão topológica;
- Em muitos casos, podem ser definidos de uma forma muito simples, talvez recursivamente.

A grande popularidade da Teoria dos Fractais como aplicação em outras áreas do conhecimento deve-se, talvez, a dois fatores. Um deles é a relativa facilidade com que fenômenos que possuem comportamento fractal são encontrados na Natureza. Assim, por exemplo, bordas de nuvens, superfícies topográficas, costas de continentes, turbulência de fluidos, podem ser descritos e modelados com conceitos fractais. Outro fator é a interpretação intuitiva bastante útil que os conceitos fractais possuem quando lidamos com fenômenos complexos. Como exemplo, na área da Física conhecida como Litografia, encontramos freqüentemente superfícies topográficas bastante irregulares que possuem comportamento fractal aproximado em determinadas faixas de escala. A dimensão fractal dessas superfícies indica, intuitivamente, o grau de rugosidade. Quanto maior a dimensão fractal, maior o grau de rugosidade.

Talvez justamente por esses fatores, diversos trabalhos têm sido publicados sobre o uso da dimensão fractal nas mais diversas áreas do conhecimento. Uma rápida observação da bibliografia desta Tese irá mostrar a quantidade e diversidade destes trabalhos. Vários deles estão no campo da Física e tentam estabelecer correlação entre a dimensão fractal de uma imagem e certas propriedades da superfície da qual foi obtida. Outros são da Química, da Metalurgia, da Biologia ou da Geologia. Na maioria deles, a estimativa da dimensão fractal assume papel central como medição de propriedades relacionadas com a rugosidade, textura, irregularidade ou grau de complexidade do sistema em estudo.

Neste contexto, trabalhos que procurem estudar maneiras novas de se estimar rapidamente e precisamente a dimensão fractal ou que procurem explorar novos métodos de se usar a Teoria de Fractais na solução de problemas encontrados na prática de diversas áreas do conhecimento são sempre bem vindos. Nosso trabalho classifica-se muito bem neste último time ao propor um método que, baseado na teoria de cálculo da dimensão fractal, pode ser utilizado para:

- Estimar os tamanhos das irregularidades que desenvolvem-se na superfície de imagens topográficas;
- Investigar as escalas em que as irregularidades desenvolvem-se com maior ou menor intensidade;
- Segmentar as imagens topográficas em áreas de maior ou menor rugosidade, numa determinada escala de observação.

Esperamos que este método, juntamente com a interface gráfica que implementamos, seja útil no estudo das superfícies topográficas encontradas comumente em áreas como a Física e, em especial, na Microeletrônica.

Capítulo 2

Motivação e Objetivo

De acordo com a Teoria dos Fractais, a dimensão fractal de um conjunto de pontos no espaço $\mathbf{R}^n$ é um número real entre 0 e n o qual, intuitivamente, indica o grau de irregularidade ou complexidade do objeto. No caso específico de imagens topográficas, esse número, conforme observado em [Pfe84, CWLK89] permite medir a textura ou rugosidade da superfície.

Entretanto, nos métodos comumente encontrados para estimativa da dimensão fractal de uma imagem topográfica, inexiste a consideração a respeito da escala, ou ordem de grandeza dos comprimentos, na observação da superfície de origem; limitando-se, os algoritmos, a devolver o tal número real que, teoricamente, deveria ser a dimensão fractal real.

Consideremos, o problema a seguir, ilustrado pelas figuras 2.1 e 2.2. Na figura 2.1, vemos o desenho de duas placas de um capacitor bastante próximas e observadas em uma escala muito pequena. Nesta escala de observação, as irregularidades na superfície da placas são muito visíveis e os efeitos físicos dessas irregularidades são bastante fortes fazendo com que, por exemplo, o campo elétrico seja bastante irregular no meio físico intermediário. Na figura 2.2 as placas encontram-se bastante afastadas e o meio intermediário já não sofre os efeitos das irregularidades de suas superfícies como antes. Ou seja, as placas comportam-se muito mais como um plano do que como uma superfície altamente irregular. Se quisermos usar a dimensão fractal ou algum valor dela derivado como medida de irregularidade das superfícies, no primeiro caso deveríamos ter um valor bem maior que no segundo, apesar de as superfícies serem as mesmas, mudando apenas a escala de observação.

O mesmo problema é encontrado em vários outros lugares. Quando olhamos a costa de um continente em uma foto tirada do espaço orbital notamos um certa irregularidade, que não é muito grande. À medida que nos aproximamos mais e mais da superfície terrestre notamos um aumento gradativo da irregularidade à medida que mais detalhes tornam-se visíveis. Entretanto, a partir de uma certa aproximação, os detalhes que antes contribuíam para uma maior irregularidade da costa, tornam-se tão grandes que já não são tão influentes. Quando nos aproximamos mais ainda, vários outros detalhes que estavam invisíveis quanto o afastamento era grande, começam a

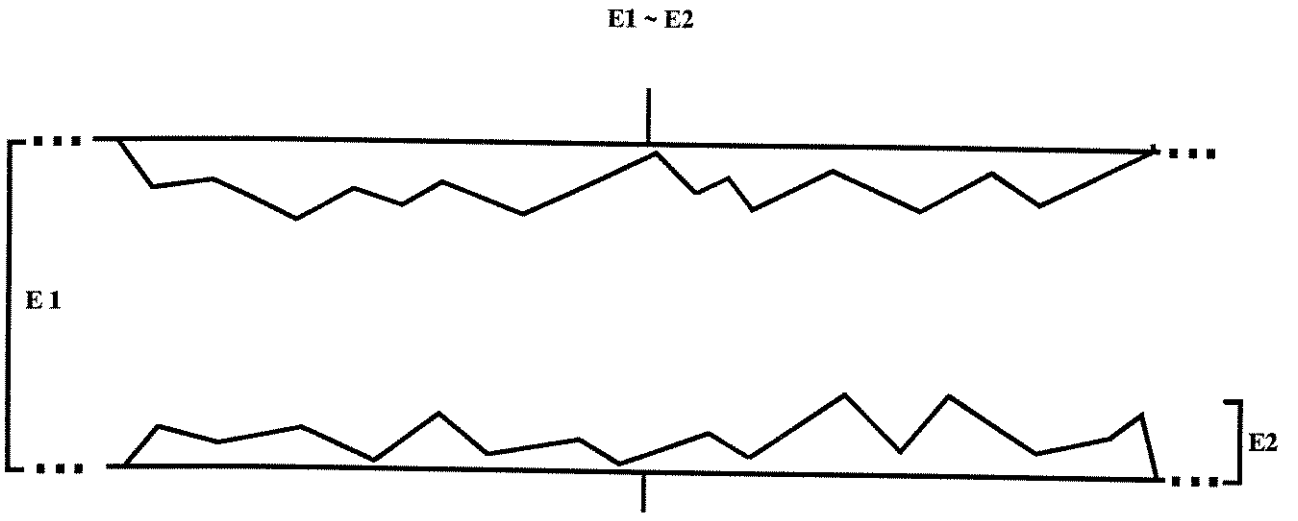

Figura 2.1: Placas de um capacitor com $E_1 \approx E_2$. As irregularidades afetam significativamente o meio itermediário.

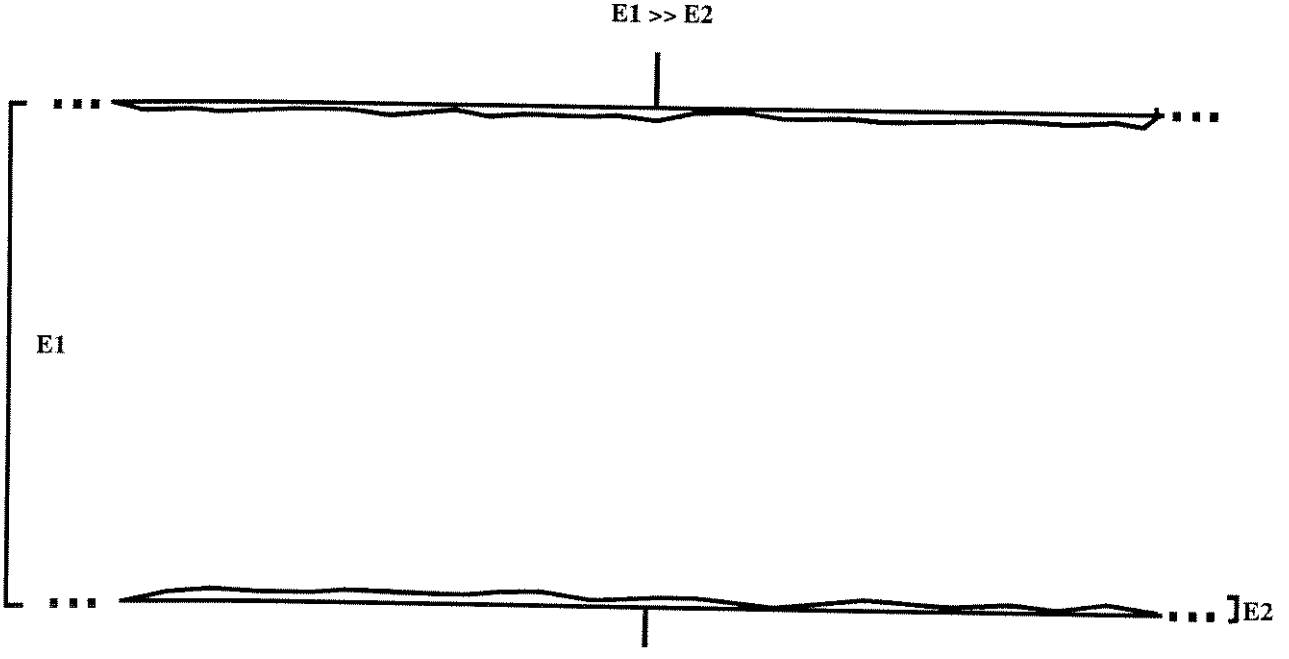

Figura 2.2: Placas de um capacitor com $E_1 \gg E_2$. As placas comportam-se como uma superfície plana.

aparecer e contribuem, por sua vez, para o grau de irregularidade na nova proximidade. Assim, à medida que nos aproximamos da superfície da Terra, o grau de irregularidade observado das costas dos continentes varia.

Ou seja, pesar de matematicamente a dimensão fractal independer da escala de observação, muitos objetos geométricos são melhor representados como possuindo dimensões fractais diferentes em escalas diferentes. Essa abordagem tem sido usada em diversos trabalhos, entre eles [Pfe84].

Mas, o que significa observar um objeto geométrico sob uma determinada escala? Significa, de alguma forma, ignorar os detalhes de comprimento razoavelmente maior ou menor que a escala considerada. Assim, nas medições de irregularidade, somente são levadas em conta aquelas com a mesma ordem de grandeza da escala em questão. E uma maneira fácil de desprezar as irregularidades de valor muito maior que uma determinada escala é sempre analisar os pontos do objeto em uma vizinhança de diâmetro aproximadamente igual à escala, desprezando, no cálculo da medida de irregularidade, os pontos fora dessa vizinhança. Por outro lado, para desprezar as irregularidades muito pequenas, a função de medição de irregularidades deve ser tal que elas influenciam muito pouco em seu valor na escala considerada.

Observemos a figura 2.3, na qual vemos o perfil de uma imagem com duas regiões bem distintas em termos dos tamanhos das irregularidades que possuem. Na parte da esquerda, as irregularidades possuem um tamanho aproximado de δ, equanto que na parte da direita o tamanho das irregularidades é aproximadamente Δ. Se compararmos as vizinhanças V_1 e V_2 de diâmetro δ, respectivamente ao redor dos pontos P_1 e P_2, notaremos que, nesta escala, o grau de irregularidade no primeiro caso é muito maior do que no segundo. Por outro lado, se compararmos os graus de irregularidades da superfície da imagem dentro das vizinhanças V_3 e V_4 de diâmetro Δ notaremos que é maior na vizinhança P_4. Ou seja, nas situações em que os tamanhos das irregularidades da superfície são muito maiores ou muito menores que a escala de observação, o grau de irregularidade medido deve ser baixo. Quando esses dois valores são da mesma ordem de grandeza, o grau de irregularidade deve apresentar valores elevados.

Considerando toda essa discussão, podemos concluir que seria bastante útil um método que pudesse estimar a dimensão fractal de uma imagem em diferentes escalas de observação, ignorando irregularidades de tamanho muito maior ou muito menor que a escala em consideração. Com tal método, poderíamos implementar duas funções bastante úteis:

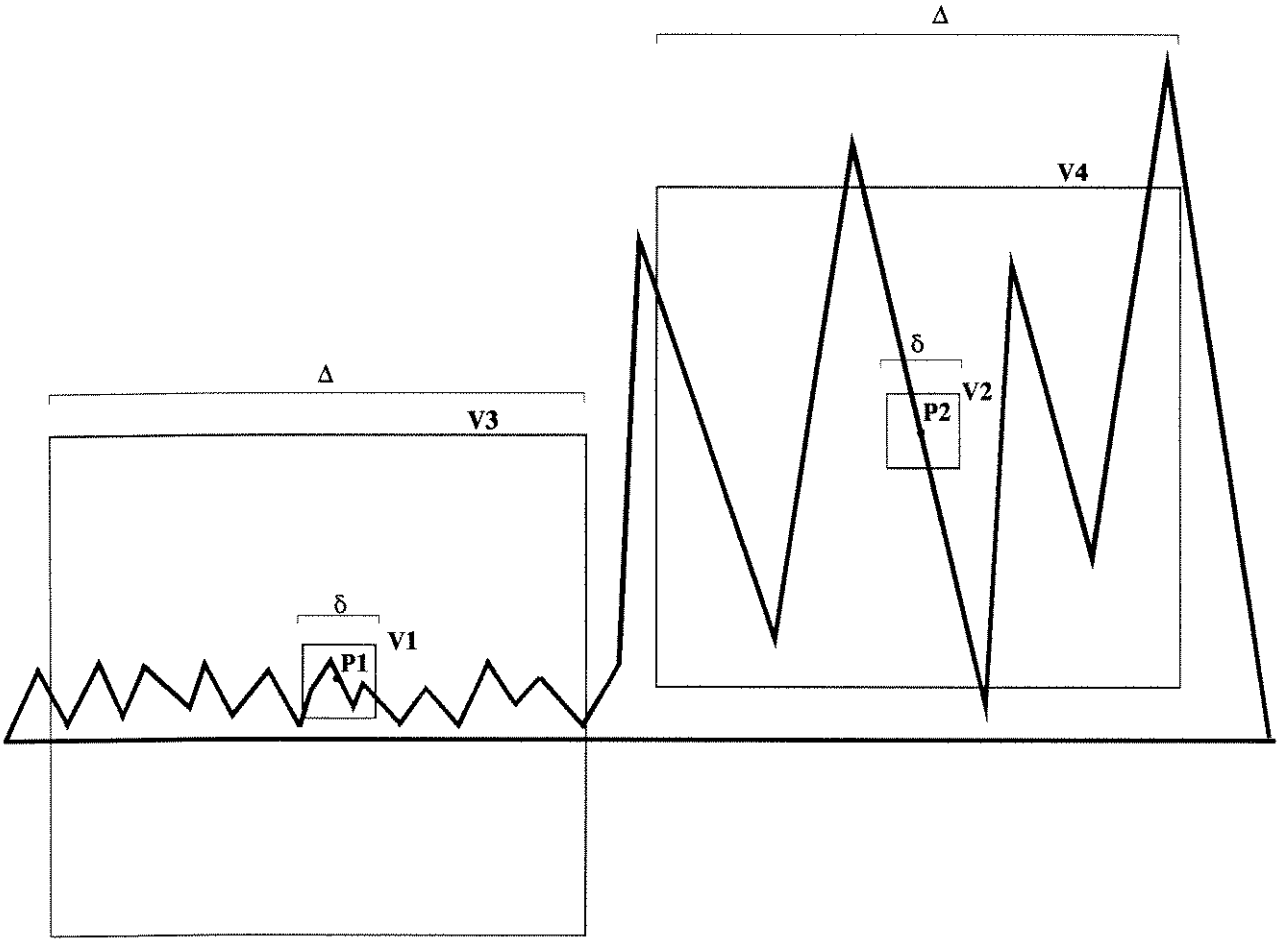

Figura 2.3: A dependência da irregularidade com a escala de observação.

1. Dado um ponto de uma superfície topográfica, calcular o conjunto das dimensões fractais para várias escalas de observação;
2. Dada uma escala de observação, calcular um mapa bi-dimensional contendo os valores da dimensão fractal para todos os pontos da imagem.

Com a primeira função, poderíamos construir um gráfico das dimensões fractais versus escalas de observação. Através da análise dos picos desse gráfico, teríamos a indicação das escalas em que a superfície mostra-se mais irregular e, de acordo com o raciocínio dos parágrafos anteriores, saberíamos os tamanhos de suas irregularidades. Com a segunda função, seríamos capazes de segmentar a imagem em regiões de maior e menor irregularidade numa determinada escala. Além disso, o uso combinado desta funcionalidade com a primeira permitiria, por meio do cálculo do mapa de dimensões para as escalas com picos de dimensão, destacar do resto da imagem as irregularidades de uma determinada escala.

Vários conceitos estão implícitos nas observações acima. Em primeiro lugar, temos os conceitos de irregularidade e tamanho de irregularidades de uma superfície topográfica, para os quais não pretendemos dar uma definição rigorosa. Intuitivamente, entendemos por irregularidade qualquer variação da altura da superfície ao longo de seu plano base. Assim, temos como irregularidades os picos e os grãos da superfície, as depressões, etc. Os tamanhos das irregularidades são as alturas e larguras dos grãos, a profundidade dos vales, as distâncias entre os grãos, etc.

Estamos procurando um método que forneça uma medida das irregularidades de uma determinada escala que ocorrem em uma superfície topográfica. Esse método será tanto melhor quanto mais eficientemente ignorar irregularidades de tamanhos muito menores e muito maiores que a escala em consideração. Esse método deve ainda fornecer uma medida local da irregularidade na vizinhança de cada ponto da imagem. Não deve, portanto, ser um método global que forneça uma medida única para a imagem como um todo. Com um método como esse podemos implementar as funcionalidades discutidas acima, as quais permitem a realização de um estudo de granulometria da superfície, bem como podem ser usadas para segmentação baseada no grau de irregularidade variável ao longo do plano base da imagem. Por fim, conforme observado por [HLD94], os algoritmos conhecidos para estimativa da dimensão fractal produzem resultados que diferem muito entre si, fazendo com que uma estimativa precisa seja muito difícil de se conseguir. Dessa forma, não estaremos interessados em medir a dimensão fractal propriamente dita, mas em ter um algoritmo que produza números que cresçam com o grau de irregularidade em um dado *pixel* numa dada escala. O método proposto produz números fora da faixa de 2 a 3 para superfícies topográficas no espaço $\mathbf{R}^3$ e, dessa forma, procuraremos chamá-los de *irregularidades calculadas* e não de *dimensões.*

Capítulo 3

Fundamentos Teóricos

Nas seções anteriores estivemos falando intuitivamente sobre vários conceitos e sobre a motivação de nosso trabalho. Nesta seção forneceremos definições rigorosas extraídas da Teoria de Medidas e da Teoria dos Fractais, as quais serão imprescindíveis para a descrição do método de estimativa de irregularidade na próxima seção. A maioria das definições e teoremas aqui apresentados foram extraídos de [Fal90]. Em primeiro lugar vamos definir vários conceitos relativos à chamada medida de Hausdorff e à dimensão de Hausdorff de um conjunto. Em seguida, apresentaremos um teorema que relaciona a dimensão de Hausdorff com uma distribuição de massa do espaço $\mathbf{R}^n$, o qual é central para o algoritmo a ser apresentado.

Dado um sub-conjunto U do espaço Euclidiano n-dimensional, $\mathbf{R}^n$, o diâmetro de U é definido como $\mid U \mid = \sup \{\mid x - y \mid : x, y \in U\}$. Ou seja, é a maior distância entre quaisquer dois pontos de U. Se $\{U_i\}$ é uma coleção enumerável de conjuntos de diâmetro de valor máximo δ que cobrem F, ou seja, $F \subset \bigcup_{i=1}^{\infty} U_i$ com $0 < \mid U_i \mid \leq \delta$ para cada i, então $\{U_i\}$ é uma δ-cobertura de F.

Suponha que F é um sub-conjunto de $\mathbf{R}^n$ e s é um número não negativo. Para qualquer $\delta > 0$, definimos:

$$\mathcal{H}^s_\delta(F) = inf\{\sum_{i=1}^{\infty} \mid U_i \mid^s : \{U_i\} \text{ é uma } \delta - \text{cobertura de } F\}. \tag{3.1}$$

Nesta equação olhamos todas as coberturas de F por conjuntos de diâmetro máximo δ e procuramos minimizar a somatória das s-ésimas potências dos diâmetros. Quando δ diminui, a classe de coberturas possíveis de F em 3.1 é reduzida. Assim, o ínfimo $\mathcal{H}^s_\delta(F)$ cresce e aproxima-se de um limite quando $\delta \to 0$. Escrevemos, então :

$$\mathcal{H}^s(F) = \lim_{\delta \to 0} \mathcal{H}^s_\delta(F). \tag{3.2}$$

Este limite existe para qualquer sub-conjunto F de $\mathbf{R}^n$, embora o valor limite seja muito freqüentemente 0 ou ∞. Chamamos $\mathcal{H}^s(F)$ de *medida de Hausdorff s-dimensional de* F. Para efeito

Figura 3.1: Uma δ-cobertura de um conjunto $F \subset \mathbf{R}^2$

de ilustração, a figura 3.1 mostra uma possível δ-cobertura de um conjunto F do espaço $\mathbf{R}^2$.

Com um certo esforço, pode-se mostrar que $\mathcal{H}^s(F)$ é uma verdadeira medida, ou seja :

- $\mathcal{H}^s(\emptyset) = 0$;
- $\mathcal{H}^s(A) \leq \mathcal{H}^s(B)$ se $A \subset B$;
- Se $A_1, A_2, \ldots$ é uma seqüência enumerável de conjuntos, então:

$$\mathcal{H}^s(\bigcup_{i=1}^{\infty} A_i) \leq \sum_{i=1}^{\infty} \mathcal{H}^s(A_i). \tag{3.3}$$

A medida de Hausdorff generaliza as idéias de comprimento, área e volume. Pode se demonstrado que, para sub-conjuntos do espaço $\mathbf{R}^n$ conhecidos como conjuntos de Borel (os quais são a grande maioria dos conjuntos de interesse), a medida de Hausdorff é proporcional à medida de Lebesgue, ou seja, a medida de volume usual. Mais precisamente, se F é um sub-conjunto de Borel do espaço $\mathbf{R}^n$, então

$$\mathcal{H}^s(F) = c_n vol^n(F). \tag{3.4}$$

A constante $c_n = \pi^{n/2}/2^n(n/2)!$ é o volume de uma bola n-dimensional de diâmetro 1. Assim, para sub-conjuntos *razoáveis* de $\mathbf{R}^n$, $\mathcal{H}^0(F)$ é o número de pontos em F, $\mathcal{H}^1(F)$ é o comprimento de uma curva suave F, $\mathcal{H}^2(F) = \pi/4 \times \acute{a}rea(F)$ de uma superfície suave, etc. Resumindo, $\mathcal{H}^m(F) = c_m \times vol^m(F)$ se F é uma sub-variedade m-dimensional de $\mathbf{R}^n$.

Voltando à equação 3.1, é claro que para qualquer conjunto F e para $\delta < 1$, $\mathcal{H}^s_\delta(F)$ é não-crescente com s. Então, $\mathcal{H}^s(F)$ em 3.2 é também não-crescente. De fato, se $t > s$ e $\{U_i\}$ é uma δ-cobertura de F, temos:

$$\sum_i | U_i |^t \leq \delta^{t-s} \sum_i | U_i |^s. \tag{3.5}$$

Tomando os ínfimos, temos $\mathcal{H}^t_\delta(F) \leq \delta^{t-s}\mathcal{H}^s_\delta(F)$. Fazendo $\delta \to 0$, vemos que se $\mathcal{H}^s(F) < \infty$, então $\mathcal{H}^t(F) = 0$ para $t > s$. O gráfico de $\mathcal{H}^s(F)$ versus s (veja figura 3.2) possui, dessa forma, um valor crítico de s no qual o valor de $\mathcal{H}^s(F)$ salta de ∞ para 0. Este valor crítico de s é chamado de *dimensão Hausdorff* de F e é representado por $dim_H F$. Formalmente, $dim_H F = inf\{s : \mathcal{H}^s(F) = 0\} = sup\{s : \mathcal{H}^s(F) = \infty\}$. Mais ainda, $\mathcal{H}^s(F) = \infty$ para $s < dim_H F$ e $\mathcal{H}^s(F) = 0$ para $s > dim_H F$. Se $s = dim_H F$, então $\mathcal{H}^s(F)$ pode ser 0, ∞ ou pode ser tal que $0 < \mathcal{H}^s(F) < \infty$. Conjuntos de Borel que satisfaçam a este último caso são chamados de s-conjuntos e são os mais convenientes de lidar.

Para uma visão intuitiva dessa discussão, considere a figura 3.3 na qual mostramos um retângulo R de lado L no espaço $\mathbf{R}^3$ e esquematizamos uma de suas δ-coberturas por cubos de lado $\delta = L/N$, onde N é um número inteiro maior que 0. Vemos que a somatória das s-ésimas potências dos diâmetros dos cubos (de acordo com a fórmula 3.1) é $M^s_\delta(R) = \sum_{i=0}^{N^2}(3^{s/2} \times \delta^s)$. Para $s = 1$, temos $M^s_\delta(R) = \sqrt{3} \times N^2 \times L/N$. Quando $\delta \to 0$, $N \to \infty$ e $M_1(R) \to \infty$. Da mesma forma, $M^3_\delta(R) = 3^{1.5} \times N^2 \times L^3/N^3$. Quando $\delta \to 0$, $M^3_\delta(R) \to 0$. Por outro lado, $M^2_\delta(R) \to 3 \times L^2$. Ou seja, embora não tenhamos tomado o ínfimo na fórmula 3.1 para todas as δ-coberturas de R, vemos que 2 é o valor de s para o qual M^s_δ pula de ∞ para 0. Assim, temos uma indicação de que a dimensão Hausdorff de R é 2, como seria de se esperar. Se usarmos a equação 3.4 e considerarmos as noções intuitivas de comprimento, área e volume, vemos que $\mathcal{H}^1(R) = comprimento(R) = \infty$, $0 < \mathcal{H}^2(R) = \pi/4 \times área(R) < \infty$ e $\mathcal{H}^3(F) = 4/3\pi \times vol(R) = 0$. Assim, $dim_H R$ é igual a 2, com $\mathcal{H}^s(R) = \infty$ se $s < 2$ e $\mathcal{H}^s(R) = 0$ se $s > 2$.

A dimensão de Hausdorff é muito elegante do ponto de vista matemático, mas torna-se praticamente inutilizável do ponto de vista computacional, justamente por lidar com conceitos como ínfimo e por trabalhar com conjuntos infinitos. Assim, é inviável calcular computacionalmente a dimensão Haurdorff de um conjunto a partir de sua definição original, dada pelas fórmulas 3.1 e 3.2. Entretanto, existem alguns teoremas que permitem estimar na prática a dimensão Haurdorff. Dentre eles, usaremos um que faz uso do conceito de distribuição de massa em $\mathbf{R}^n$. Por definição, uma distribuição de massa μ é uma medida em um sub-conjunto limitado de $\mathbf{R}^n$ de forma que $0 < \mu(\mathbf{R}^n) < \infty$. Ou seja,

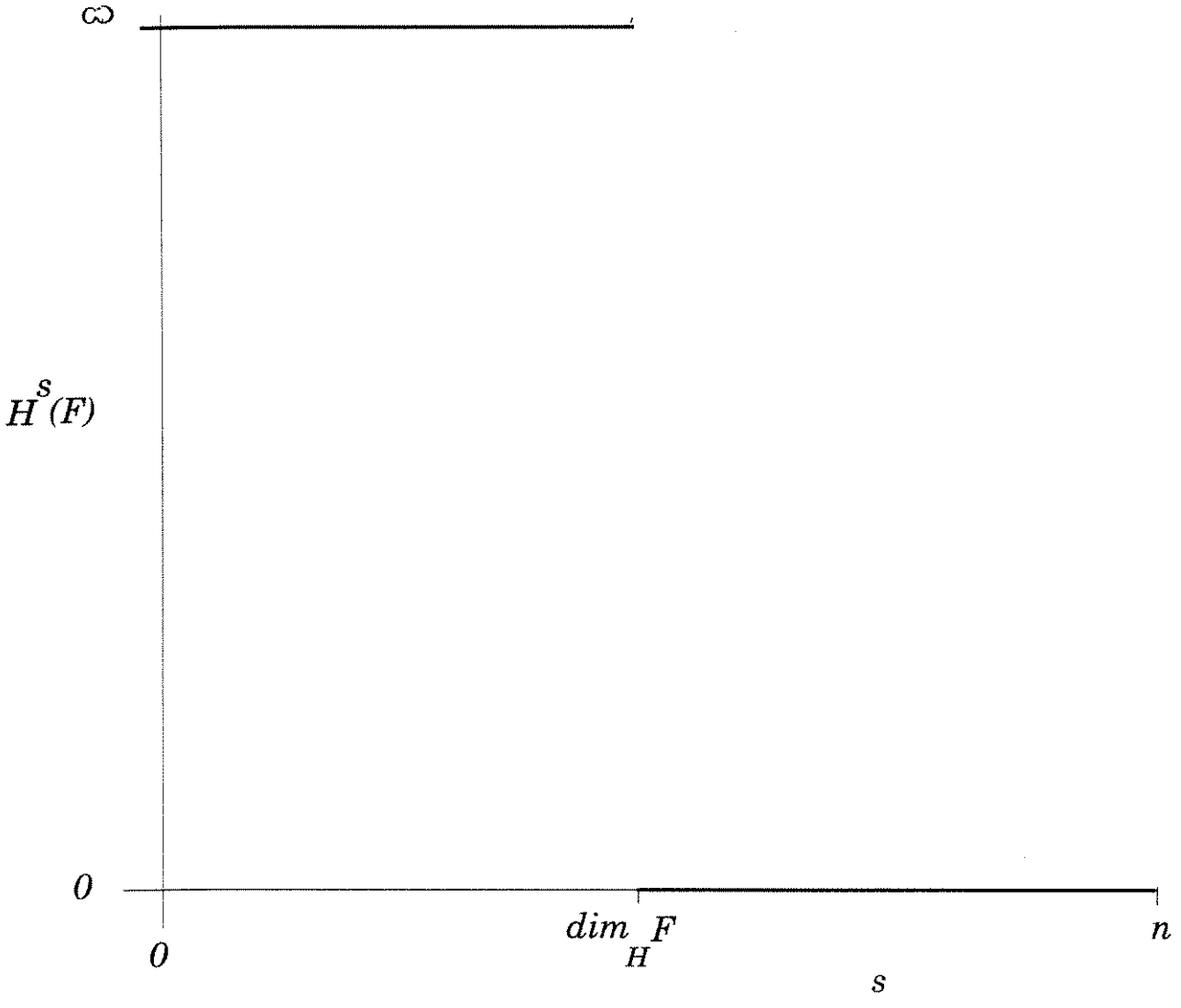

Figura 3.2: Gráfico de $\mathcal{H}^s(F)$ versus s. A dimensão de Hausdorff é o valor de s no qual ocorre o salto de ∞ para 0.

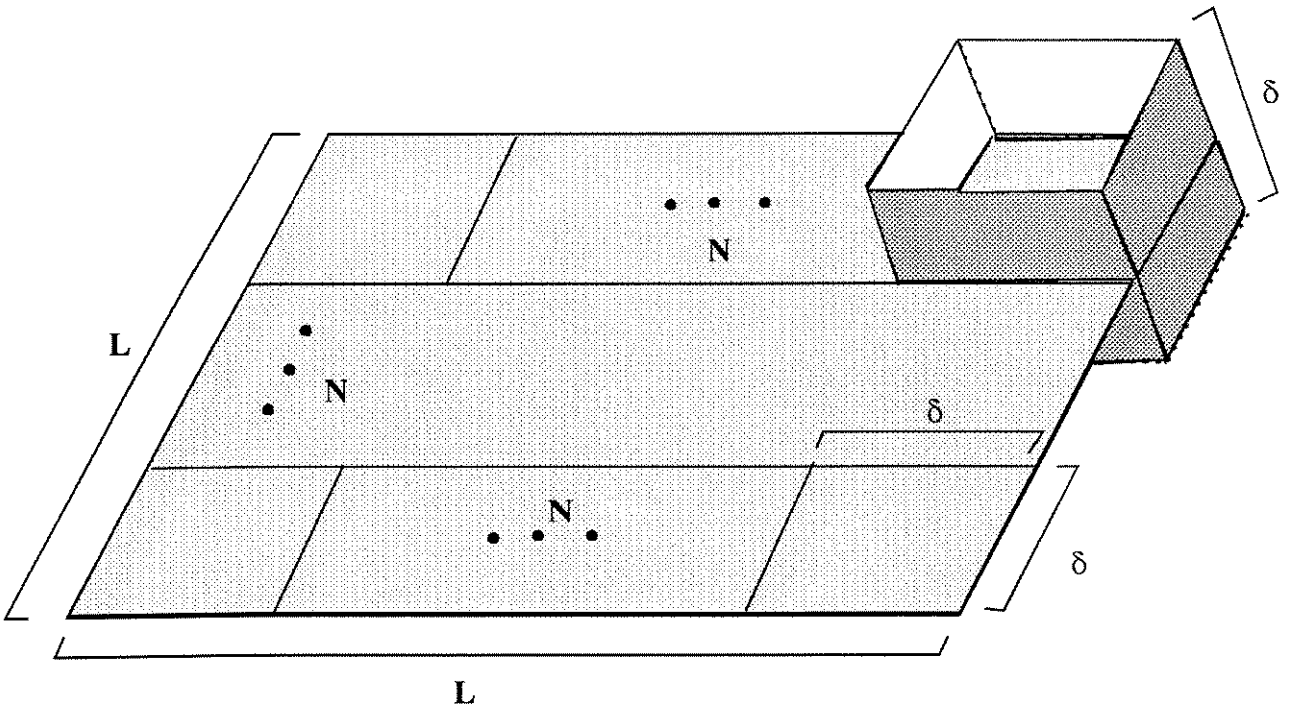

Figura 3.3: Cobertura de um retângulo por cubos de lado δ.

- $\mu(\emptyset) = 0$;
- $\mu(A) \leq \mu(B)$ se $A \subset B$;
- Se $A_1, A_2, \ldots$ é uma seqüência enumerável de conjuntos, então:

$$\mu(\bigcup_{i=1}^{\infty} A_i) \leq \sum_{i=1}^{\infty} \mu(A_i); \tag{3.6}$$

- $0 < \mu(\mathbf{R}^n) < \infty$.

Nesse caso, diz-se que o sub-conjunto limitado A de $\mathbf{R}^n$ no qual μ está definida possui massa $\mu(A)$. Intuitivamente, uma distribuição de massa pode ser obtida pegando-se uma certa massa e distribuindo-a de alguma forma no conjunto A. As condições exigidas pela definição de medida serão, então, automaticamente satisfeitas.

Relacionando a medida de Hausdorff e conceito de distribuição de massa temos o seguinte teorema:

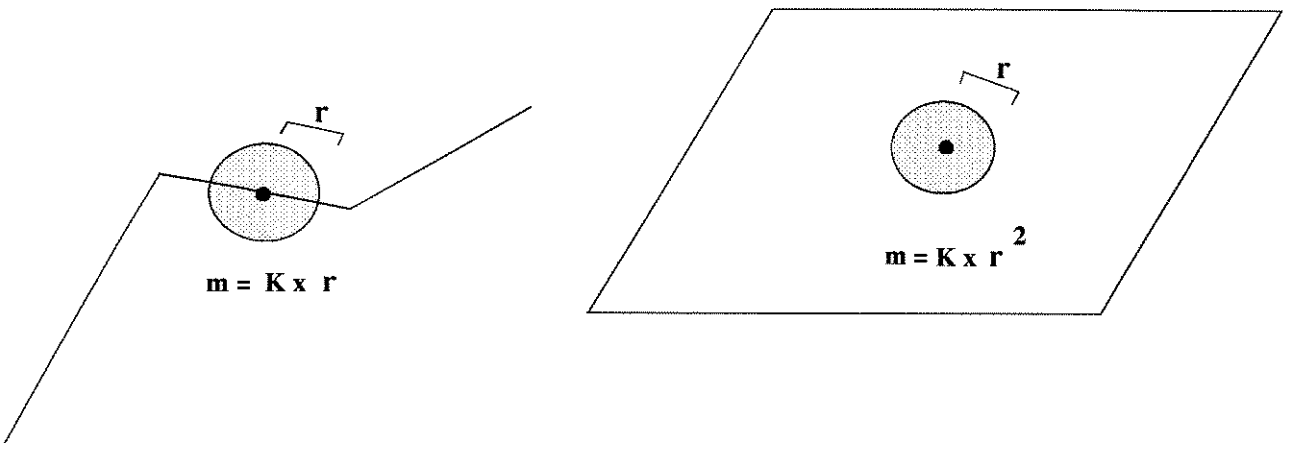

Figura 3.4: Duas distribuições homogêneas de massa.

Teorema 1 *Seja μ uma distribuição de massa em $\mathbf{R}^n$, $F \subset \mathbf{R}^n$ um conjunto de Borel, $B_r(x)$ uma bola n-dimensional de raio r centrada em um ponto x de F e $0 < c < \infty$ é uma constante.*

1. *Se $\overline{\lim}_{r \to 0} \mu(B_r(x))/r^s < c$ para todos os pontos $x \in F$, então $\mathcal{H}^s(F) \geq \mu(F)/c$;*
2. *Se $\overline{\lim}_{r \to 0} \mu(B_r(x))/r^s > c$ para todos os pontos $x \in F$, então $\mathcal{H}^s(F) \leq 2^s \mu(\mathbf{R}^n)/c$;*

Muito mais importante para o algoritmo que apresentaremos, pode ser derivado facilmente do teorema 1 acima o seguinte corolário:

Corolário 1 *A dimensão Hausdorff $dim_H F$ de um conjunto $F \subset \mathbf{R}^n$ pode ser calculada por $\lim_{r \to 0} \log \mu(B_r(x))/\log r$, se este limite existe.*

Intuitivamente, se o limite do corolário 1 acima assumir o valor s para os pontos x de um conjunto $F \subset \mathbf{R}^n$, podemos pensar que a massa de uma pequena vizinhança ao redor x cresce com potência s com o valor do raio dessa vizinhança. Ou seja, $\mu(B_r(x)) \approx K \times r^s$ para uma constante $K > 0$. Isso é ilustrado na figura 3.4, na qual mostramos duas distribuições homogêneas de massa. Na primeira delas, temos uma distribuição de massa linear, ao longo de segmentos de reta. Nesse caso, $\mu(B_r(x)) \approx K \times r^1$ e a dimensão Hausdorff do conjunto é 1. No segundo caso, temos uma distribuição ao longo de um retângulo, pela qual $\mu(B_r(x)) \approx K \times r^2$. A dimensão Hausdorff do retângulo é claramente 2.

Muito freqüentemente um conjunto $F \subset \mathbf{R}^n$ possui uma distribuição de massa natural μ. Se $\log \mu(B_r(x))/\log r \to s$ quando $r \to 0$ para todos os pontos $x \in F$, então a dimensão Hausdorff de F é s. Esse teorema pode ser usado para estimar praticamente a dimensão de um conjunto F com uma distribuição natural de massa. Para um ponto típico $x \in F$, pode-se estimar $\mu(F \cap B_r(x))$ para um conjunto de pequenos valores de r e calcular a inclinação de uma reta que interpole o gráfico de $\log \mu(F \cap B_r(x)) \times \log r$.

Capítulo 4

O Método

Nosso objeto de estudo serão as assim chamadas imagens topográficas, as quais são, simplesmente, um mapa bidimensional de *pixels* que armazenam, cada um, um número que representa a altura da imagem naquele ponto.

Em todo processo de captura e representação de imagens, enfrenta-se o problema da quantização que consiste em representar por um conjunto de valores discretos finitos os valores que a imagem assume, os quais podem ser contínuos na escala observada. Portanto, infinitos. Nesse processo, temos que mapear faixas de valores assumidos pela imagem num único valor representante daquela faixa. Esse mapeamento será tanto melhor quanto maior a quantidade de valores disponíveis para representar a imagem, o que depende do número de *bits* disponíveis para cada *pixel*.

Quando estudamos imagens como se fossem fractais, o problema complica-se um pouco mais. Como vimos, os fractais devem apresentar detalhes em todas as escalas de observação, ou seja, detalhes que são arbitrariamente pequenos. Quando capturamos e quantizamos uma imagem, naturalmente colocamos um limite inferior na faixa de escalas que podemos estudar, porque é impossível representar detalhes arbitrariamente pequenos com um conjunto limitado de valores. Por outro lado, se olharmos a representação de uma imagem em computador, veremos que ela não passa de um gráfico de barras, com um prisma de base retangular colocado sobre cada um de seus *pixels*. A altura do prisma e, portanto, o valor da imagem no *pixel* correspondente, é apenas um valor dentre os infinitos possíveis para os pontos localizados em sua base. Se esta base for pequena e se a imagem não sofrer oscilações muito bruscas, esse valor representativo estará, naturalmente, bastante próximo de todos os valores possíveis que o *pixel* poderia assumir. Entretanto, se uma dessas condições não for satisfeita, a representação facilmente prederá muitos dos detalhes da imagem original. Veja a figura 4.1, na qual mostramos um corte do *gráfico de barras* que representa uma imagem topográfica juntamente com o perfil da imagem original, antes de ser capturada e quantizada. Note que a altura da imagem original varia consideravelmente e apresenta valores freqüentemente muito distintos por toda a extensão do *pixel*, os quais diferem

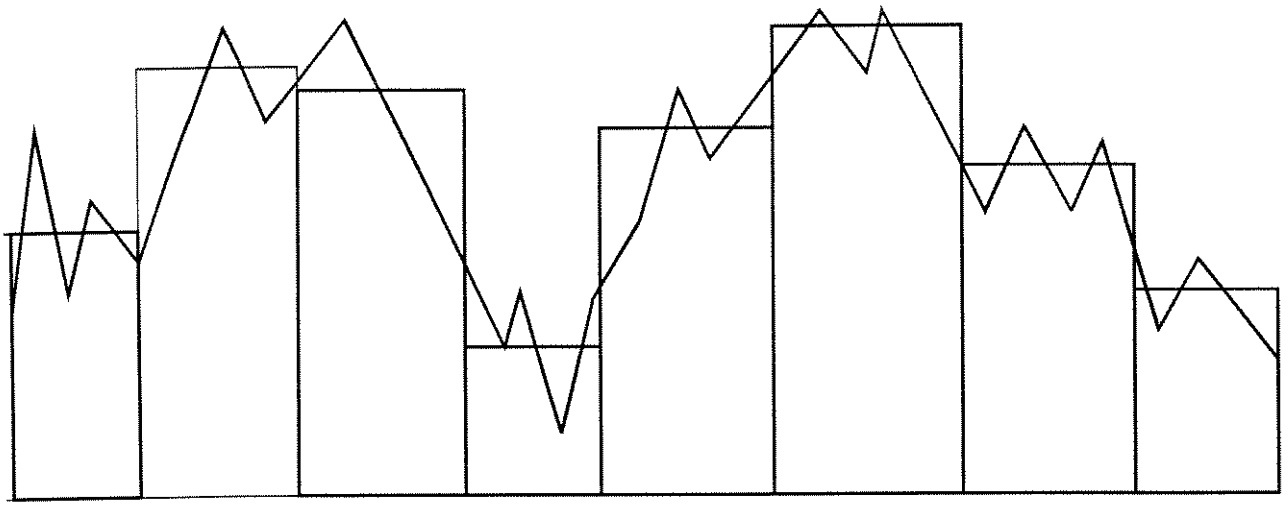

Figura 4.1: Perfil de uma imagem topográfica e sua representação em computador.

também do valor representativo do pixel, dado pela altura do prisma.

Por essas e outras razões que serão colocadas posteriormente, preferimos estudar as imagens topográficas do ponto de vista estatístico. Nessa abordagem, ao olharmos, numa determinada escala δ, para a vizinhança de um determinado *pixel* P, consideraremos que a altura da imagem para os *pixels* dentro de um quadrado de lado δ centrado em P não pode ser determinada com exatidão, assumindo, entretanto, valores que oscilam, com um determinada amplitude, ao redor de um valor médio. A figura 4.2 ilustra, através de um corte da imagem, esse conceito. Nela, consideramos que a altura da imagem para os *pixels* no interior do quadrado de *pixels* de lado δ oscila imprevisivelmente ao redor do valor médio T com amplitude A.

Quando estudamos uma imagem topográfica como uma representação de um fractal, sabemos que sua dimensão Hausdorff deve ficar entre 2 e 3. Quando esse valor é baixo (próximo de 2), a imagem é muito suave e assemelha-se a um plano. Quando esse valor é alto (próximo de 3), a superfície é altamente irregular. Neste último caso, podemos considerar intuitivamente que a superfície preenche uma parcela do espaço tridimensional que a circunda. E esse preenchimento é tanto maior quanto maior for sua dimensão fractal. Agora, quando olhamos a representação em computador de uma imagem topográfica, notamos que, por ser simplesmente um gráfico de barras (prismas), ela é, essencialmente, bidimensional. Ou seja, a superfície da imagem é reduzida a um conjunto finito de topos retangulares de prismas de alturas variáveis. Como cada um desses topos possui dimensão Hausdorff 2, a imagem inteira pode ser encarada como uma superfície bidimensional. Obviamente essa não é a abordagem correta. Um algoritmo que procura estimar a dimensão fractal de uma imagem topográfica deve extrair, o mais precisamente possível, da variação

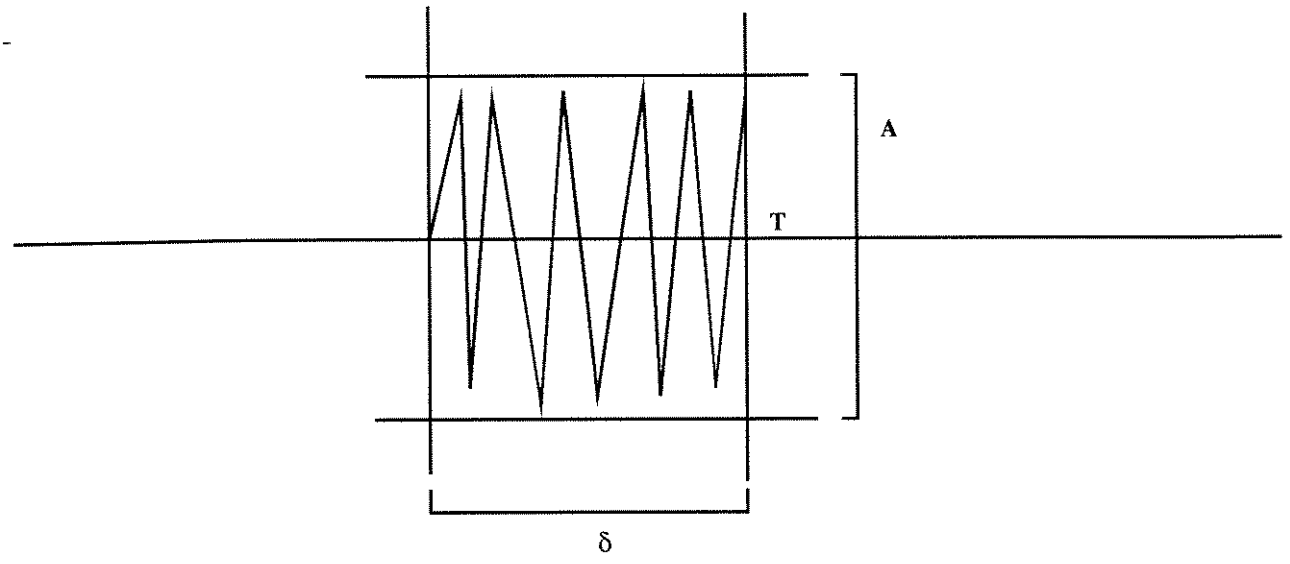

Figura 4.2: A altura da imagem oscila imprevisivelmente entre $T - A/2$ e $T + A/2$ dentro de um quadrado de lado δ.

das alturas dos prismas o grau de preenchimento do espaço que ela consegue.

Nesse sentido, a abordagem estatística da altura da imagem é muito útil. Ao observarmos o corte da imagem representado na figura 4.2, podemos considerar que a imagem preenche todo o espaço representado pela área de oscilação dentro do quadrado, uma vez que estamos supondo que não podemos prever a localização exata dos pontos superfície dentro do quadrado. Assim, ao analisarmos a imagem sob uma escala de observação δ, podemos segmentar o mapa de *pixels* em uma grade de quadrados de lado δ e considerar, dentro de cada quadrado, que a altura da imagem ocupa todo o espaço tridimensional entre dois valores de altura, um valor mínimo e um máximo. Observe a figura 4.3 para uma representação esquemática dessa abordagem.

Consideremos agora um pouco mais formalmente estas idéias. Dada uma escala de observação δ, consideremos que o mapa de *pixels* da imagem é segmentado em uma grade de quadrados de lado δ orientados de acordo com a disposição das linhas e colunas da imagem. Assim, por exemplo, a imagem pode ser dividida em $I \times J$ quadrados, cada um composto de M colunas e N linhas (veja a figura 4.4). Dentro de cada quadrado, composto pelas colunas $C_1 \ldots C_M$ e pelas linhas $L_1 \ldots L_N$, consideraremos que a altura da imagem oscila em torno do valor médio $T = \sum_{i=C_1}^{C_M} \sum_{j=L_1}^{L_N} H(i,j)/M \times N$, onde H(i,j) é a altura da imagem no *pixel* localizado na coluna i e linha j. Este valor não passa da média das alturas dos *pixels* da imagem. Consideraremos também que a imagem oscila, dentro do quadrado, entre os valores $T - h$ e $T + h$, onde h é o valor

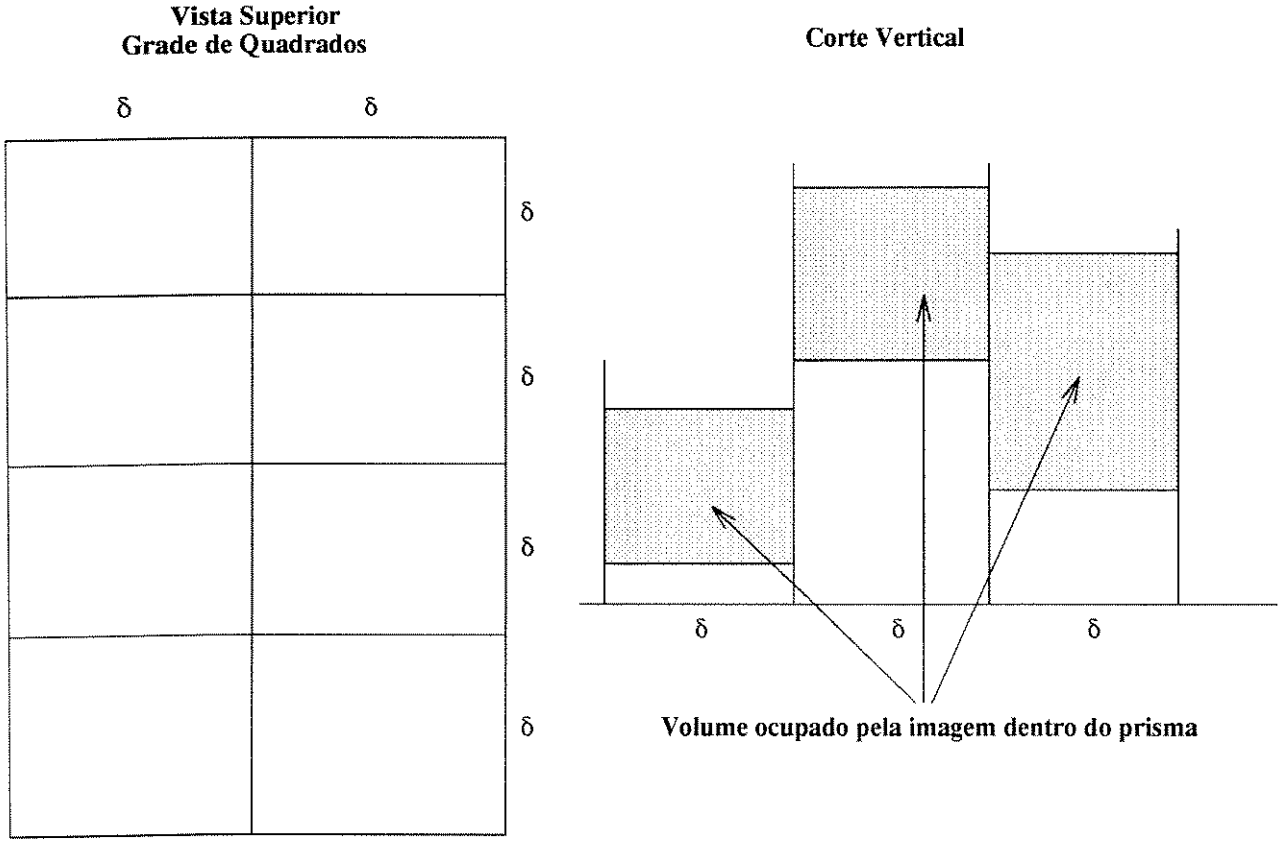

Figura 4.3: Segmentação da imagem por uma grade de quadrados. Dentro de cada quadrado, a altura da imagem localiza-se entre um valor mínimo e um máximo.

do desvio padrão da altura da imagem para os *pixels* do quadrado. Ou seja, $h = \sqrt{var(H(i,j))}$ para $i = C_1 \ldots C_M$ e $j = L_1 \ldots L_N$. Assim, dado que E é o operador estatístico que devolve o valor esperado de uma variável aleatória, temos:

$$h = E(H^2) - E(H)^2 = \frac{\sum_{i=C_1}^{C_M} \sum_{j=L_1}^{L_N} H(i,j)^2}{M \times N} - \left(\frac{\sum_{i=C_1}^{C_M} \sum_{j=L_1}^{L_N} H(i,j)}{M \times N}\right)^2 \tag{4.1}$$

Podemos agora definir o conceito de *massa em escala* δ. Veja a figura 4.5, na qual mostramos dois cubos de lados de tamanho δ com bases cujas projeções sobre o plano da imagem coincidem com quadrados da grade de segmentação do mapa de *pixels*. Esses quadrados possuem como centros os *pixels* P_1 e P_2 (estamos supondo que os quadrados possuem números ímpares de colunas e linhas). Os cubos estão posicionados de tal forma que os pontos médios de seus lados verticais estejam contidos nos planos definidos pelos valores médios T das alturas dos *pixels* da imagem dentro dos respectivos quadrados da grade. As áreas de oscilação da imagem dentro dos quadrados, as quais possuem amplitudes verticais iguais a $2 \times h_1$ e $2 \times h_2$ são marcadas pelas áreas hachuradas. No caso da esquerda, temos que a amplitude de oscilação $2 \times h_1$ é menor do que δ e a imagem pode ser imaginada como estando totalmente contida dentro do cubo de lado δ. No caso da direita, a amplitude de oscilação $2 \times h_2$ é maior do que δ e a imagem oscila fora do cubo de lado δ. Chamaremos o volume determinado pela interseção da área hachurada com o cubo de lado δ de *massa em escala* δ dentro de uma vizinhaça de *diâmetro* δ ao redor do *pixel* em questão. Representaremos esse valor por $\mu_P^\delta(d)$, onde P é o *pixel* central do quadrado e $d = \delta$ é o *diâmetro* (lado do cubo) da vizinhança. No caso da esquerda, temos $\mu_{P_1}^\delta = 2 \times h_1 \times \delta^2$. No caso da direita, temos $\mu_{P_2}^\delta = \delta \times \delta^2$.

Vejamos agora que a *massa em escala* δ pode ser usada como uma medida do grau de irregularidade de *tamanho* δ da superfície. Veja a figura 4.6. Na parte da esquerda mostramos, pela área hachurada, o valor de massa $\mu_{P_1}^\delta(\delta)$ para o *pixel* P_1. Veja que os tamanhos das irregularidades são da ordem de grandeza do valor de δ. Veja também que $\mu_{P_1}^\delta(\delta)$ é razoavelmente grande quando comparado com a massa do cubo de lado δ (δ^3). No caso da direita, $\mu_{P_2}^\delta(\delta)$ é bastante pequeno quando comparado com o volume do cubo de lado δ e a superfície apresenta-se muito pouco irregular, quando observada sob essa escala. Esses são dois exemplos de como a massa em escala δ, quando comparada com a massa δ^3, mede o grau das irregularidades de tamanho δ da superfície. Veja também a figura 4.7, na qual mostramos uma superfície com irregularidades muito maiores que o valor de δ. Também nesse caso, a massa $\mu_P^\delta(\delta)$ é pequena quando comparada com o volume δ^3. Em resumo, a massa em escala δ é relativamente maior quando os tamanhos das irregularidades da superfície são da ordem de grandeza de δ.

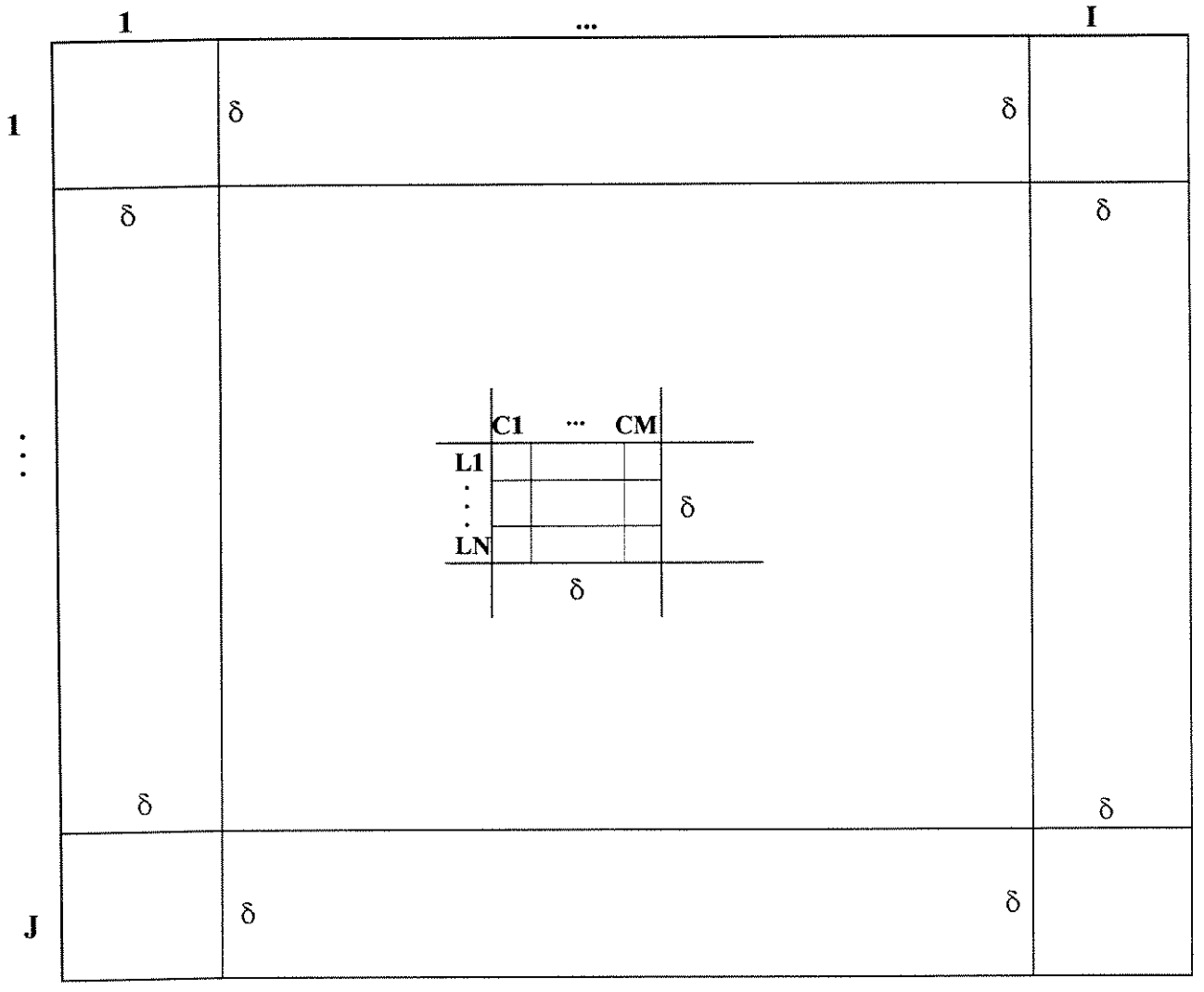

Figura 4.4: Segmentação da imagem por uma grade de quadrados. Cada quadrado possui lado δ e engloba M colunas e L linhas.

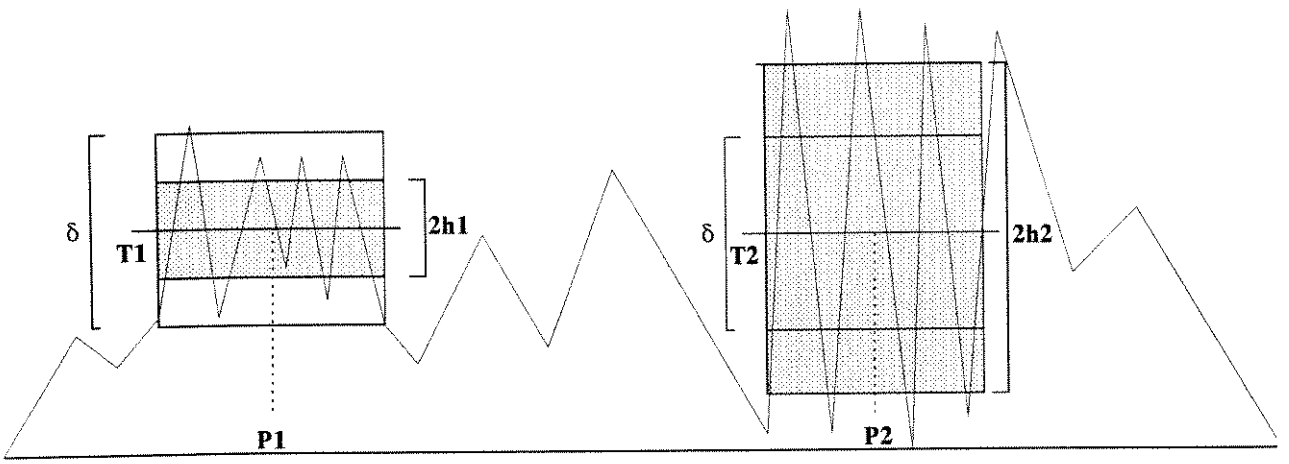

Figura 4.5: O conceito de massa em escala δ.

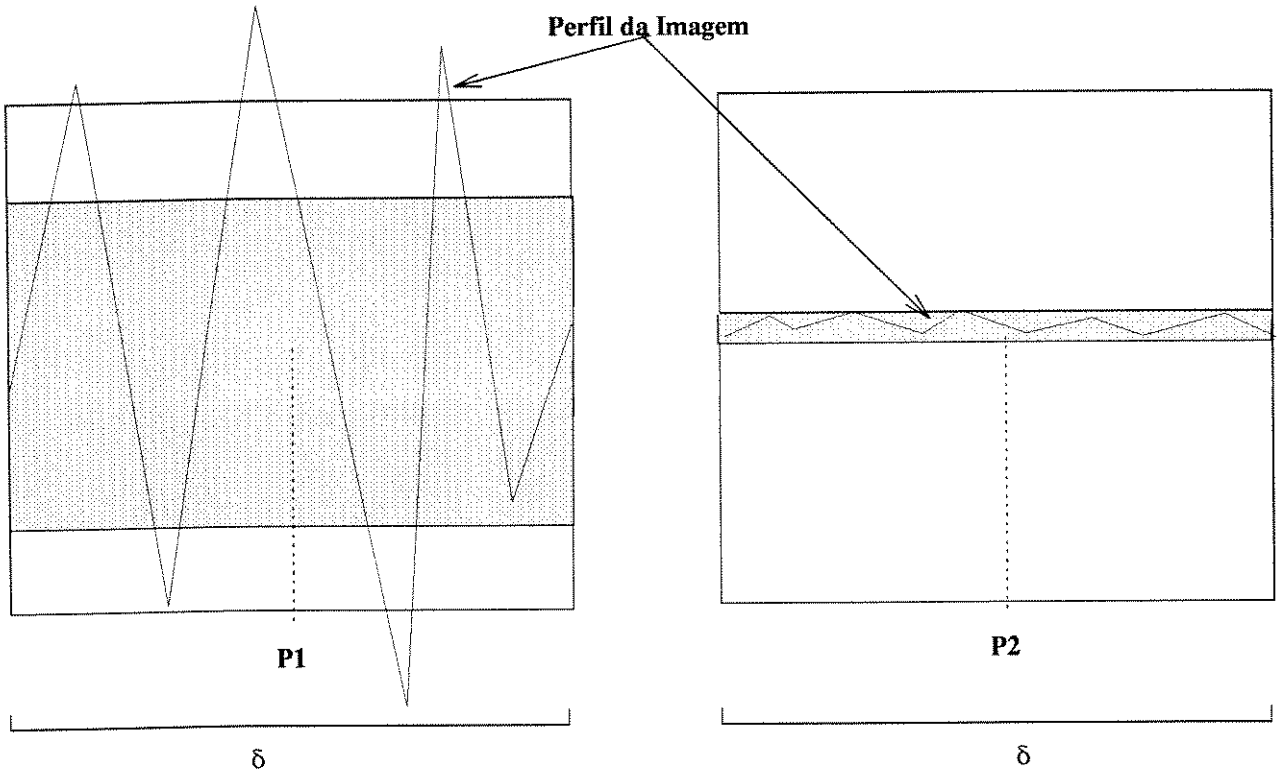

Figura 4.6: A massa em escala δ como medida de irregularidade (I).

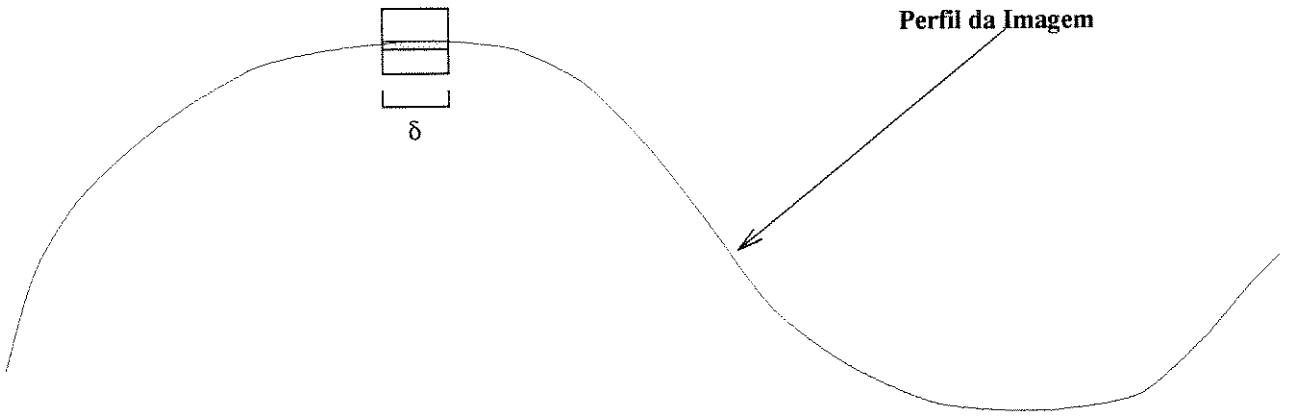

Figura 4.7: A massa em escala δ como medida de irregularidade (II).

Vejamos agora como poderíamos calcular a *massa em escala* δ ($\mu_P^\delta(d)$) para uma vizinhança de diâmetro arbitrário d ao redor de um pixel P. Para isso, veja a figura 4.8 na qual mostramos um perfil da segmentação de uma imagem por uma grade de quadrados, com as áreas hachuradas indicando as oscilações da superfície dentro dos quadrados. Para calcular a massa da superfície dentro de uma vizinhança de diâmetro d ao redor do *pixel* P, devemos somar as massas em escala δ das frações dos quadrados que ficam dentro desta vizinhança. No caso mostrado na figura, somanos as massas dos quadrados Q_2 e Q_3 com parcelas das massas dos quadrados Q_1, e Q_4 (além dos quadrados que não aparecem na figura). O cálculo dessas parcelas é feito pela porcentagem da área do quadrado que fica dentro da vizinhança. Veja a figura 4.9 na qual mostramos uma visão de cima da segmentação do mapa de *pixels* e uma vizinhança de diâmetro d. Nesse caso, a parcela da massa da superfície dentro do quadrado Q_1 que fará parte da somatória no cálculo da massa dentro da vizinhança é dada pela razão entre a área hachurada e a área do quadrado, no caso δ^2.

Considere agora uma grade de $(2n+1) \times (2n+1)$ quadrados de lado δ cada um e dispostos ao redor de um *pixel* P conforme a figura 4.10. Ou seja, os quadrados da grade devem possuir lados paralelos à disposição das linhas e colunas de *pixels* da imagem e o *pixel* P deve estar centralizado no quadrado do meio. Cada quadrado pode ser identificado por $Q_{(i,j)}$, onde $i = 0 \ldots 2n$ e $j = 0 \ldots 2n$. O quadrado do meio é $Q_{(n,n)}$. Chamemos de $\mu^\delta_{(i,j)}(\delta)$ a *massa em escala* δ do quadrado $Q_{(i,j)}$. Seja Q_P^d um quadrado de lado d centralizado também no *pixel* P, de forma que $d \leq (2n+1)\delta$. Chamemos de $F_P^d(i,j)$ a porcentagem da área do quadrado $Q_{(i,j)}$ que fica dentro de Q_P^d. A *massa em escala* δ da superfície da imagem para os *pixels* dentro do quadrado Q_P^d é:

$$\mu_P^\delta(d) = \sum_{i=0}^{2n+1} \sum_{j=0}^{2n+1} \mu^\delta_{(i,j)}(\delta) \times F_P^d(i,j). \tag{4.2}$$

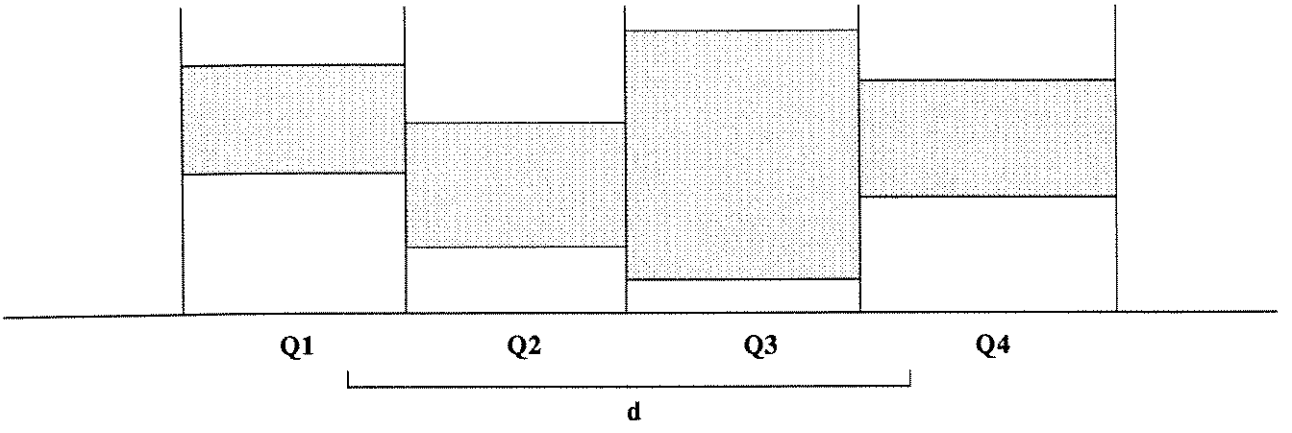

Figura 4.8: O cálculo de $\mu_P^\delta(d)$ (I).

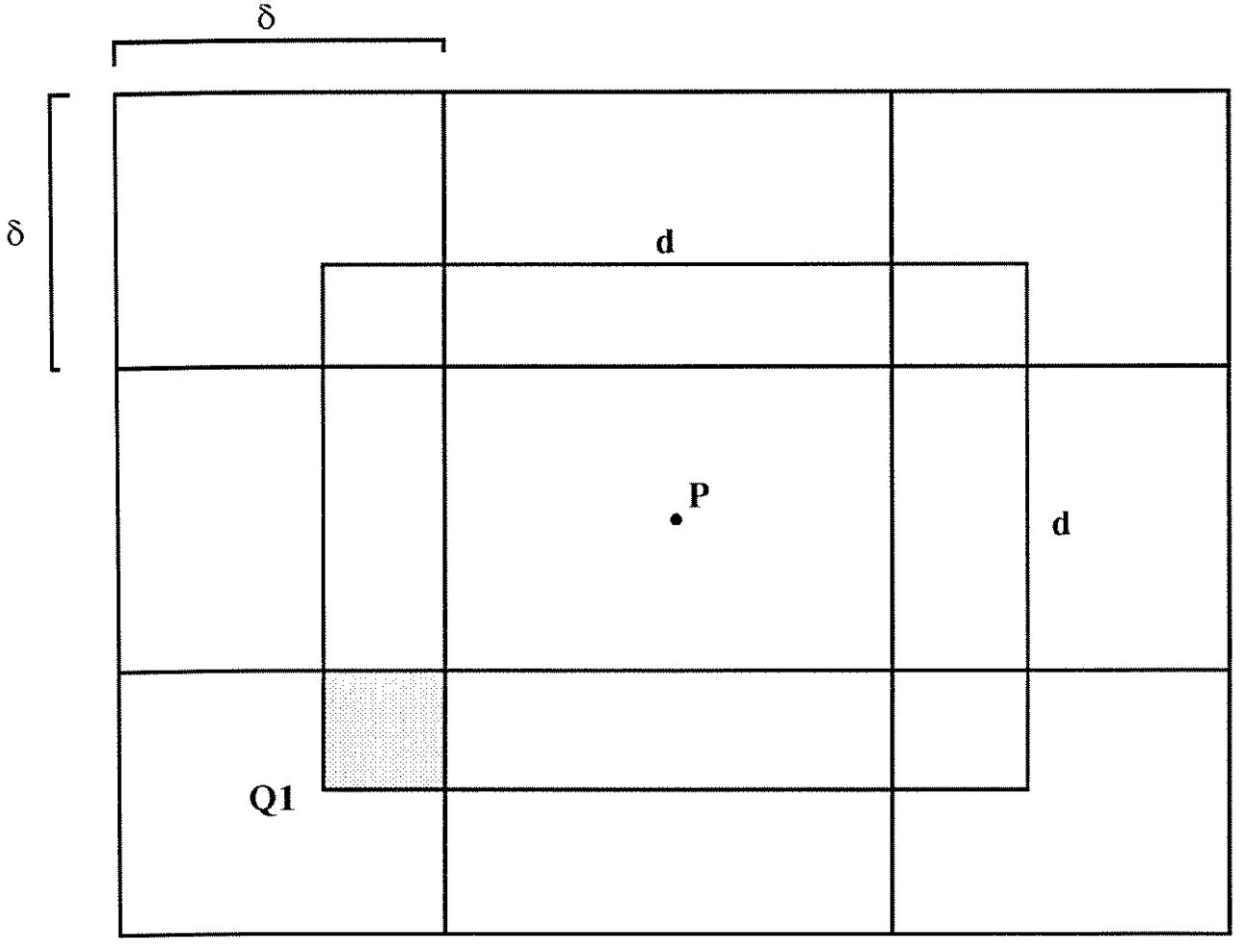

Figura 4.9: O cálculo de $\mu_P^\delta(d)$ (II).

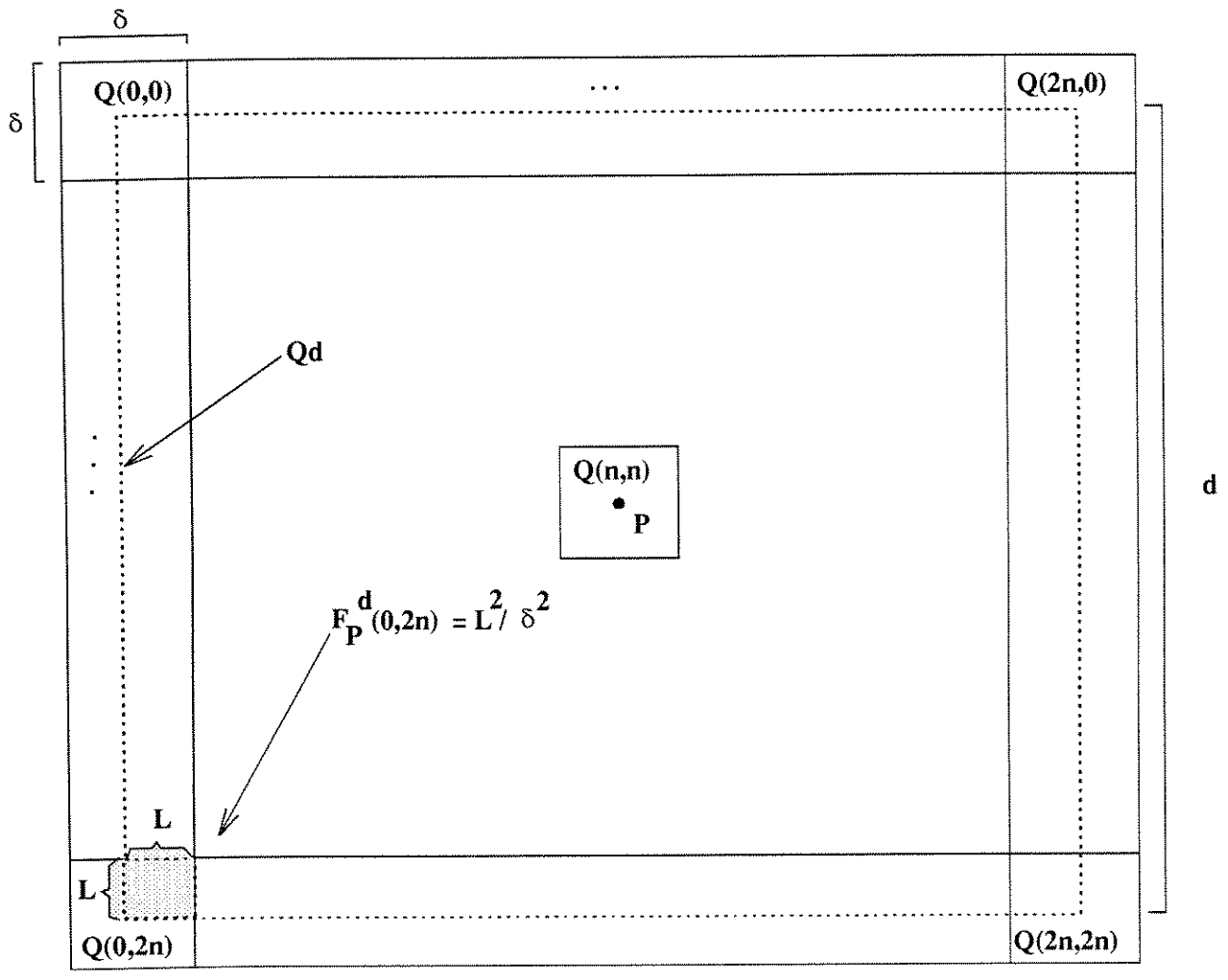

Figura 4.10: O cálculo de $\mu_P^\delta(d)$ (III).

O valor $\mu_P^\delta(d)$ mede a *massa em escala* δ da superfície da imagem para os *pixels* dentro de uma vizinhança de diâmetro d ao redor do *pixel* P.

Observe que $\mu_P^\delta(d)$ calcula uma medida de massa apenas para as partes da superfície definidas por um conjunto de *pixels* dentro de um quadrado de lado d centralizado num *pixel* P, dada uma determinada escala δ. Poderíamos pensar em generalizar $\mu_P^\delta(d)$ para calcular uma medida de massa $\mu(S)$ para qualquer conjunto $S \subset \mathbf{R}^3$. Isso pode ser feito facilmente, considerando quadrados que envolvem a projeção desse conjunto sobre o plano definido pelo mapa de *pixels* da imagem. Em seguida, poderíamos tentar provar que $\mu(S)$ define uma distribuição de massa em $\mathbf{R}^3$ e usar o corolário 1 para calcular a dimensão fractal de pontos da imagem. Entretanto, o uso literal desse corolário dificilmente seria útil nesse caso, uma vez que pelo processo de segmentação do

mapa de *pixels* da imagem em quadrados de lado δ, a imagem em si aparece constituída como um conjunto de paralelepípedos, cada um correspondendo a um quadrado da segmentação. Ou seja, uma vizinhança pequena de pontos da imagem observada sob o ponto de vista da segmentação teria muito freqüentemente comportamento tridimensional e possuiria dimensão igual a 3.

O corolário 1 tenta medir o expoente de crescimento da massa de uma pequena vizinhança de um ponto com o raio dessa vizinhança. Para usar essa idéia de forma útil por meio da aplicação do operador $\mu_P^\delta(d)$ para calcular a massa de uma vizinhança de diâmetro d, devemos de alguma forma considerar uma seqüência pequena de valores de d e, em seguida, usar a fórmula 1. Os valores de d não podem ser muito pequenos pela razão analisada no parágrafo anterior. Por outro lado, os valores de d não devem ser muito grandes, uma vez que o corolário 1 exige o cálculo de um limite quando o raio da vizinhança tende a zero. Se fizermos com que os valores de d sejam muito grandes, estaremos perdendo a característica de levar em conta no cálculo apenas vizinhanças pequenas ao redor do ponto em questão.

Certamente se tomarmos valores de d todos menores que δ, teremos $\mu_P^\delta(d) = Kd^2$, para alguma constante K. Isso faz com que a inclinação do gráfico $\log \mu_P^\delta(d) \times \log d$ seja igual a 2. Assim, pelo próprio processo de segmentação da imagem em quadrados de lado δ, os valores de d devem ser tais que não sejam muito menores que δ. Por outro lado, se forem muito maiores que δ teremos o problema de considerar uma vizinhança muito grande. Assim, parece uma boa idéia considerar uma seqüência de valores de d de tal forma que sejam todos da ordem de grandeza de δ. Uma idéia para a seqüência de valores de d é fazer $d = \delta, 3\delta, \ldots, (2n+1)\delta$, para valores pequenos de n, por exemplo, $n = 1$ ou $n = 2$. Outra é fazer, para um número real $0 < f < 1$ e para um valor pequeno de n ($n = 1, 2, \ldots$ *ou* 5), de forma que $nf < 1$, com que :

$$d = -nf\delta, -(n-1)f\delta, \ldots, -f\delta, \delta, f\delta, \ldots, (n-1)f\delta, nf\delta. \tag{4.3}$$

Neste trabalho, estaremos estudando a segunda seqüência, por considerar uma vizinhança bem menos ampla que a primeira.

Existe uma vantagem prática no cálculo da amplitude de oscilação da imagem dentro de um quadrado de lado δ por meio do cálculo da variância da altura, de acordo com a equação 4.1. Veja a figura 4.11. Nela mostramos dois quadrados de $n \times n$ *pixels*. O primeiro composto pela regiões X e Y e o segundo composto pelas regiões Y e Z. As regiões X e Z são formadas por uma única coluna de *pixels*, de forma que o quadrado da direita é obtido do da esquerda por um deslocamento horizontal de uma coluna. A variância da altura dos *pixels* no quadrado da esquerda

é $var(H_{X\cup Y}) = E(H^2_{X\cup Y}) - \{E(H_{X\cup Y})\}^2$, onde E é operador estatístico que devolve a média de um conjunto de valores e H_S é o conjunto de valores de alturas para os *pixels* do conjunto S e H^2_S é o conjunto dos quadrados dos valores de alturas para os *pixels* dos conjunto S. Assim temos:

$$var(H_{X\cup Y}) = E(H^2_{X\cup Y}) - (E(H_{X\cup Y}))^2 \tag{4.4}$$

$$var(H_{Y\cup Z}) = E(H^2_{Y\cup Z}) - (E(H_{Y\cup Z}))^2 \tag{4.5}$$

$$E(H^2_{Y\cup Z}) = E(H^2_{X\cup Y}) - \sum_{i=1}^{i=n} H^2(1,i)/n^2 + \sum_{i=1}^{i=n} H^2(n+1,i)/n^2 \tag{4.6}$$

$$E(H_{Y\cup Z}) = E(H_{X\cup Y}) - \sum_{i=1}^{i=n} H(1,i)/n^2 + \sum_{i=1}^{i=n} H(n+1,i)/n^2 \tag{4.7}$$

Se temos os valores $E(H^2_{X\cup Y})$ e $E(H_{X\cup Y})$, podemos calcular facilmente os valores $E(H^2_{Y\cup Z})$ e $E(H^2_{Y\cup Z})$ e, portanto, a variância $var(H_{Y\cup Z})$, através da realização de $O(n)$ operações aritméticas, onde n é o número de *pixels* do lado do quadrado e O é o operador de complexidade computacional da Teoria da Computação. Dado, então, que temos já calculados o valor esperado e o quadrado do valor esperado dos *pixels* dentro de um quadrado centrado em um dado *pixel* (i,j), podemos obter em $O(n)$ o valor esperado, o quadrado do valor esperado e a variância dos valores dos *pixels* do quadrado centrado em qualquer dos *pixels* $(i+1,j),(i-1,j),(i,j-1)$ *ou* $(i,j+1)$. Não precisamos recalcular em $O(n^2)$ a variância para cada um desses quadrados.

O mesmo raciocínio pode ser aplicado quando temos dois quadrados de *pixels* com mesmo centro. Observe a figura 4.12 na qual vemos um quadrado de lado δ_1 com $(n-2)\times(n-2)$ *pixels*, indicado pela região B, e um quadrado de lado δ_2 com $n \times n$ *pixels*, obtido pela união das regiões A e B. A região hachurada A é composta por um anel de *pixels* e é exatamente a diferença dos *pixels* entre o quadrado maior e o quadrado menor. Temos:

$$var(H_B) = E(H_B^2) - (E(H_B))^2 \tag{4.8}$$

$$var(H_{A\cup B}) = E(H_{A\cup B}^2) - (E(H_{A\cup B}))^2 \tag{4.9}$$

$$E(H_{A\cup B}^2) = \frac{(n-2)^2}{n^2} \times E(H_B^2) + \sum_{(i,j)\in A} H^2(i,j)/n^2 \tag{4.10}$$

$$E(H_{A\cup B}) = \frac{(n-2)^2}{n^2} \times E(H_B) + \sum_{(i,j)\in A} H(i,j)/n^2 \tag{4.11}$$

Ou seja, se temos os valores $E(H_B^2)$ e $E(H_B)$ para o quadrado menor, podemos calcular em $O(n)$ os valores $E(H_{A\cup B}^2)$, $E(H_{A\cup B})$ e $var(H_{A\cup B})$ para o quadrado maior, onde n é o número de *pixels* do lado do quadrado maior. Isso é bem melhor do que obter, por meio do cálculo direto, a variância dos valores dos *pixels* do quadrado maior em $O(n^2)$.

Finalmente, podemos agora descrever como implementar as duas funcionalidades descritas na seção 2. Na primeira delas, queremos calcular a irregularidade de um *pixel* P da superfície para várias escalas de observação. Para isso, fazemos o seguinte:

1. Posicione uma grade de $(2n+1) \times (2n+1)$ quadrados de lado δ de forma que o quadrado central tenha como centro o *pixel* P, de acordo com as figuras 4.9 e 4.10. O valor de n deve ser tal que $(2n+1)\delta \geq d$, para o maior valor d das vizinhanças escolhidas abaixo.
2. Calcule as médias dos valores dos *pixels* e dos quadrados dos valores dos *pixels* para cada um dos quadrados da grade.
3. Calcule a variância dos valores dos *pixels* para cada um dos quadrados da grade.
4. Calcule $\mu_{(i,j)}^{\delta}(\delta)$ para cada um dos quadrados $Q_{(i,j)}$ da grade.
5. Usando a equação 4.2, calcule $\mu_P^{\delta}(d)$, para um seqüência de valores 4.3 de d.
6. Calcule a inclinação da reta que melhor adapta-se aos pontos do gráfico $\log \mu_P^{\delta}(d) \times \log d$. Esse valor fornece o grau de irregularidade da superfície no *pixel* P e para a escala δ.
7. Aumente os lados dos quadrados por um certo número de *pixels*. Calcule o novo valor de escala correspondente a esse novo lado. Use as equações 4.8, 4.9, 4.10 e 4.11 para calcular as novas médias dos valores dos *pixels* e dos quadrados dos valores dos *pixels* para os quadrados.
8. Repita o procedimento a partir do passo 3 até que o valor da escala δ seja maior ou igual à diferença entre o maior e menor valor de *pixels* da imagem.

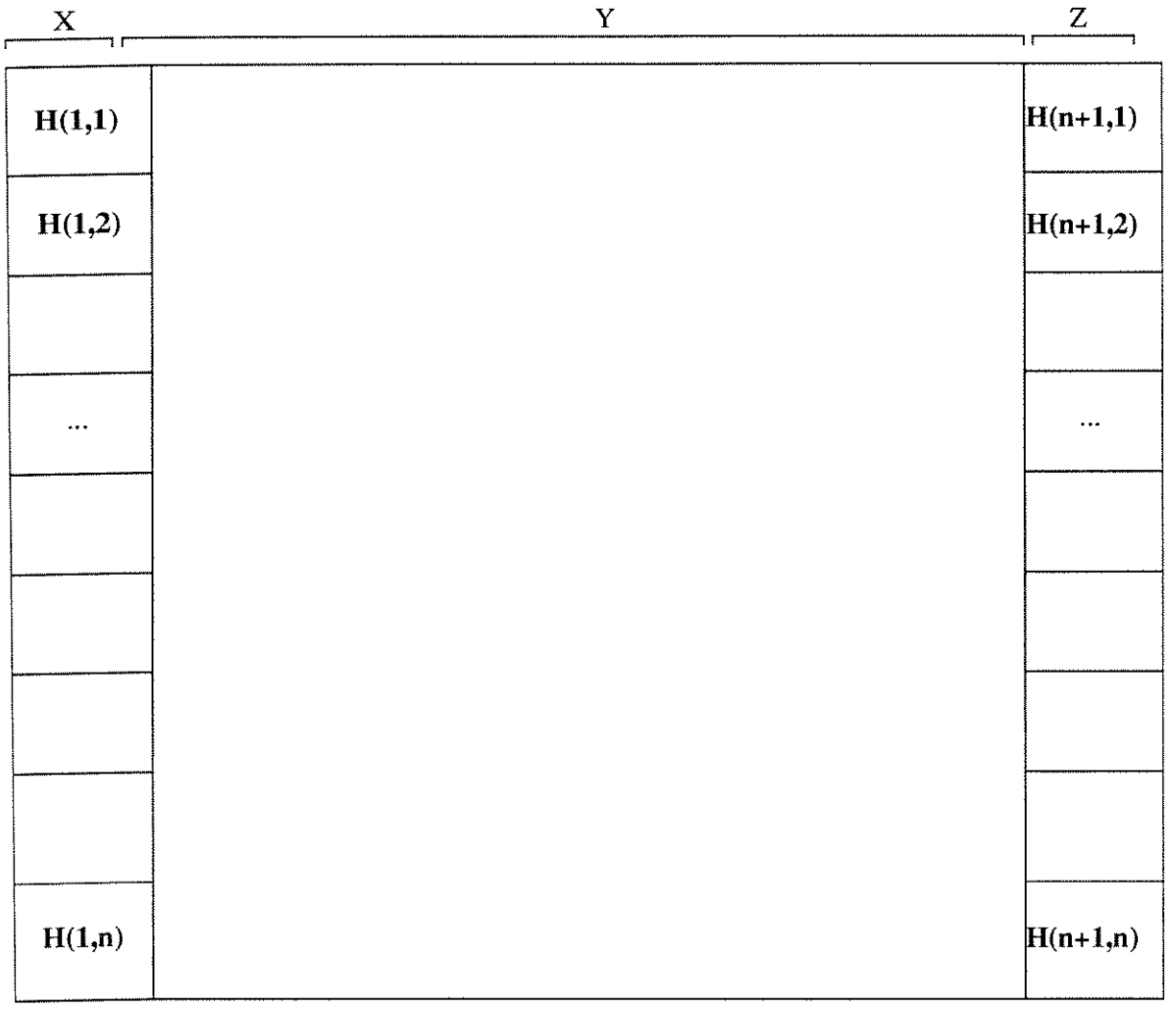

Figura 4.11: Como aproveitar a adjacência de quadrados de *pixels* no cálculo da variância de seus valores (I).

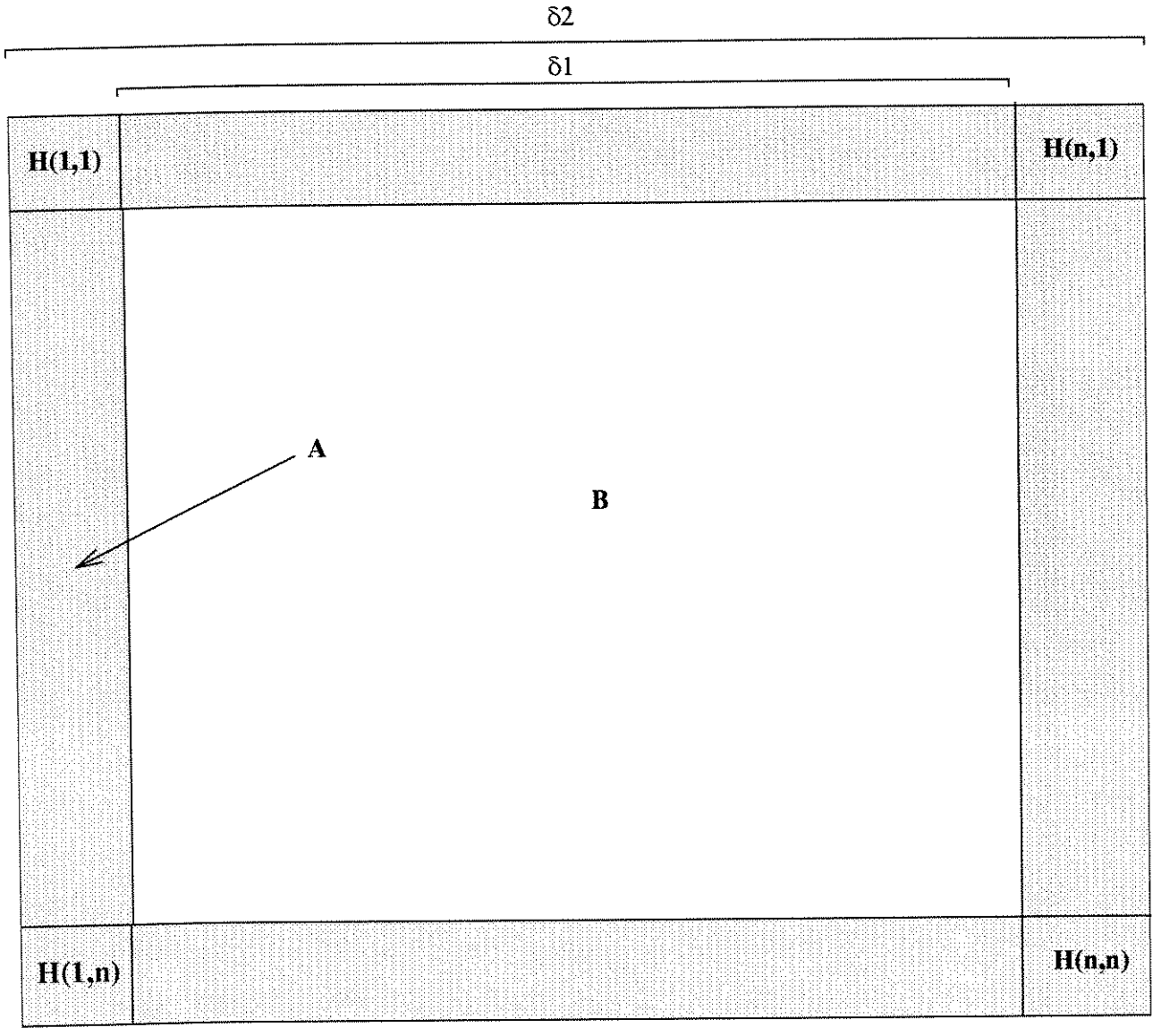

Figura 4.12: Como aproveitar a adjacência de quadrados de *pixels* no cálculo da variância de seus valores (II).

Para implementar a segunda funcionalidade, ou seja, calcular os valores de irregularidades de todos os *pixels* para uma dada escala de observação δ (mapa de dimensões), basta fazer:

1. Posicione uma grade de $(2n+1) \times (2n+1)$ quadrados de lado δ de forma que o quadrado central tenha como centro o *pixel* no canto superior esquerdo da imagem, ou seja, do *pixel* de coluna $i = 0$ e linha $j = 0$. O valor de n deve ser tal que $(2n+1)\delta \geq d$, para o maior valor d de diâmetros das vizinhanças d escolhidas abaixo.
2. Calcule as médias dos valores dos *pixels* e dos quadrados dos valores dos *pixels* para cada um dos quadrados da grade.
3. Calcule a variância dos valores dos *pixels* para cada um dos quadrados da grade.
4. Calcule $\mu^{\delta}_{(i,j)}(\delta)$ para cada um dos quadrados $Q_{(i,j)}$ da grade.
5. Usando a equação 4.2, calcule $\mu^{\delta}_{P}(d)$, para a seqüência de valores 4.3 de d.
6. Calcule a inclinação da reta que melhor adapta-se aos pontos do gráfico $\log \mu^{\delta}_{P}(d) \times \log d$. Esse valor fornece o grau de irregularidade da superfície no *pixel* P e para a escala δ.
7. Se i for par e j não for a última coluna da linha, desloque os quadrados da grade uma coluna para a direita, de forma que o novo centro da grade seja o *pixel* P com coluna $i+1$ e linha j. Se i for par e j for a última coluna da linha, desloque os quadrados da grade uma linha para baixo. Se i for ímpar e se j não for a primeira coluna da linha, desloque os quadrados da grade uma coluna para a esquerda. Se i for ímpar e se j for a primeira coluna da linha, desloque os quadrados da grade uma linha para baixo. Em cada um desses deslocamentos, utilize as equações 4.4, 4.5, 4.6 e 4.7 para calcular as novas médias dos valores e dos quadrados dos valores dos *pixels* dos quadrados da grade.
8. Repita o procedimento a partir do passo 3.

Existem vários detalhes envolvidos nestas descrições. Um deles é como escolher a reta que melhor adapta-se aos pontos do gráfico $\log \mu^{\delta}_{P}(d) \times \log d$. Neste assunto, é bastante instrutivo o trabalho [HLD94], no qual várias considerações e análises são realizadas para tentar identificar a melhor maneira de escolha. Não pretendemos implementar uma maneira sofisticada para realizar essa escolha e estaremos usando nas análises que faremos nas seções seguintes o método mais imediato, ou seja, a escolha da reta será feita pelo método dos quadrados mínimos [PFTV89] aplicado aos pontos do gráfico $\log \mu^{\delta}_{P}(d) \times \log d$. Os outros detalhes são, basicamente, detalhes de implementação, para os quais não apresentaremos descrição. Entretanto, eles podem ser facilmente apreendidos por um simples estudo do código e de seus comentários que acompanham esta descrição.

Por fim, um dos aspectos envolvidos no cálculo do espectro de irregularidades do ponto de vista do usuário do sistema é como escolher o *pixel* a ser tomado como base de cálculo. Se a imagem for bastante homogênea com relação aos graus e tipos de irregularidade ao longo de seu plano base, quaisquer dos *pixels* da imagem levarão a espectros muito parecidos. Dessa forma, a escolha do *pixel* pode ser aleatória. Se os graus de irregularidade variarem bastante ao longo do mapa de *pixels* da imagem, os espectros de irregularidade serão bastante distintos. Nesse caso, aparecerão variações bruscas no espectro de irregularidades devido à localização do *pixel* escolhido. Por exemplo, se o *pixel* escolhido localizar-se a uma distância D de uma cadeia de irregularidades na superfície da imagem, haverá uma variação brusca no espectro quando a escala for aproximadamente igual a D. Caberá ao usuário do sistema dar a interpretação correta a esse valor de escala. Por outro lado, se houver na superfície da imagem uma quantidade significativa de irregularidades de mesmo tamanho (mesma altura, largura ou profundidade) distribuídas de forma razoavelmente homogênea ao longo do mapa de *pixels*, os espectros calculados para diferentes pontos da superfície da imagem apresentarão variações de forma a indicar o tamanho dessas irregularidades. Ou seja, mesmo que os espectros sejam diferentes, as escalas em que ocorrem quantidades significativas de irregularidades homogeneamente distribuídas serão refletidas no espectro de irregularidades. Em resumo, o usuário do sistema deverá adquirir experiência para dar interpretações corretas às variações do espectro.

Capítulo 5

O Ambiente Fraktal

Com o objetivo de analisar o algoritmo descrito na seção anterior e de fornecer uma base de *software* para teste e análise de outros algoritmos similares, elaboramos o ambiente *Fraktal.*

Este ambiente é composto de uma biblioteca de classes, elaborada de acordo com a filosofia de Orientação a Objetos, que permite que novos algoritmos que trabalham com o cálculo da dimensão fractal ou rugosidade de imagens topográficas sejam desenvolvidos facilmente em um ambiente de janelamento, em especial, em um ambiente *X11.*

A biblioteca foi elaborada de forma que existam duas camadas de classes. Uma delas, que implementa o núcleo da estrutura do sistema, independe da biblioteca de janelamento (XView, Motif, etc.) usada na implementação. A segunda, implementada a partir da derivação das classes da primeira, implementa a funcionalidade dependente dessas bibliotecas. Existe também uma terceira categoria de classes, independente desta hierarquia, que é encarregada de fazer o trabalho real de cálculo da irregularidade e de implementar o núcleo da funcionalidade dos objetos usados no cálculo como, por exemplo, imagens, mapas e espectros de dimensões.

As classes básicas do primeiro grupo são:

- *FKTIObj*: Implementa um objeto gráfico básico que o sistema manipula. Os objetos gráficos manipulados devem possuir alguma forma de representação gráfica com o uso de janelas no sistema de janelamento *X11.* Um objeto *FKTIObj* pode servir de entrada para um algoritmo que produzirá como saída um ou mais objetos *FKTIObj* filhos. Assim, cada objeto *FKTIObj* possui, tipicamente, uma lista de objetos *FKTIObj* filhos e um objeto *FKTIObj* pai. Nesta classe existem os seguintes métodos virtuais que devem ser especializados nos objetos reais implementados:
  - *draw*: Deve construir a representação gráfica do objeto no sistema de janelamento. Este método tipicamente constrói uma janela e nela mapeia os dados do objeto;
  - *isSelected*: Retorna True se o objeto foi selecionado pelo usuário como possível entrada para os algoritmos. Caso contrário, retorna False. Deve haver alguma forma de

UNICAMP

selecionar o objeto por meio de sua representação gráfica;

- *select*: Marca o objeto como selecionado. Esse método deve, de alguma forma, evidenciar graficamente que o objeto está selecionado;
- *unselect*: Marca o objeto como não selecionado. Como no método anterior, deve haver alguma representação gráfica para indicar que o objeto não está selecionado;
- *getName*: Este método retorna uma identificador alfanumérico do objeto.

Além desses métodos existem os seguintes métodos que implementam a funcionalidade básica da classe.

- *getParent*: Retorna o objeto pai, ou seja, o objeto que lhe deu origem por meio da execução de algum algoritmo;
- *getProducer*: Retorna o algoritmo que produziu o objeto;
- *getFirstChild*: Inicia um iterador interno e devolve o primeiro objeto filho deste objeto, se existir ao menos um;
- *getNextChild*: Incrementa o iterador interno e devolve o próximo objeto filho deste objeto, se houver ainda objetos filhos;
- *insertChild*: Insere um objeto na lista de objetos filhos deste objeto. A partir desse momento, o objeto inserido pode ser recuperado pela chamada às funções *getFirstChild* e *getNextChild*.

- *FKTIAlg*: Implementa a funcionalidade básica de um algoritmo qualquer que trabalhe com objetos do sistema. Esta classe possui os seguintes métodos virtuais que devem ser redefinidos nas classes derivadas que implementam os algoritmos reais do sistema:
  - *getName* : Deve retornar uma identificação alfanumérica do algoritmo;
  - *run* : Deve executar o algoritmo, escolhendo como entrada objetos dentre os objetos selecionados do sistema e produzindo como saída objetos filhos desses objetos.

  Além desses métodos os seguintes métodos já estão implementados:
  - *insertObj*: Usado para inserir um objeto no sistema. O objeto é tipicamente produzido como execução do algoritmo. Após a chamada deste método, o sistema irá encarregar-se de mapear graficamente o objeto, por meio de uma chamada ao método *draw*;
  - *getFirstSelObj*: Inicia um iterador interno e devolve o primeiro objeto selecionado do sistema, se existir ao menos um. Este método é usado no começo da execução do

algoritmo para escolher os objetos sobre os quais o usuário quer realizar a execução. Uma das versões desse método recebe um parâmetro que identifica a classe do objeto a ser selecionado, o que discutiremos posteriormente;

- *getNextSelObj*: Incrementa o iterator interno e devolve o próximo objeto selecionado do sistema, se ainda existir mais um. Chamada sucessivas a este método após uma única chamada ao método *getFirstSelObj* permite percorrer a lista de objetos selecionados do sistema. Como no método anterior, existe uma sobrecarga desse método com um parâmetro indicando a classe do objeto a ser selecionado. Isso é apenas para comodidade na implementação do algoritmo e será discutido posteriormente.

- *FKTInterface* : Representa o núcleo da interface. Ela é composta basicamente de duas listas, uma dos algoritmos do sistema e outra dos objetos atualmente presentes. São os seguintes os métodos públicos dessa classe:

  - *getNumAlgs* : Retorna o número de algoritmo implementados e inseridos no sistema;
  - *insertAlg* : Insere um novo algoritmo na lista de algoritmos interna;
  - *getFirstAlg* : Inicia um interador e retorna o primeiro algoritmo da classe;
  - *getNextAlg*: Chamadas sucessivas a esse método após uma única chamada ao método anterior retornam, um após um, os algoritmos do sistema;
  - *getFirstObj*: Inicia um iterador e retorna o primeiro objeto do sistema;
  - *getNextObj*: Incrementa o iterador interno e retorna o próximo objeto do sistema. A cada chamada a este método, um novo objeto é retornado até que a lista de objetos esgote-se;
  - *getFirstSelObj*: Como o método *getFirstObj*, apenas retornando o primeiro objeto selecionado se ao menos um existir. Existe uma sobrecarga deste método que requer como parâmetro a classe do objeto a ser retornado, dentre os objetos selecionados;
  - *getNextSelObj*: Executa da mesma forma que o método *getNextObj*, apenas retornando o próximo objeto selecionado do sistema, se ainda exitir ao menos um. Existe uma sobrecarga deste método que requer como parâmetro a classe do objeto a ser retornado, dentre os objetos selecionados.

Resumindo, o núcleo do sistema *Fraktal* é composto de uma classe *FKTInterface* que possui uma coleção de algoritmos e um conjunto de objetos gráficos. Cada objeto gráfico possui um pai (que pode existir ou não) e um conjunto de objetos filhos, produzidos pela execução de algum dos algoritmos que recebeu como entrada o objeto pai. Existe um série de métodos para manipular

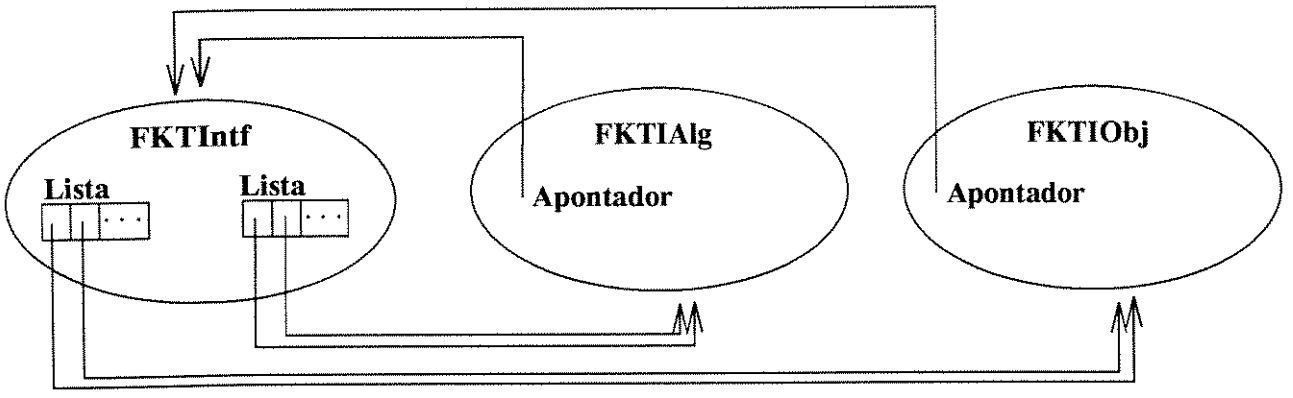

Figura 5.1: Relacionamento entre as classes *FKTInterface*, *FKTIAlg* e *FKTIObj*.

todas essas classes, sendo que alguns deles são virtuais e devem ser usados na implementação de objetos gráficos e algoritmos reais do sistema. A figura 5.1 mostra o relacionamento entre as classes *FKTInterface*, *FKTIAlg* e *FKTIObj*. A classe *FKTInterface* contém uma lista de objetos gráficos *FKTIObj* e uma lista de algoritmos *FKTIAlg*. Cada uma dessas classes possui, por outro lado, um apontador para a classe *FKTInterface* à qual pertence.

A segunda camada de classes foi implementada com o uso da biblioteca de janelamento *XView*. Suas classes sãos derivadas das classes do primeiro grupo e especializam algumas de suas funcionalidades. Temos as seguintes classes implementadas nesse grupo:

- *FKTXVObj* : Esta classe representa a classe básica da qual derivam todas as classes que representam objetos gráfico desse grupo. Ela possui os seguintes métodos:
  - *getBaseFrame* : Este é um método virtual puro e deve devolver o objeto Frame principal do objeto gráfico. Supõe-se que as classes derivadas dessa classe construam um objeto Frame para representação gráfica dos objetos associados;
  - *showFrame* : Este método posiciona o *frame* retornado pelo método anterior no topo da pilha de janelas do servidor *X11*. É usado quando se quer chamar a antenção para algum objeto em especial;
  - *getFirstXVChild* : Chama o método *getFirstChild* da classe *FKTIObj* e realiza sua conversão (*cast*) para um objeto do tipo *FKTXVObj*. Este método é implementado apenas para comodidade de uso;
  - *getNextXVChild* : Chama o método *getNextChild* da classe *FKTIObj* e realiza sua conversão (*cast*) para um objeto do tipo *FKTXVObj*. Este método é implementado apenas para comodidade de uso;

  - *getCircNextXVChild*: Chama repetidamente os métodos *getFirstXVChild* e *getNextXVChild* de forma a retornar circularmente os filhos do objeto atual;
  - *getInterf* : Retorna o objeto *FKTXVInterface* que contém o objeto atual. A classe *FKTXVInterface* é a especialização *XView* da classe *FKTInterface.*

- *FKTXVImage* : Esta classe implementa a representação gráfica de uma imagem topográfica com o uso do sistema *XView.* Possui métodos para obtenção de alguns dos parâmetros da imagem e um método (*getImage*) para devolver o objeto *FKTImage* associado. Esta última classe representa a estrutura de dados para armazenamento de imagens topográficas e possui diversos métodos para atribuição e recuperação de seus parâmetros. Os métodos *draw*, *isSelected*, *select* e *unselect* definidos na classe *FKTIObj* são finalmente implementados;
- *FKTXVRealMap*: Implementa a representação gráfica de um mapa bibimensional de números reais, os quais podem ser os valores de rugosidade dos *pixels* de uma imagem topográfica. Possui o método *getMap* que retorna um objeto *FKTRealMap* que representa toda a funcionalidade de um mapa de números reais. Redefine os métodos *draw*, *isSelected*, *select* e *unselect* herdados da classe *FKTIObj*;
- *FKTXVProfile* : Implementa a representação gráfica com o uso de *XView* de um conjunto de pares de números reais, constituindo um gráfico que mapeia o primeiro elemento no segundo elemento de cada par. Implementa os métodos *draw*, *isSelected*, *select* e *unselect* herdados da classe *FKTIObj*;
- *FKTXVParms* : Essa classe é a implementação gráfica de um mecanismo de obtenção de parâmetros de execução para os algoritmos do sistema. O algoritmo que deseja obter certos tipos de parâmetros do usuário precisa apenas instanciar um objeto dessa classe, especificar os tipos de parâmetros que deseja e chamar o método *getParms.* Com isso, será aberta uma janela com entradas para os parâmetros especificados. Após a seleção por parte do usuário, esses parâmetros poderão ser usados para a execução do algoritmo. Maiores detalhes sobre este mecanismo serão fornecidos posteriormente na discussão da classe *FKTParms.* Implementa os métodos *draw*, *isSelected*, *select* e *unselect* herdados da classe *FKTIObj*;
- *FKTXVAlg* : Esta classe é derivada da classe FKTIAlg e redefine o método virtual *run.* Define também o método virtual *exec*, que é usado em conjunto com a classe *FKTXVParms.* Maiores detalhes o uso deste método serão discutidos posteriormente;
- *FKTXVInterface* : Esta classe é derivada da classe *FKTInterface.* Possui um método *init* para iniciação do ambiente gráfico do sistema e redefine o método *run* da classe *FKTInterface.* Esse método representa o laço principal de execução do sistema e acaba desembocando

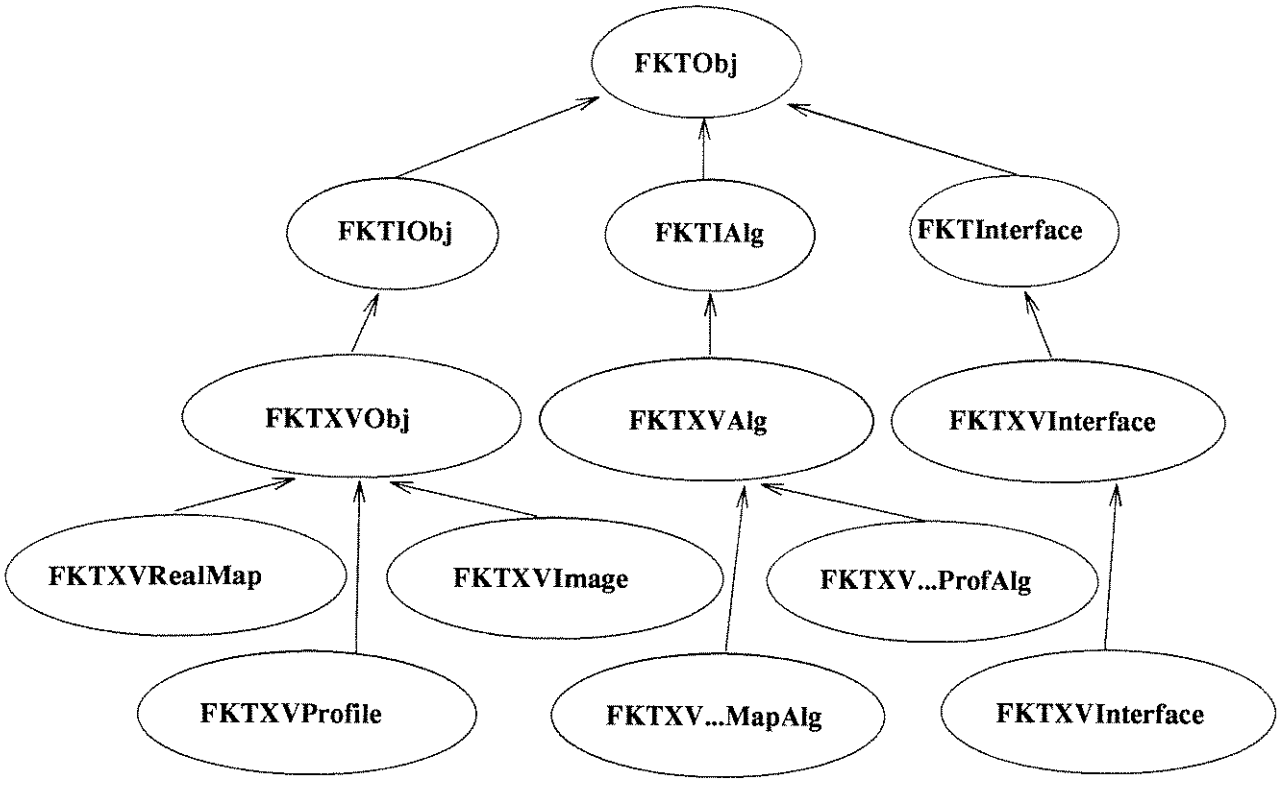

Figura 5.2: Hierarquia de classes parcial do sistema Fraktal.

numa chamada para a função *XView xv_main_loop*. Possui vários métodos para recuperar os atributos específicos ao *XView* do sistema;

- *FKTXVVarNeighMapAlg* : Esta classe implementa o interfaceamento gráfico com o sistema para cálculo do mapa de dimensões discutido na seção 4. A funcionalidade dessa classe resume-se em selecionar um objeto FKTXVImage como entrada, usar a classe *FKTVarGridSmallAlg* para calcular um mapa de dimensões *FKTRealMap*, construir um objeto *FKTXVRealMap* com o uso desse mapa e inserí-lo da lista de objetos do sistema. O objeto pai do objeto *FKTXVRealMap* criado é a imagem FKTXVImage selecionada como entrada. A classe FKTXVParms é usada nessa execução para receber parâmetros do usuário;
- *FKTXVVarNeighProfAlg* : Funciona da mesma forma que a classe anterior, apenas selecionando um objeto FKTXVImage como entrada e produzindo um objeto *FKTXVProfile*.

Além dessas classes existem várias outras, tais como:

- *FKTImage* : Implementa a manipulação de uma imagem topográfica, contendo métodos

para escrita em arquivo ou leitura de arquivo e para recuperação ou atribuição de inúmeros parâmetros;

- *FKTRealMap* : Implementa a manipulação de um mapa bidimensional de números reais, contendo métodos para escrita em arquivo ou leitura de arquivo e para recuperação ou atribuição de inúmeros parâmetros;
- *FKTDimProfile* : Implementa um conjunto de pares de números reais que representa os pontos de um gráfico em $\mathbf{R}^2$;
- *FKTNano* : Implementa métodos para leitura e escrita de imagens geradas pelo sistema *Nanoscope II*;
- *FKTVarGridSmallAlg* : Implementa o algoritmo que calcula um mapa de dimensões ou um espectro de dimensões para uma dada imagem topográfica;
- *FKTList* : Implementa uma lista ligada genérica;
- *FKTString* : Implementa o conceito de cadeia de caracteres.

Observe que toda a funcionalidade estrutural do sistema foi implementada com as classes *FKTObj*, *FKTIObj*, *FKTIAlg FKTInterface*. A parte da funcionalidade dependende da bibliteca de janelamento, no caso *XView*, foi implementada através das classes do segundo grupo, ou seja, das classes *FKTXVObj*, *FKTXVAlg* e *FKTXVInterface*. Na ponta do processo de derivação encontram-se as classes *FKTXVRealMap*, *FKTXVProfile* e *FKTXVImage* que implementam os objetos manipulados pelo sistema. Encontram-se também as classes *FKTXVVarNeighMapAlg* e *FKTXVVarNeighProfAlg* que representam os algoritmos implementados no sistema e a classe *FKTXVInterface* que implementa a parte gráfica da arquitetura do sistema.

Novos tipos de objetos podem ser implementados por meio da derivação da classe *FKTXVObj* e através da implementação dos métodos virtuais *draw*, *isSelected*, *select* e *unselect*. A partir desse momento, esse novo objeto poderá ser manipulado automaticamente pelo sistema.

Da mesma forma, novos algoritmos podem ser facilmente implementados por meio da derivação da classe FKTXVAlg e da redefinição dos métodos virtuais *getName*, *run* e *exec*. A partir desse momento, o algoritmo pode ser inserido no sistema por meio do uso do método *insertAlg*, tornando-se disponível para execução por parte do usuário.

5.1 A Implementação de Novos Algoritmos

Já foi mencionado que os algoritmos implementados sob o sistema Fraktal devem escolher os objetos gráficos (imagens topográficas, por exemplo) sobre os quais atuam com o uso dos métodos *getFirstSelObj* e *getNextSelObj* herdados da classe *FKTIAlg*. Como todos os objetos inseridos no sistema são armanenados como ponteiros para a classe básica *FKTIObj*, deve-se, no momento da seleção, identificar os tipos dos objetos selecionados, escolhendo apenas os que interessam ao algoritmo. Uma maneira clássica de resolver este problema é definir um conjunto de constantes globais, cada uma identificando um objeto que o sistema suporta e, no momento da criação do objeto, atribuir-lhe uma dessas constantes que identificará seu tipo. Dessa forma, no momento da seleção, basta comparar o tipo armazenado do objeto com as constantes apropriadas para identificar os objetos de interesse. Entretanto, esta abordagem tem uma falha que é a necessidade de manter sempre atualizadas as definições dessa constantes de identificação globais e, dessa forma, a inserção de um tipo novo de objeto no sistema irá requerer a criação de um novo tipo de constante.

Como cada objeto do sistema no fundo corresponde a uma classe C++, poderíamos realizar a identificação dos objetos desejados para cada algoritmo se pudéssemos identificar a classe à qual um dado objeto pertence. Ou seja, suponhamos que temos um objeto X da classe *ClasseX* recebido como um ponteiro para a classe mãe *FKTIObj*. Se pudéssemos, em C++, identificar em tempo de execução que o objeto X pertence à classe *ClasseX*, então poderíamos selecionar facilmente os objetos de interesse. Entretanto, não existe mecanismo implícito em C++ para realizar esta identificação.

Para implementar um mecanismo que permita identifcar dessa forma as classes às quais os objetos pertencem, podemos usar de dois recursos disponíveis em C++ : os métodos virtuais e os membros estáticos. Suponhamos que para cada classe nova que representa um objeto gráfico a ser manipulado pelo sistema declaremos os seguintes métodos e membros:

```
classe NovoTipoDeObjeto : public FKTXVObj
{
    static  long getClassType()   {return fktClassType;}
    virtual long getObjectType()  {return fktClassType;}
  private:
    static long fktClassType;
};
```

Ou seja, definimos um membro estático *fktClassType* que, no fundo, irá identificar a classe declarada. A iniciação desse membro poderá ser feita automaticamente, no momento da definição dos membros estáticos da classe, da seguinte forma:

```
long  NovoTipoDeObjeto::fktClassType = FKTClassTypeMaxID+=1;
```

A variável *FKTClassTypeMaxID* é global a todo o sistema e é feita ser inicialmente igual a 0. Assim, a cada introdução de uma classe nova para um novo objeto gráfico, o membro estático desta classe irá ganhar um identificador novo.

No início da execução do algoritmo no sistema, este receberá ponteiros para a classe *FKTIObj*. Podemos, então, identificar o tipo do objeto comparando o valor retornado pelo método *getObjectType* com o valor retornado pela chamada a *TipoDesejado::getClassType*, onde *TipoDesejado* é o nome da classe cujos objetos queremos selecionar. A chamada ao método virtual *getObjectType* irá retornar o tipo da classe do objeto real recebido através do ponteiro para a classe *FKTIObj*. A chamada a *TipoDesejado::getClassType* irá retornar o tipo da classe *TipoDesejado*, ou seja, o valor de seu membro estático *fktClassType*. Se esses valores coincidirem, então o objeto recebido é da classe *TipoDesejado*.

Assim, no começo da execução do algoritmo, ele deverá possuir um pedaço de código parecido com o código abaixo para selecionar objetos de um determinado tipo. Suponhamos que o algoritmo deseja selecionar objetos do tipo *FKTXVImage*.

```
FKTIObj * obj = getFirstSelObj();
while (obj != 0)
{
  if (obj->getObjectType() == FKTXVImage::getClassType())
    break;
  obj = getNextSelObj();
}
if (obj != 0)
{
  FKTXVImage * image = (FKTXVImage*) obj; // Cast para o tipo desejado
  // Execute aqui o algoritmo
}
```

O algoritmo poderá também usar o método *getFirstSelObj* como abaixo, o qual já traz implícita a comparação de tipo acima.

```
FKTXVImage * image;

image = (FKTXVImage*)getFirstSelObj(FKTXVImage::getClassType());

// Supondo que qualquer objeto deste tipo sirva ...
if (image != 0)
{
  // Execute aqui o algoritmo
}
```

Para facilitar esse trabalho de implementação de novos objetos gráficos, o sistema define duas macros, como abaixo:

```
#define FKT_CLASSTYPE_DECLARE(name)                    \
       public:                                         \
         static  long getClassType()                   \
             {return fktClassType;}                    \
         virtual long getObjectType()                  \
             {return fktClassType;}                    \
       private:                                        \
         static long fktClassType

#define FKT_CLASSTYPE_DEFINE(name)                     \
       long  name::fktClassType = FKTClassTypeMaxID+=1
```

Ambas as macros recebem como parâmetro o nome da classe que se quer definir. A primeira deve ser usada no momento de declaração da classe, logo antes de fechar a declaração com o caracter } final, como abaixo:

```
classe NovoTipoDeObjeto : public FKTXVObj
{

  FKT_CLASSTYPE_DECLARE(NovoTipoDeObjeto);
};
```

Assim, automaticamente será criado o código para definição do tipo da classe. A segunda macro deve ser usada no momento de definição da classe, tipicamente no arquivo de implementação (arquivo com extensão .C ou .CPP), como abaixo:

```
FKT_CLASSTYPE_DEFINE(NovoTipoDeObjeto);
```

Dessa forma, automaticamente será criado o código para definição do membro estático *NovoTipoDeObjeto::fktClassType*. Existem macros parecidas para realizar essas funções quando a classe implementada é um *template*.

Vejamos agora como um algoritmo sob o sistema Fraktal pode receber parâmetros do usuário. Para isso existe a classe *FKTXVParms*, a qual possui várias sobrecargas do método *insertParm* para que o programador selecione os tipos de parâmetros que deseja receber e o método *getParms* que abrirá uma janela para realizar o trabalho final de recepção. Suponhamos, para efeito de ilustração, que o algoritmo precise receber dois parâmetros:

- *minEscala* e *maxEscala* : Valores mínimo e máximo das escalas para cálculo do espectro de irregularidades, dentro da faixa de valores possíveis de 0 a 100. Os valores selecionados devem estar dentro dessa faixa, sendo espaçados pelo valor *passoEscala*.
- *metodo* : Método de execução, de um conjunto de três métodos possíveis *METODO1*, *METODO2* e *METODO3*.

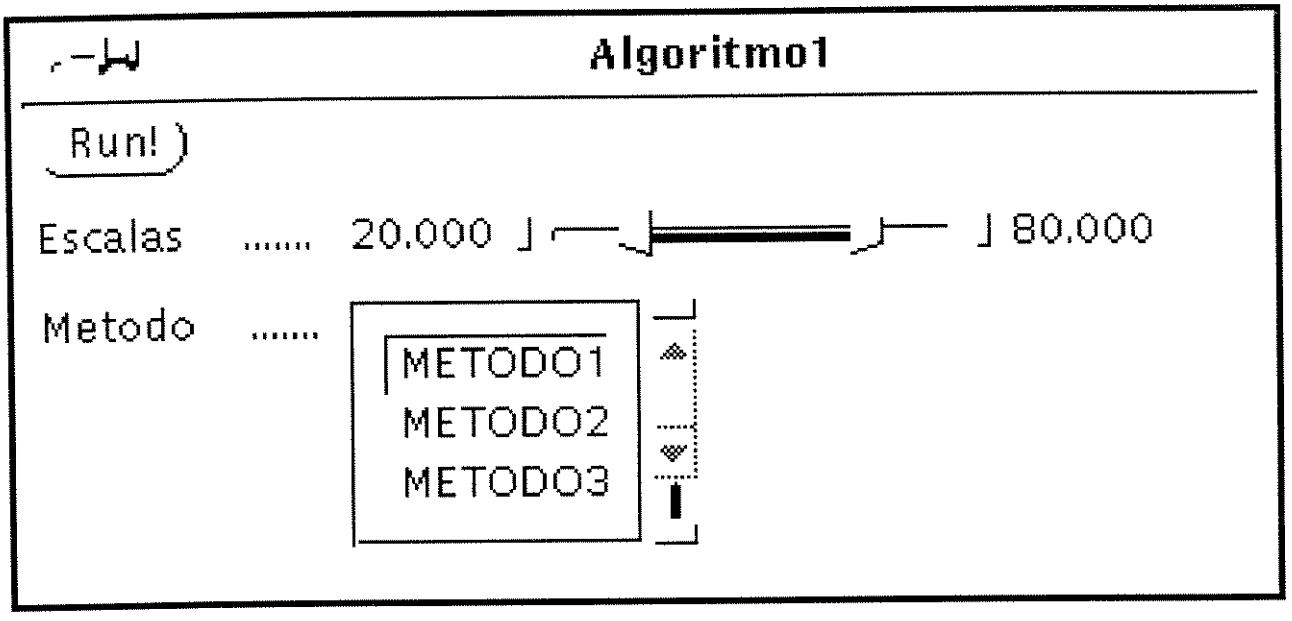

Figura 5.3: A janela aberta pela classe *FKTXVParms*.

Para isso, o algoritmo precisaria, no começo do método *run*, escrever o seguinte código:

```
if (parms != 0)
  delete parms;

parms = new FKTXVParms(getXVInterf(),this);
assert(parms != 0);

valPossiveis.append(FKTString("METODO1"));
valPossiveis.append(FKTString("METODO2"));
valPossiveis.append(FKTString("METODO3"));

parms->insertParm(FKTString("Escalas"),minEscala,maxEscala,
                                        0,100,
                                        passoEscala);

parms->insertParm(FKTString("Metodo"),
                  valPossiveis,valEscolhidos,True);

return parms->getParms();
```

Neste código estamos supondo que as variáveis *parms*, *valPossiveis* e *valEscolhidos* são membros da classe que implementa o algoritmo, os dois últimos sendo do tipo *FKTList<FKTString>*, ou seja, lista de cadeias de caracteres, de acordo com os objetos básicos *FKTList* e *FKTString* definidos no sistema Fraktal.

Quando chamamos os métodos *insertParm*, inserimos os valores possíveis que o usuário pode escolher bem como as variáveis que receberão os valores escolhidos pelo usuário. Na primeira chamada, estamos dizendo que o usuário deve escolher um faixa de números reais entre os valores 0 e 100, intervalados por um espaçamento de *passoEscala* e que o valor mínimo escolhido deve ser armazendo em *minEscala* e o valor máximo escolhido deve ser armazenado em *maxEscala*. Na segunda chamada, estamos dizendo que a lista de valores possíveis de escolha é *valPossiveis* e que os valores escolhidos devem ser armazenados na lista *valEscolhidos*. O terceiro parâmetro é feito igual a *True*, indicando que a lista de escolha é exclusiva, ou seja, somente um valor pode ser escolhido. A classe *FKTXVParms* possui métodos *insertParm* para vários tipos de parâmetros, entre eles, número inteiros, faixas de números inteiros, números reais, valores *booleanos* e cadeias de caracteres.

No momento da chamada ao método *getParms*, a janela mostrada na figura 5.3 é criada e o método retorna. A partir desse momento o usuário pode escolher os valores que deseja. Na figura, ele escolheu a faixa *minEscala* = 20 e *maxEscala* = 80 e o método *METODO1*. Quando ele estiver satisfeito, basta pressionar o botão *Run!* da janela. Nesse momento, a classe *FKTXVParms* irá encarregar-se de chamar o método *exec* do algoritmo. Se as variáveis *minEscala*, *maxEscala* e *valEscolhidos* são membros da classe que implementa o algoritmo, então, os valores escolhidos poderão ser acessados. Observe que o cabeçalho da janela na figura 5.3 é *Algoritmo1*. Esse nome é recuperado a partir de uma chamada ao método *getName* do algoritmo.

Em resumo, cada novo algoritmo implementado no sistema precisa redefinir os métodos *getName*, *run* e *exec*. Se ele quiser usar a classe *FKTXVParms*, poderá fazer isso dentro do método *run*, o qual é chamado quando o usuário seleciona, por meio do sistema, o algoritmo para executar. Nesse momento, o algoritmo pode também selecionar o objeto gráfico sobre o qual irá atuar com o uso dos métodos *getFirstSelObj* e *getNextSelObj*. Isso, aliás, é feito com freqüência, uma vez que a definição dos parâmetros a serem recebidos do usuário depende do objeto selecionado. Por fim, o trabalho real do algoritmo é executado no método *exec*, o qual é chamado automaticamente pela classe *FKTXVParms*. Se o algoritmo não precisar de parâmetros do usuário, ele pode executar todo o trabalho dentro do método *run*, sem precisar implementar o método *exec*. Em todo caso, como resultado da execução, o algoritmo irá produzir um ou mais objetos gráficos, os quais devem ser

inseridos no sistema com o uso do método *insertObj*. Suponhamos, para ilustração, que o algoritmo recebeu um objeto *imagem1*, um ponteiro para o tipo *FKTXVImage*, e produziu um objeto *mapa1*, um ponteiro para o tipo *FKTXVRealMap*. O código para fazer a inserção desse novo objeto é o seguinte (O objeto *imagem1* será feito pai do objeto *mapa1*):

```
insertObj(imagem1,mapa1);
```

5.2 A Implementação de Novos Objetos Gráficos

Para implementar um novo tipo de objeto gráfico no sistema Fraktal, será preciso derivar a classe a ser implementada do classe *FKTXVObj* e definir os seguintes métodos virtuais:

- *getName* : Este método deve devolver um objeto *FKTString* contendo uma descrição alfa-numérica do nome do objeto criado;
- *getBaseFrame* : Com o uso da biblioteca de desenvolvimento *XView*, cada objeto gráfico no sistema Fraktal fatalmente irá possuir um objeto *Frame* e vários outros objetos *XView*. Este método deve devolver o *Frame* principal do objeto criado;
- *select* : Este método simplemente deve marcar um membro como *True*, indicando que o objeto foi selecionado pela interface;
- *unselect* : Este método deve marcar o membro citado no item anterior como *False*, indicando que o objeto não está mais selecionado;
- *isSelected* : Este método deve devolver *True* se o objeto está selecionado (por meio do valor do membro citado nos dois itens anteriores) ou *False* se o objeto não está selecionado;
- *draw* : Este é o método mais trabalhoso, uma vez que envolve a criação de vários objetos *XView* para mostrar graficamente o objeto. Não entraremos em maiores detalhes, uma vez que a implementação deste método exige um conhecimento razoalvelmente profundo da biblioetca *XView*. Um rápida inspeção do código fornecido com o sistema Fraktal vale muito mais que uma descrição longa.

Implementar novos algoritmos que trabalham com os objetos pré-definidos no sistema é muito fácil, conforme discutimos na seção anterior. A implementação de novos objetos gráficos, por outro lado, envolve um conhecimento detalhado não só do sistema *XView*, bem como da estrutura do sistema Fraktal. Isso vem indicar que o sistema Fraktal foi basicamente elaborado para implementação e testes de novos algoritmos que trabalham com os objetos pré-definidos.

5.3 Usos e Limitações do Sistema Fraktal

Para o leigo em computação, o sistema Fraktal poder ser usado como ferramenta de análise de imagens topográficas. Através de sua interface gráfica, o usuário encontrará maneiras de ler imagens geradas pelo sistema *Nanoscope II* ou com um formato próprio, gravar imagens em arquivo, calcular e visualizar graficamente o mapa de rugosidades de imagens, calcular e visualizar graficamente o espectro de rugosidades, entre algumas outras funcionalidades.

Para o desenvolvedor de algoritmos, o sistema Fraktal pode ser usado em dois níveis. No primeiro deles, pode-se implementar facilmente novos algoritmos que trabalham com os objetos gráficos pré-implementados, conforme descrito na seção 5.1. Com isso, todo o mecanismo de visualização dos objetos recebidos como entrada ou produzidos pelo algoritmo, bem como o mecanismo de recepção de parâmetros do usuário é provido automaticamente pelo sistema, tornando a implementação e teste muito mais fáceis para o programador. No segundo nível, encontramos o desenvolvedor que quer usar toda a funcionalidade provida pelas classes que não dependem do sistema *XView*, ou porque ele quer usar outra biblioteca de janelamento ou porque ele não quer usar mecanismos de visualização gráfica. Nesse caso, todas as outras classes implementadas no sistema podem ser muito úteis, provendo funcionalidade para manipular estruturas de dados que modelam imagens topográficas, mapas de números reais, entre vários outros objetos. Além disso, nesse grupo encontram-se as classes que implementam o núcleo da funcionalidade do algoritmo para estimativa da rugosidade de imagens que descrevemos. O código que implementa o sistema Fraktal foi escrito de tal forma que as partes independentes da biblioteca *XView* encontram-se totalmente isoladas em arquivos separados, permitindo que o implementador apenas use os arquivos de interesse.

Atualmente, após implementar um novo algoritmo, o programador deverá inserir no corpo da função *main* do sistema uma chamada ao métdodo *insertAlg* da classe *FKTXVInterface*, o que irá fazer com que o algoritmo torne-se visível nos menus de seleção disponíveis para o usuário. Ou seja, o sistema *Fraktal* deve ser recompilado a cada inserção de um novo algoritmo, o que o torna inapto para uma distribuição fechada, com fornecimento apenas do código executável. Para que isso fosse possível sem retirar do programador a possibilidade de desenvolver novos algoritmos, deveria ser usado o mecanismo de ligação dinâmica. Com isso, o desenvolvedor poderia gerar o código objeto com o novo algoritmo implementado, o qual seria ligado em tempo de execução ao sistema, eliminando a necessidade de recompilação.

Essa é uma das principais limitações do sistema, além da óbvia dificulade de implementação de novos objetos gráficos. Entretanto, o sistema Fraktal foi elaborado para desenvolvi-

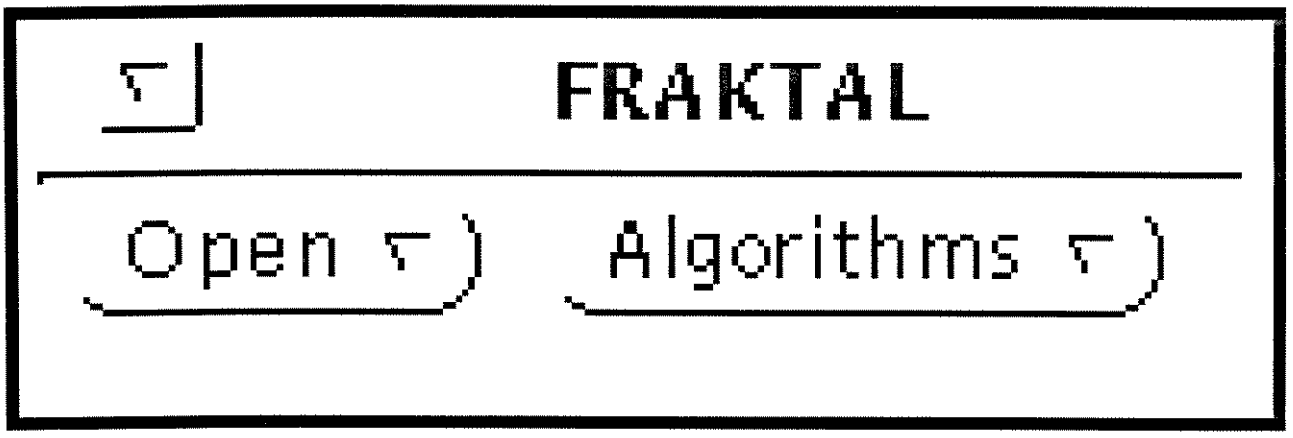

Figura 5.4: Janela principal do sistema Fraktal.

mento e testes de novos algoritmos que trabalham com imagens topográficas, em especial algoritmos similares ao apresentado neste trabalho. Além disso, o código de todo o sistema é totalmente livre e, portanto, a necesssidade de prover ligação dinâmica para impedir sua distribuição é bastante supérflua. Assim, para as finalidades de teste do algoritmo apresentado e para desenvolvimento de algoritmos similares, o sistema Fraktal apresenta-se muito adequado, permitindo grande facilidade de implementação.

5.4 Como Usar o Sistema Fraktal

Nesta seção iremos descrever como usar o sistema Fraktal para realização de análises de imagens topográficas.

A janela principal do sistema é mostrada na figura 5.4. Nela existem dois *menus* : *Open* e *Algorithms*. O primeiro deles permite abrir e visualizar arquivos que armazenam imagens topográficas em dois formatos, o formato gerado pelo *Nanoscope II* e o formato padrão entendido pelo sistema Fraktal. O segundo *menu* permite selecionar os algoritmos que irão executar sobre os objetos gráficos selecionados como, por exemplo, as imagens lidas.

Ao abrirmos o *menu Open* e selecionarmos a entrada referente ao formato *Nanoscope II*, é apresentada a janela mostrada na figura 5.5, a qual permite-nos digitar o nome do arquivo de entrada. Após a digitação, basta pressionar o botão *Read!* para que a imagem seja lida e mostrada como na figura 5.6. O cabeçalho da janela de visualização contém o nome da imagem, conforme armazenado no arquivo de entrada. O algoritmo de visualização realiza um mapeamento dos

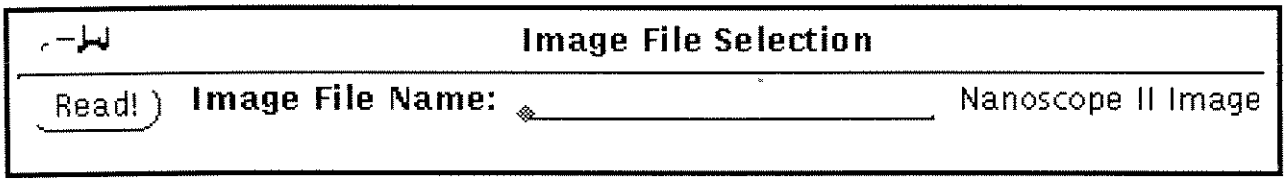

Figura 5.5: A entrada do nome de um arquivo de imagem.

valores da imagem para níveis de cinza, de forma que os *pixels* com valores menores são mapeados para valores próximos de 0 e *pixels* com valores maiores são mapeados para valores próximos de 255. O mapeamento realizado depende da opção *Normalize/Detail Emphasis* discutida no próximo parágrafo.

Se pressionarmos o botão direito do *mouse* dentro do *canvas* desta janela, será mostrado um *frame* com as várias funcionalidades disponíveis para manipulação de imagens, como ilustrado na figura 5.7. As entradas *Column* e *Row* mostram e permitem selecionar o *pixel* cujo valor é mostrado no campo *Value*. Neste caso, temos selecionado o *pixel* da coluna 73 e linha 104, cujo valor aproximado é 131.49 nanômetros. O *pixel* a ser mostrado pode ser também selecionado apertando-se o botão esquerdo do *mouse* no ponto apropriado da imagem. Nesse caso, os campos *Column*, *Row* e *Value* são automaticamente atribuídos. O campo *Distance* mostra a distância horizontal entre dois *pixels*, selecionados da seguinte forma: pressione o botão *Shift*, pressione o botão esquerdo do *mouse* no primeiro *pixel* e, em seguida, pressione o botão esquerdo do *mouse* no segundo *pixel*. Os campos *Horizontal Scale* e *Vertical Scale* mostram a largura e altura da imagem na unidade de medida utilizada. Dentre as operações disponíveis, temos o botão *Father* que seleciona a janela que mostra o objeto *pai* da imagem, o botão *Next Son* que, sempre que pressionado, mostra um dos objetos filhos da imagem em seqüência, caso haja ao menos um. O botão *Unselect All* marca todos os objetos do sistema como não selecionados, fazendo com que chamadas ao método *getFirstSelObj* não devolva nenhum objeto. A entrada *Selected* permite que o objeto seja marcado como selecionado (opção *Yes*) ou não selecionado (opção *No*). Finalmente, temos a operação *Normalize/Detail Emphasis* que, quando selecionada (opção *Yes*), realiza uma quantização estatística dos valores dos *pixels* da imagem, de forma que mais níveis de cinza são atribuídos aos *pixels* cujos valores são mais freqüentes. Essa opção será detalhada posteriormente.

Vejamos como executar um algoritmo, por exemplo o algoritmo que calcula o mapa de irregularidades, sobre uma imagem. Na janela mostrada na figura 5.7, presione o botão *Unselect All* e, em seguida, selecione a opção *Yes* da entrada *Selected*. Isso faz com que a imagem correspondente à janela seja o único objeto selecionado. Em seguida, selecione a entrada *Variance-Small Grid Map*

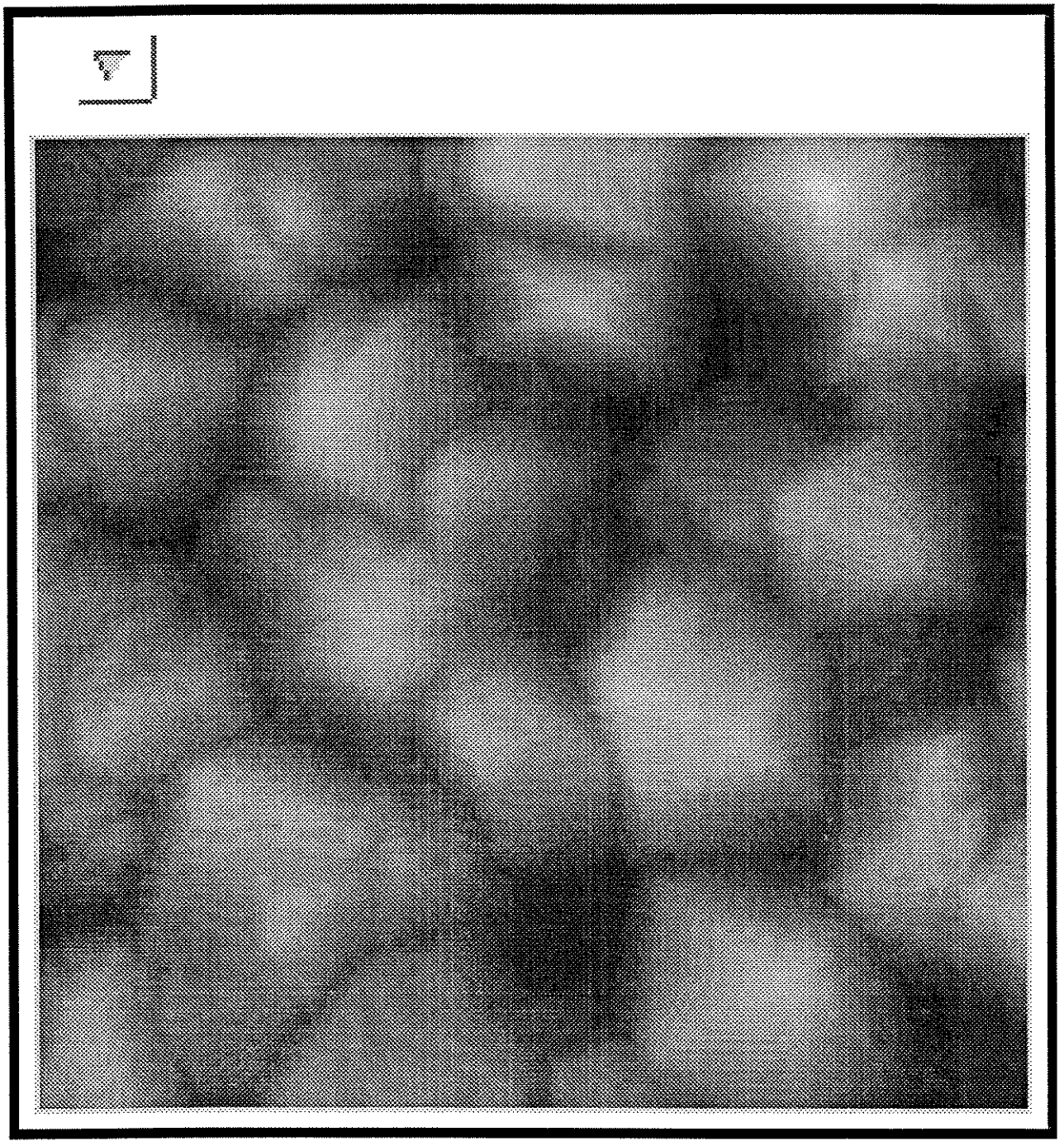

Figura 5.6: A janela de visualização de uma imagem.

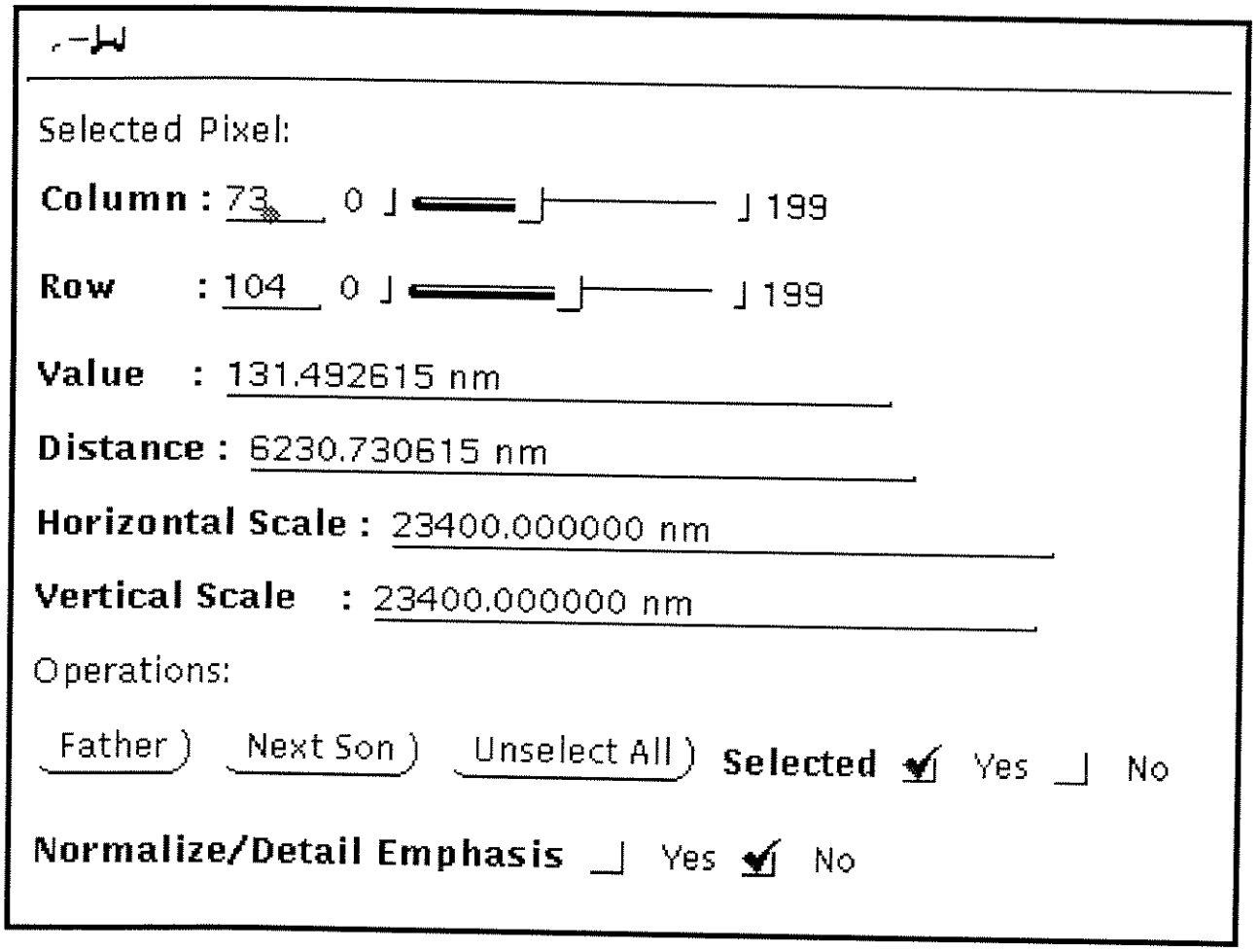

Figura 5.7: A janela de operações para imagens.

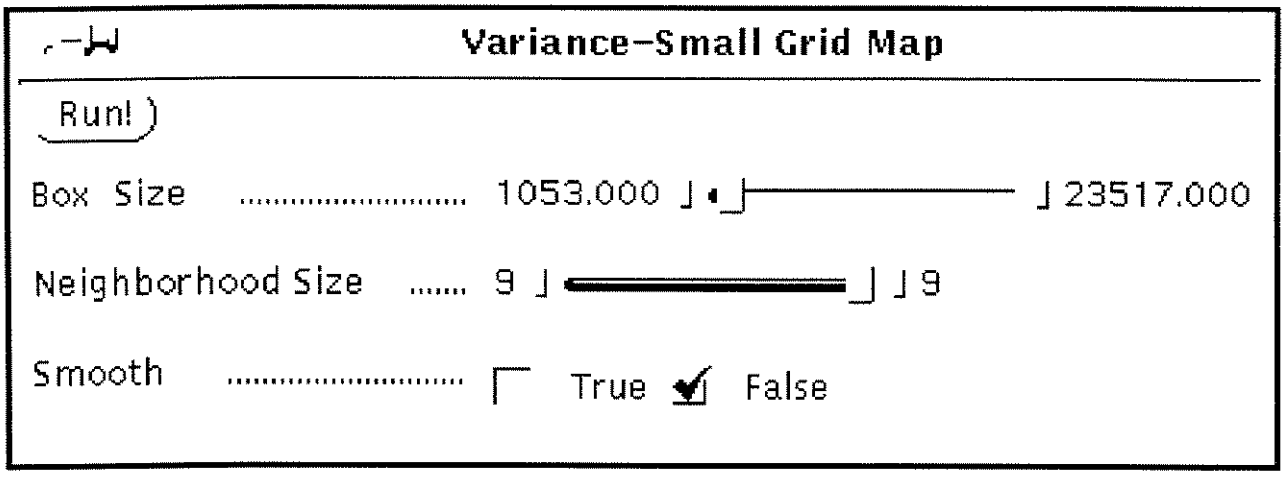

Figura 5.8: A janela de execução do algoritmo que calcula o mapa de irregularidades de uma imagem.

do *menu* em Algorithms. Será mostrada a janela da figura 5.8. A entrada *Box Size* é o lado do quadrado de segmentação da imagem usado na execução do algoritmo. O valor máximo possível é mostrado no lado direito do campo e corresponde ao valor máximo dentre a largura e altura da imagem. A unidade de medida usada é a unidade mostrada na janela de operações sobre a imagem (figura 5.7). A entrada *Neighborhood Size* permite selecionar o valor de n na equação 4.3 (O valor de f usado na implementação corrente é 0.1). O campo *Smooth* permite selecionar se o mapa de irregularidades gerado será suavisado após a execução do algoritmo. O mapa de dimensões originalmente gerado pelo algoritmo contém o valor de irregularidade calculado para cada um dos *pixels* da imagem. Se colocarmos a opção *Smooth* como *Yes*, após a geração do mapa de dimensões original, um novo mapa será calculado de forma que o valor de cada um de seus *pontos* será igual à média aritmética dos pontos do mapa original dentro de um quadrado de lado igual ao valor selecionado em *Box Size*. Isso pode ser útil, uma vez que padrões em formatos de faixas verticais e horizontais com espaçamento igual ao tamanho do quadrado de segmentação usado no algoritmo costumam aparecer no mapa de irregularidades originalmente gerado. A suavização dos valores desse mapa por um algoritmo de média dentro de um quadrado de mesmo lado tende a minimizar esses padrões, sob a penalidade de alterar os valores de irregularidade originalmente calculados. Entretanto, isso pode não ser um problema, quando o algoritmo é usado para identificar as regiões da imagem de maior ou menor irregularidade.

Como resultado da execução do algoritmo, teremos um mapa de irregularidades mostrado na figura 5.9. Assim como na visualização das imagens, nos mapas quanto maior o valor da irregularidade de um ponto, maior é o nível de cinza. Deste modo, os pontos com maior irre-

gularidade aparecem mais claros. Se pressionarmos o botão direito do *mouse* dentro dessa janela, obteremos a janela da figura 5.10. Os campos *Column* e *Row* permitem selecionar a coluna e a linha do ponto do mapa cujo valor calculado da irregularidade será mostrado no campo *Value*. O ponto pode também ser selecionado pressionando-se o botão esquerdo do *mouse* dentro da janela de visualização do mapa, mostrada na figura 5.9.

Vejamos agora como calcular o espectro de irregularidades para um determinado *pixel* da imagem. Para isso, selecione no *menu Algorithms* a entrada *Variance-Small Grid Profile*, o que mostrará a janela da figura 5.11. O campo *Neighborhood Size* é o valor de n na seqüência 4.3, para $f = 0.1$. O campos *Scale Range* permite selecionar a faixa de escalas para as quais serão calculados os valores de irregularidade do *pixel* da imagem. A coluna e a linha do *pixel* da imagem são selecionadas pelos campos *Column* e *Row*. Selecionados todos os valores, basta pressionar o botão *Run!* para executar o algoritmo sobre a imagem selecionada. Como resultado da execução será produzido um gráfico mostrado na figura 5.12, cujo eixo vertical representa os valores de irregularidades e cujo eixo horizontal representa os diversos valores de escalas. Se o usuário pressiona o botão direito do *mouse* sobre esta janela, será mostrada a correspondente janela de operações, mostrada na figura 5.13. Esta janela mostra nos campos *Column* e *Row* os valores da coluna e linha do *pixel* para o qual foi calculado o espectro de irregularidades. O campo *Scale* permite selecionar a escala cujo valor de irregularidade calculado será mostrado no campo *Dimension*. O usuário pode também selecionar outra escala apenas pressionando o botão esquerdo do *mouse* sobre um dos pontos do gráfico na janela da figura 5.12. Automaticamente serão atualizados os campos *Scale* e *Dimension*. Assim como nas janelas de operações sobre imagens e mapas de irregularidades, existem os botões *Father*, Next Son e Unselect All, além do campo *Selected*, permitindo mostrar o objeto pai desse objeto, mostrar seqüencialmente os objetos filhos, marcar todos os objetos como não selecionados e selecionar o objeto em si.

5.5 A Opção Normalize/Detail Emphasis

Essa opção, disponível na janela de operações de imagens, permite que os valores dos *pixels* sejam quantizados estatisticamente em 256 níveis de cinza. Nessa quantização estatística, os valores dos *pixels* da imagem são divididos em 255 faixas de igual tamanho entre os valores mínimo e máximo atingidos por eles. Em seguida, são calculadas as quantidades de *pixels* cujos valores ficam em cada uma das faixas. A cada uma das faixas é atribuída, então, um número de níveis de cinza proporcional a essas quantidades, fazendo com que as faixas com maior número de valores de

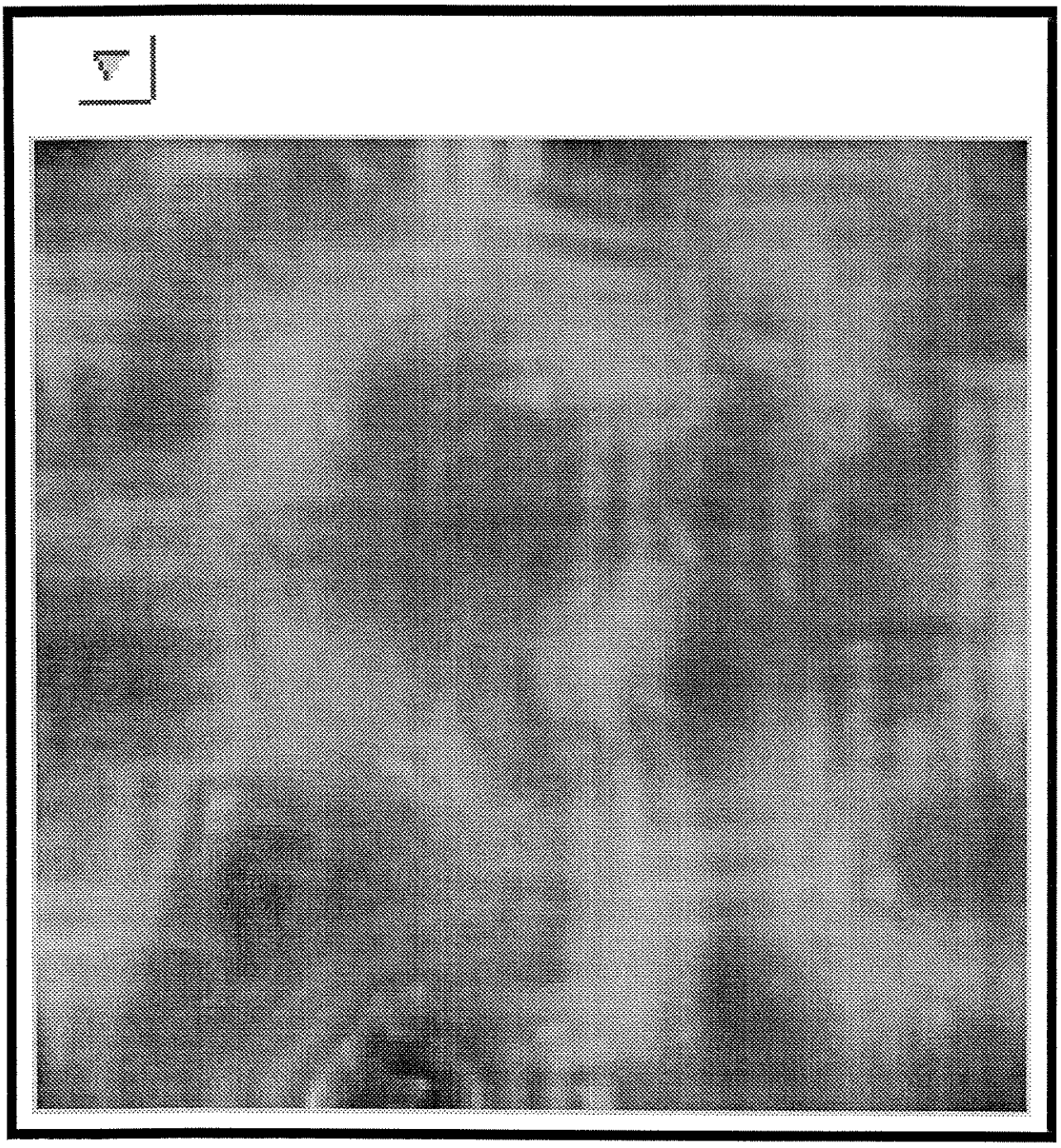

Figura 5.9: Um mapa de irregularidades.

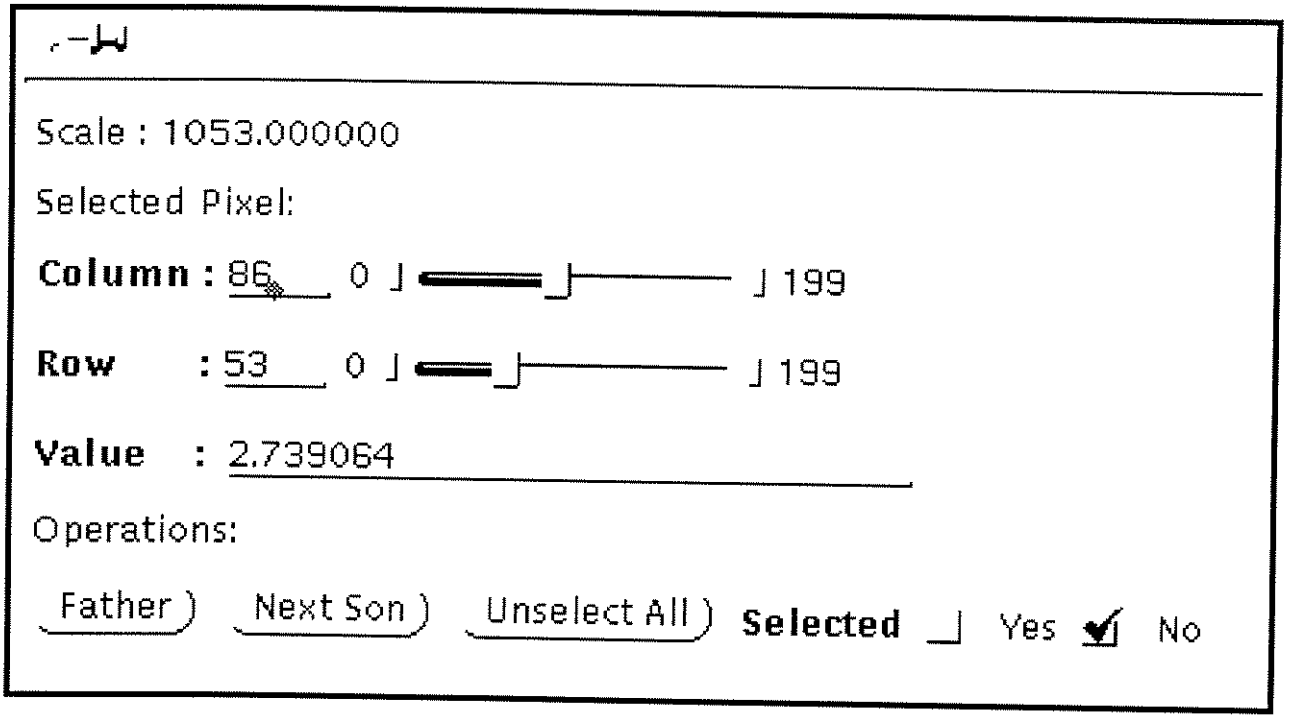

Figura 5.10: A janela de operações de um mapa de irregularidades.

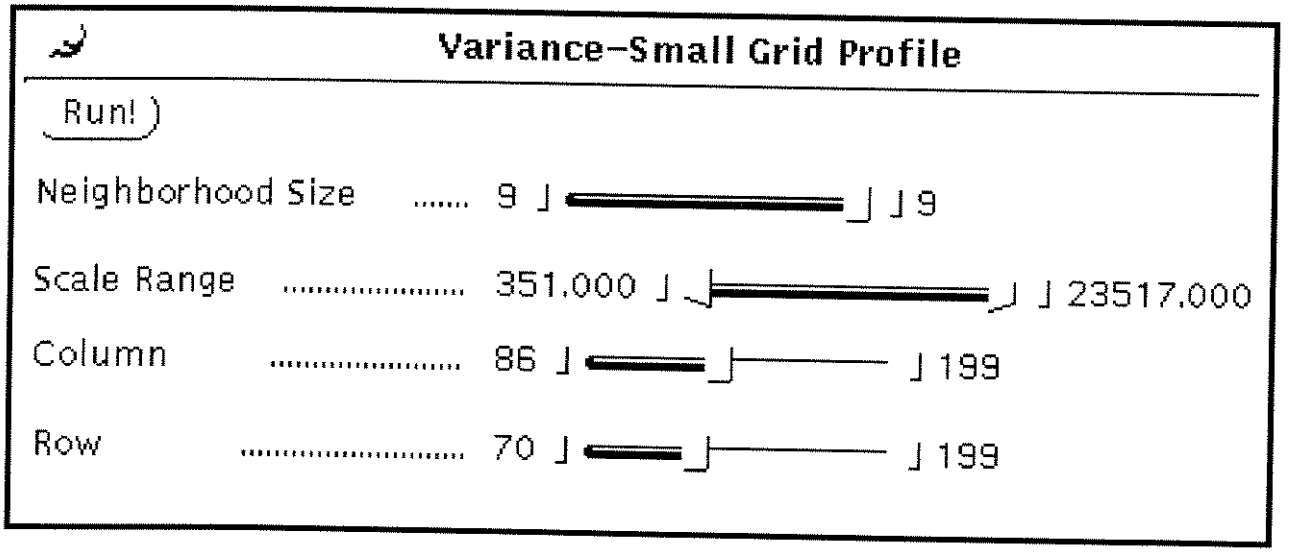

Figura 5.11: A janela de execução do algoritmo que calcula o espectro de irregularidades de um ponto da imagem.

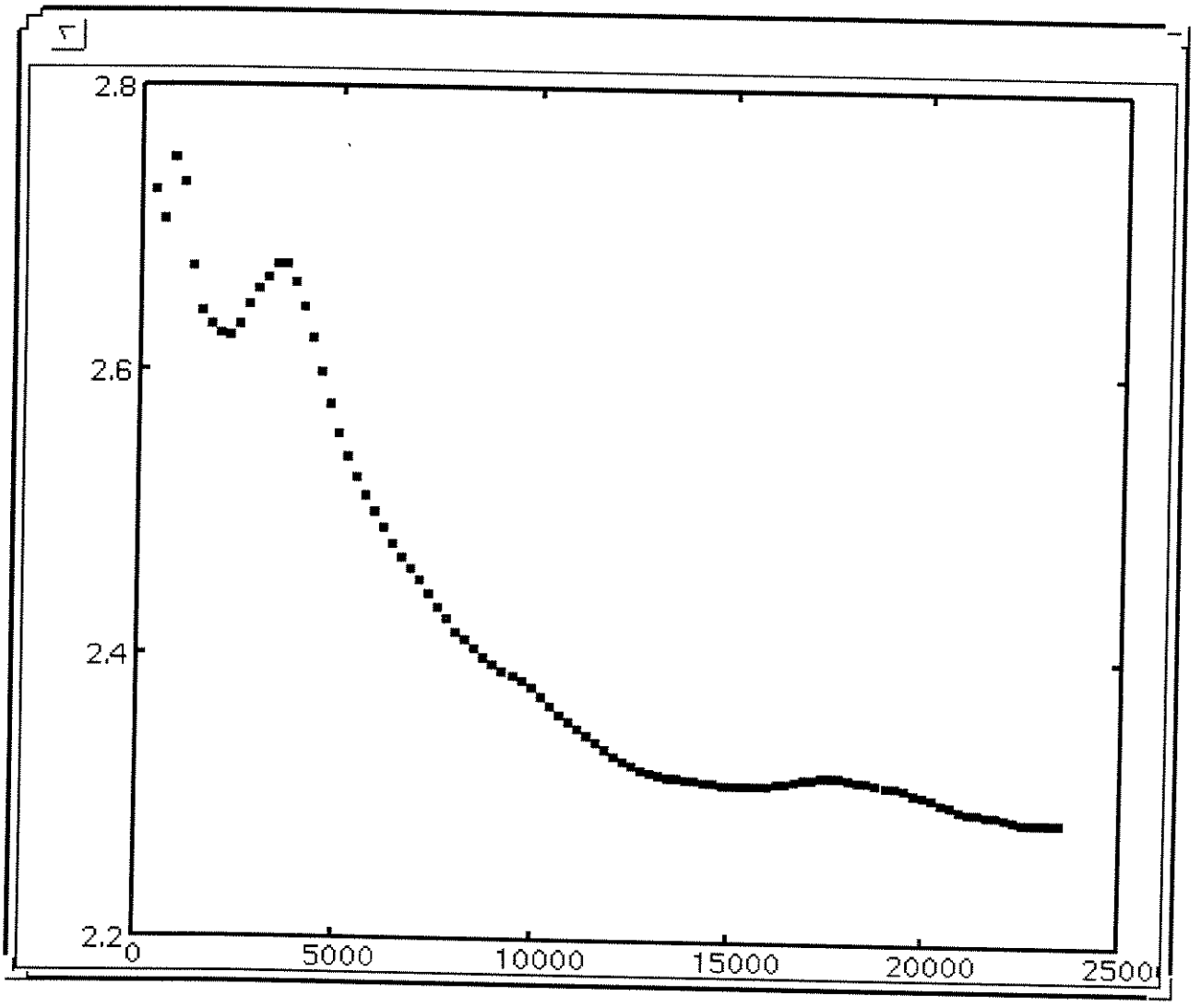

Figura 5.12: O gráfico que mostra os valores de irregularidades de um *pixel* para várias escalas. Os valores do eixo das abscissas estão em nanômetros. O eixo das ordenadas é adimensional.

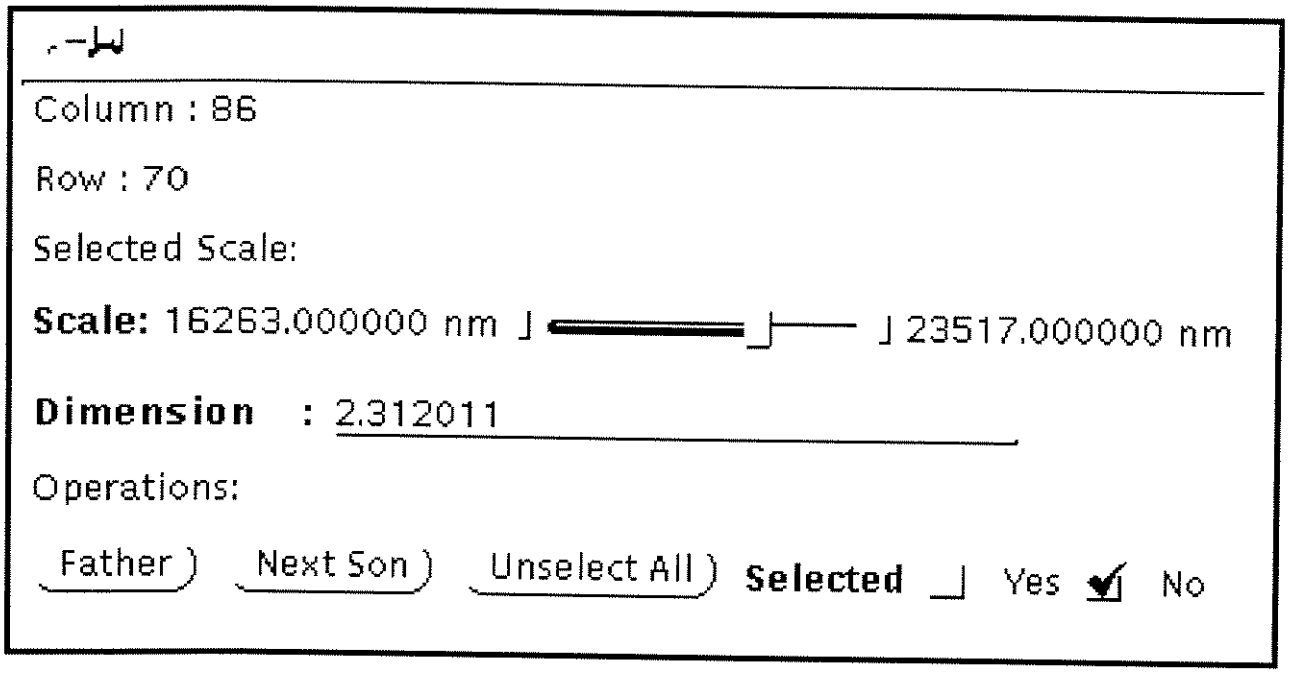

Figura 5.13: A janela de operação sobre o espectro de irregularidades.

pixels, recebam maior quantidade de níveis de cinza, dentre os 256 possíveis. Dessa forma, as faixas de valores mais populosas serão estendidas permitindo uma maior visualização de seus detalhes. A figura 5.14 mostra uma ilustração esquemática desse processo. No eixo horizontal são mostrados os valores dos *pixels* e no eixo vertical são mostrados os níveis de cinza. Na figura 5.14 mostramos uma situação em que os *pixels* da imagem assumem valores entre 200 e 2750, de forma que cada uma das 255 faixas de quantização tem extensão de 200. O gráfico mostrado fornece, para cada valor de *pixel*, o nível de cinza correspondente após a quantização. Veja que a primeira faixa da esquerda possui mais níveis de cinza que a segunda faixa justamente pelo fato de possuir maior número de valores de *pixels*, os quais estão representados pelo pontos.

O importante é observar que esse método de quantização é muito melhor para visualização do que simplesmente distribuir linearmente os valores do *pixels* entre os 256 níveis de cinza. Imagine um caso degenerado de imagem em que praticamente todos os *pixels* assumem valores muito próximos entre si e os outros assumem valores muito maiores. No caso de mapeamento linear, a imagem será mostrada com praticamente um único nível de cinza correspondendo aos *pixels* mais numerosos. Com isso, os detalhes da imagem nesses *pixels* ficarão pouco visíveis. Se, por outro lado, for usado o mecanismo de quantização exposto, isso não acontecerá.

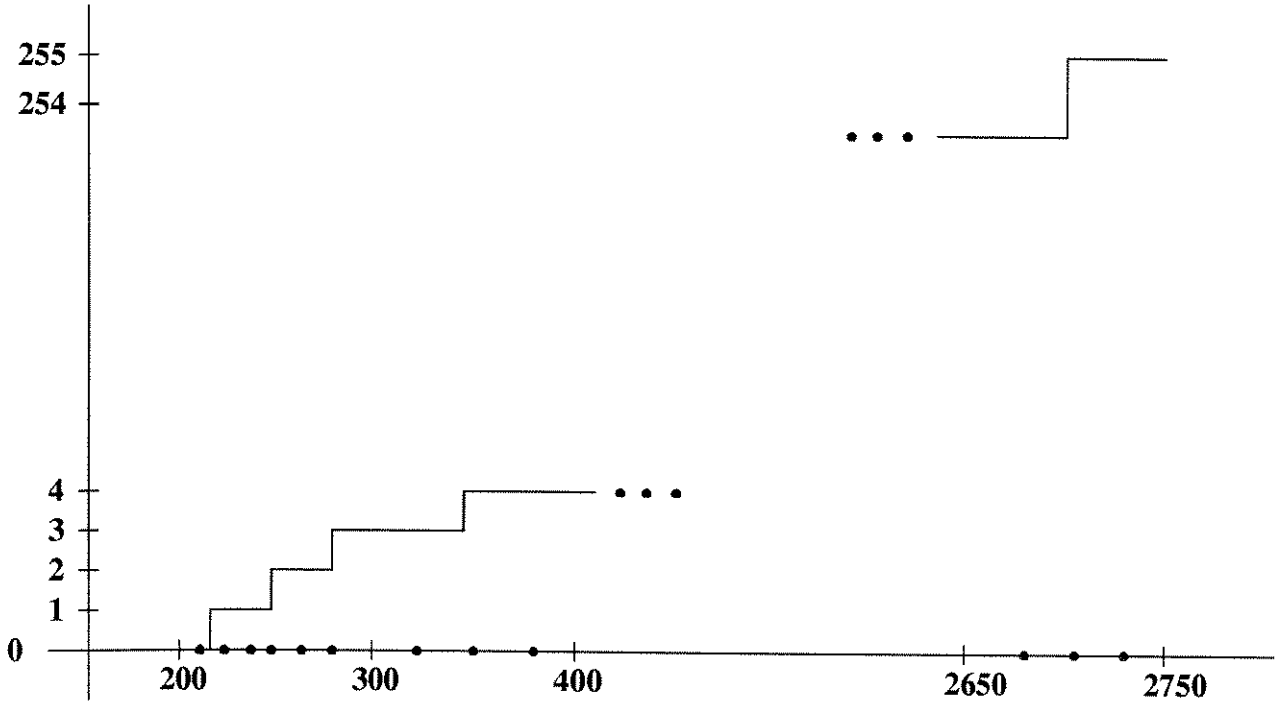

Figura 5.14: O processo de quantização do valores dos *pixels*.

Capítulo 6

Estudo de Imagens

Queremos aqui testar quão eficiente é o algoritmo que elaboramos na determinação dos tamanhos das irregularidades (de acordo com o conceito exposto na seção 2) da superfície de imagens topográficas e na segmentação dessas imagens em regiões de maior e menor irregularidade em determinada escala.

Antes de começar a análise de algumas imagens, precisamos colocar algumas observações que justificam o uso das imagens que escolhemos. Um aspecto central do método que propusemos é o cálculo da massa em escala δ, $\mu_P^\delta(\delta)$, ao redor de um *pixel* P da imagem. Esse cálculo envolve a avaliação do desvio padrão dos valores dos *pixels* determinados pelo quadrado resultante da projeção de cubos de lado δ sobre o plano base da imagem. Se o valor de δ for maior que a amplitude máxima $\mathcal{A}$ de oscilação da imagem, ou seja, a diferença entre os valores máximo e mínimo de seus *pixels*, o valor do desvio padrão dos *pixels* dentro de cada quadrado de lado δ será sempre menor ou igual a δ, levando a um valor máximo previsível para a massa $\mu_P^\delta(\delta)$. Ou seja, $\mu_P^\delta(\delta) \leq \mathcal{A}\delta^2$. Assim, $\mu_P^\delta(d)$, para a seqüência 4.3 de valores de d, será menor ou igual a $K_1\mathcal{A}\delta^2$, para uma dada constante K_1 calculável a partir da seqüência 4.3. Por outro lado, por motivos práticos de cálculo numérico, fizemos com que $\mu_P^\delta(\delta) \geq K_2\delta^2$, para uma dada constante K_2. Se assim não fosse e tivéssemos uma imagem que fosse um plano horizontal, teríamos valores de $\mu_P^\delta(\delta) = 0$, o que poderia causar problemas por ocasião do cálculo dos logaritmos de $\mu_P^\delta(d)$. Como resultado, para valores de $\delta > \mathcal{A}$, os valores $\mu_P^\delta(d)$ ficam limitados a um faixa de amplitude máxima que independe de δ. Por outro lado, na seqüência 4.3, os valores de d ocupam um faixa de amplitude igual a $2nf\delta$, ou seja, crescente com δ. A figura 6.1 ilustra essas observações. Nela vemos os gráficos $\log(\mu_P^\delta(d)) \times \log(d)$ para dois valores de $\delta > \mathcal{A}$. Em ambos os casos, os pontos do gráfico possuem suas ordenadas limitadas à faixa entre os valores y_1 e y_2. No caso da escala δ_1, a faixa de valores das abscissas dos pontos do gráfico possui amplitude igual a $x_2 - x_1$, a qual é menor que a amplitude da faixa de valores de abscissas dos pontos do gráfico para a escala δ_2. Veja que, ao fazermos regressão linear com os pontos do gráfico, é mais provável que a inclinação da reta no primeiro caso seja maior que a inclinação no segundo caso. Ou seja, a inclinação máxima da reta de regressão do gráfico $\log(\mu_P^\delta(d)) \times \log(d)$ tende a fica cada vez menor à medida que δ aumenta, o que é equivalente a dizer

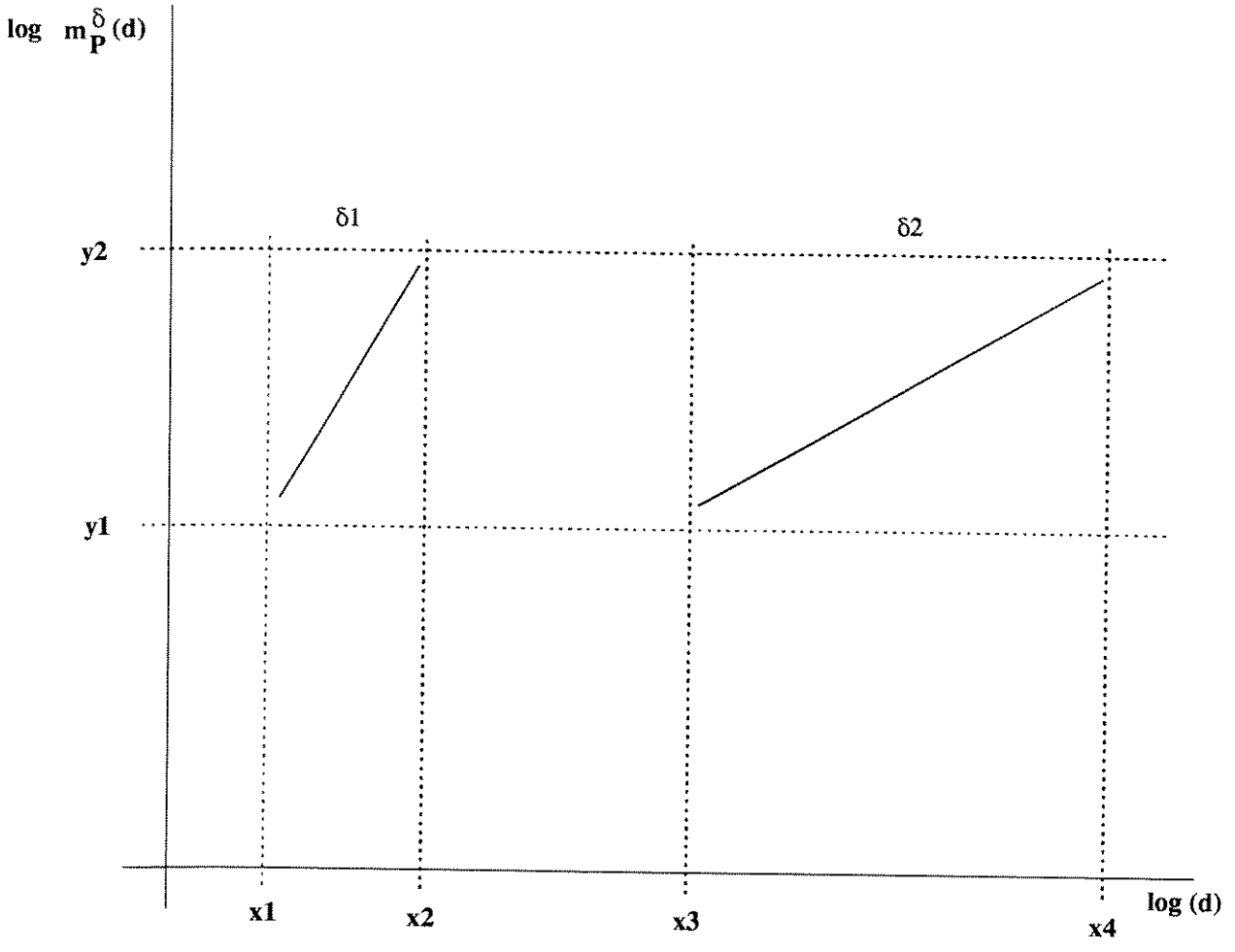

Figura 6.1: O decréscimo da irregularidade para $\delta > \mathcal{A}$.

que a irregularidade calculada tende a apresentar valores máximos decrescentes com *delta* $> \mathcal{A}$. Como resultado desse efeito, o gráfico que mostra o espectro de dimensões para várias escalas para um dado *pixel* P tenderá a apresentar valores cada vez menores e oscilará dentro de uma faixa de amplitude cada vez menor quando a escala cresce com valores maiores que $\mathcal{A}$. A figura 6.2 ilustra esse efeito, mostrando o espectro de irregularidades para um *pixel* de uma imagem com $\mathcal{A} = 40$. É importante observar, entretanto, que não só a oscilação vertical da imagem contribui para aumento da irregularidade, conforme mostraremos a seguir. Em muitos casos, os valores das irregularidades para $\delta > \mathcal{A}$ oscilam com amplitudes maiores ou iguais que no caso em que $\delta < \mathcal{A}$, por existirem irregularidades com tamanho ou espaçamento maiores que $\mathcal{A}$ na superfície da imagem.

Outro ponto a observar é que usaremos muito freqüentemente, no cálculo do espectro de dimensões, imagens com uma distribuição homogênea de irregularidades ao longo de seu plano

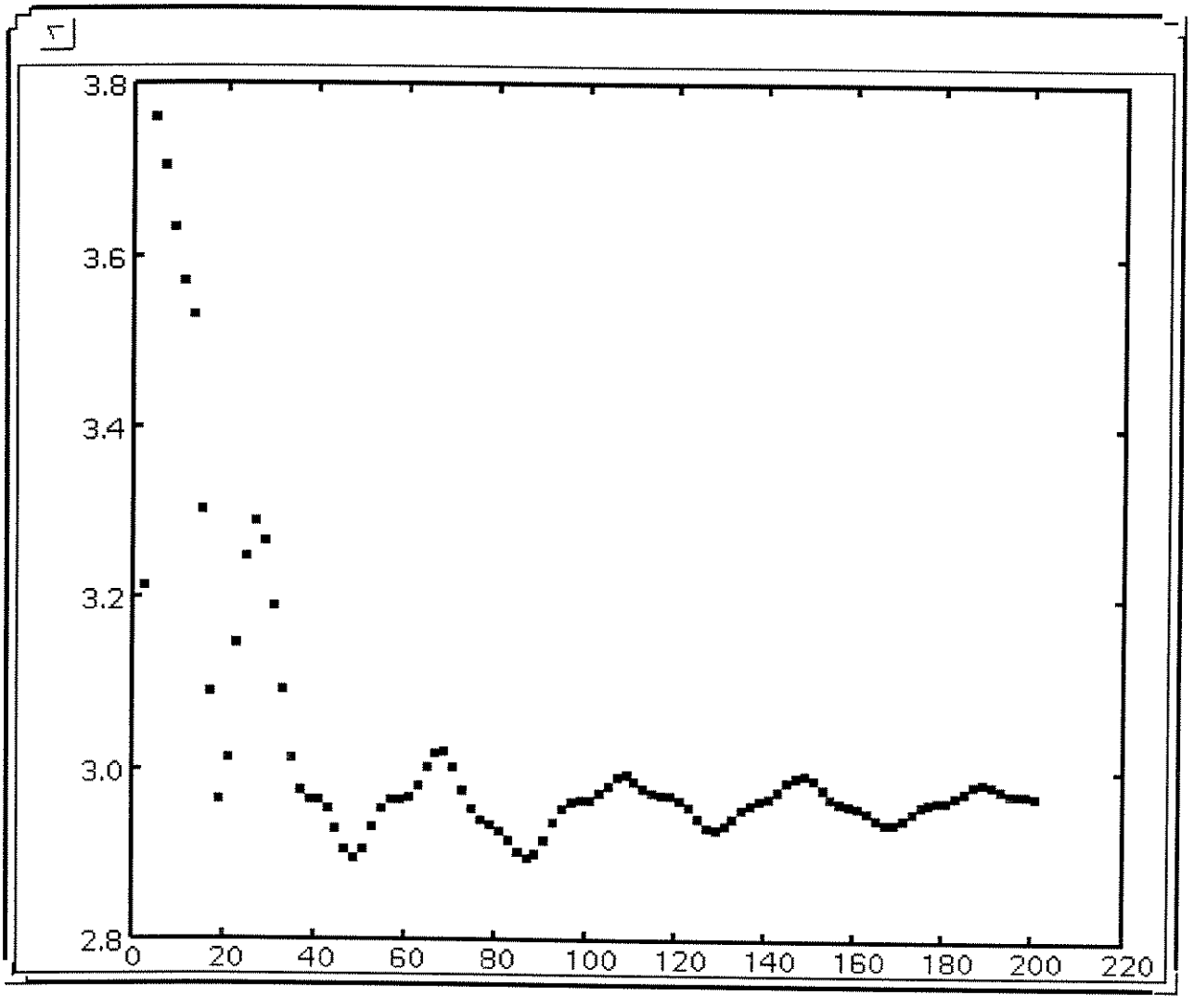

Figura 6.2: O espectro de irregularidades para uma imagem com $\mathcal{A} = 40$. O valores do eixo das abscissas representam quantidades de *pixels*. O eixo das ordenadas é adimensional.

base. Se usássemos imagens cujos tipos e tamanhos de irregularidades variassem ao longo do mapa de *pixels*, estaríamos mascarando o cálculo na tentativa de determinar justamente os tamanhos dessas irregularidades. Nesse caso, seria de se esperar que o espectro de irregularidades variasse muito de *pixel* a *pixel*, tornando difícil ou impedindo a análise. Por outro lado, no cálculo do mapa de irregularidades para uma determinada escala, estaremos propositadamente usando imagens com variação dos tipos e tamanhos de irregularidades ao longo do mapa de *pixels* para ilustrar o uso do algoritmo na segmentação de imagens em regiões de menor ou maior irregularidade.

6.1 O Uso do Espectro de Irregularidades

Vamos analisar, em primeiro lugar, o uso do espectro de irregularidades na determinação da largura dos grãos que ocorrem na superfície da imagem e do espaçamento entre os mesmos.

Para isso, usaremos imagens compostas por pirâmides de mesma altura e com base quadrada de mesmo tamanho dispostas de forma que haja sempre um mesmo número de pirâmides por linha e coluna. O espaçamento entre as pirâmides é sempre o mesmo e cada pirâmide localizada num *pixel* na posição (i, j) possui quatro vizinhos imediatos $(i-d, j)$, $(i+d, j)$, $(i, j-d)$ e $(i, j+d)$, onde d é o espaçamento entre as pirâmides. Ou seja, usaremos imagens com uma distribuição tabular de pirâmides de mesmo formato.

Queremos mostrar que podemos determinar o espaçamento e o tamanho horizontal das irregularidades, no caso pirâmides, observando as variações do espectro de irregularidades. Dessa forma, ao usarmos pirâmides de mesmo tamanho, estamos isolando o efeito das oscilações verticais da imagem. Se o espectro de irregularidades mostrar variações devido a oscilações verticais, deverá ser fácil isolá-las devido os valor único das alturas das pirâmides.

Como primeiro exemplo, usaremos imagens com pirâmides de altura igual a 5 *pixels*, base quadrada de lado 30 *pixels* e com espaçamentos hozizontal e vertical entre seus centros iguais a 40 *pixels*. A figura 6.5 mostra a imagem gerada no sistema Fraktal. No processo de captura dessa imagem, o programa utilizado não permitiu reproduzir com exatidão os níveis de cinza utilizados. Entretanto, quanto maior a elevação topográfica da imagem, maior o nível de cinza utilizado, o que resulta numa gradação crescente de níveis de cinza da base ao topo da pirâmide. Por isso, as regiões mais claras correspondem a posições mais elevadas da imagem.

Em ambos os espectros de irregularidades calculados para essa imagem usamos $n = 2$ na seqüência 4.3. No espectro da figura 6.4, usamos como base de cálculo um *pixel* (i, j) resultado

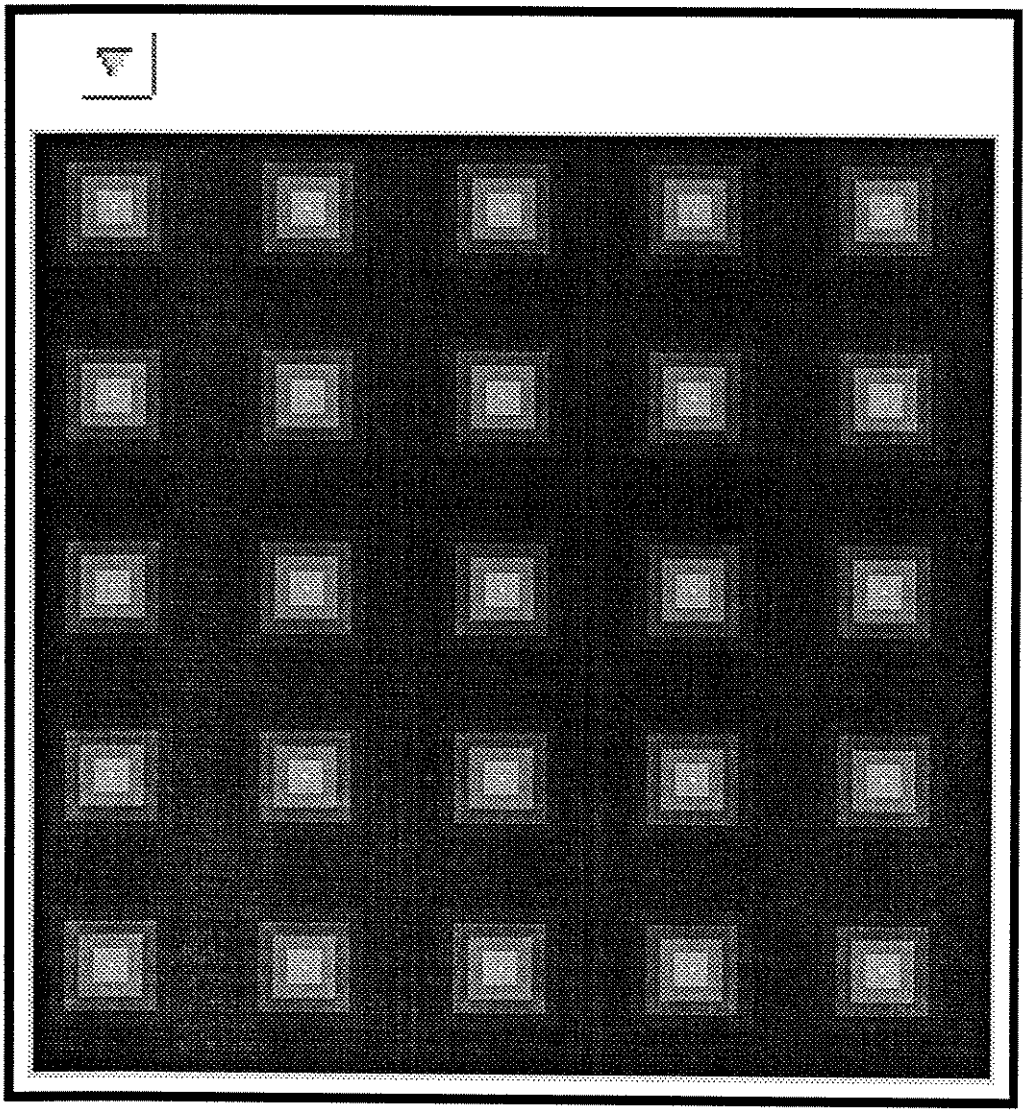

Figura 6.3: Uma imagem com pirâmides de altura 5, base de lado 30 e com espaçamentos horizontal e vertical iguais a 40.

da projeção do topo de uma das pirâmides sobre o plano base da imagem. No espectro da figura 6.5, tomamos como base de cáculo um *pixel* localizado no ponto médio (k, l) entre quatro pirâmides com bases centralizadas nos *pixels* $(k-20, l-20)$, $(k+20, l-20)$, $(k-20, l+20)$ e $(k+20, l+20)$. Veja que, no primeiro caso, o espectro apresenta picos por volta dos valores 31, 57, 75, 141 e um vale em torno do valor 95. Para entender o porquê desses valores, temos que lembrar que cada um deles representa uma escala que define uma grade de quadrados de lado igual à escala, centralizada no *pixel* de cálculo. Como usamos a seqüência 4.3 com $n = 2$ e $f = 0.1$, temos que a grade é composta de 3×3 quadrados, como mostra a figura 4.9. Ou seja, estamos considerando uma vizinhança em torno do quadrado central com lado igual à escala escolhida. Quando aumentamos sucessivamente o valor da escala, aumentamos o diâmetro desta vizinhança, fazendo com que ela atinja e passe por oscilações que acontecem na superfície da imagem. Essas oscilações levam, naturalmente, a aumentos ou diminuições dos valores calculados para $\mu_P^\delta(d)$. Ou seja, quando a vizinhança cresce e aproxima-se de um conjunto de irregularidades (conjunto de picos ou vales da superfície), os valores $\mu_P^\delta(d)$ tendem a crescer ou diminuir com o valor de d (diâmetro da vizinhança). Se $\mu_P^\delta(d)$ aumenta significativamente quando d cresce, fazendo com que $\log(\mu_P^\delta(d))$ cresça mais com $\log(d)$, temos um aumento do valor calculado da irregularidade. Por outro lado, se $\mu_P^\delta(d)$ diminui quando d cresce, fazendo com que $\log(\mu_P^\delta(d))$ tenha um crescimento muito menor com $\log(d)$, temos uma diminuição do valor calculado da irregularidade. No primeiro caso, temos um pico no espectro de dimensões e no segundo caso temos um vale. Assim, fica fácil interpretar os valores citados. O *pixel* (i, j) é cercado por anéis de pirâmides. O primeiro deles é composto pelas pirâmides $(i-40, j-40), (i-40, j), (i-40, j+40), (i, j-40), (i, j+40), (i+40, j-40), (i+40, j), (i+40, j+40)$, o segundo é composto pelas pirâmides centralizadas nos *pixels* $(i-80, j-80), (i-80, j-40), (i-80, j), (i-80, j+40), (i-80, j+80), (i-40, j-80), (i-40, j+80), (i, j-80), (i, j+80), (i+40, j-80), (i+40, j+80), (i+80, j-80), (i+80, j-40), (i+80, j), (i+80, j+40), (i+80, j+80)$ e assim sucessivamente. O valor 31 é atingido quando a vizinhança de cálculo da irregularidade (quadrado central da grade de 3×3 quadrados) engloba toda a base da pirâmide centralizada em (i, j). O segundo pico (57) ocorre quando a vizinhança atinge o primeiro anel de pirâmides. O terceiro pico ocorre aproximadamente quando a vizinhança atinge os topos das pirâmides deste primeiro anel e o vale ocorre quando a vizinhança aproximadamente engloba essas pirâmides. Por fim, o pico em 141 ocorre quando o quadrado central da grade aproximadamente atinge o segundo anel de pirâmides ao redor do *pixel* (i, j).

A mesma análise pode ser feita quando usamos o espectro calculado com base no *pixel* (k, l). Esse *pixel* é cercado por anéis de pirâmides, o primeiro composto pelas pirâmides centralizadas nos *pixels* $(k-20, l-20), (k-20, l+20), (k+20, l-20), (k+20, l+20)$, o segundo composto pelas

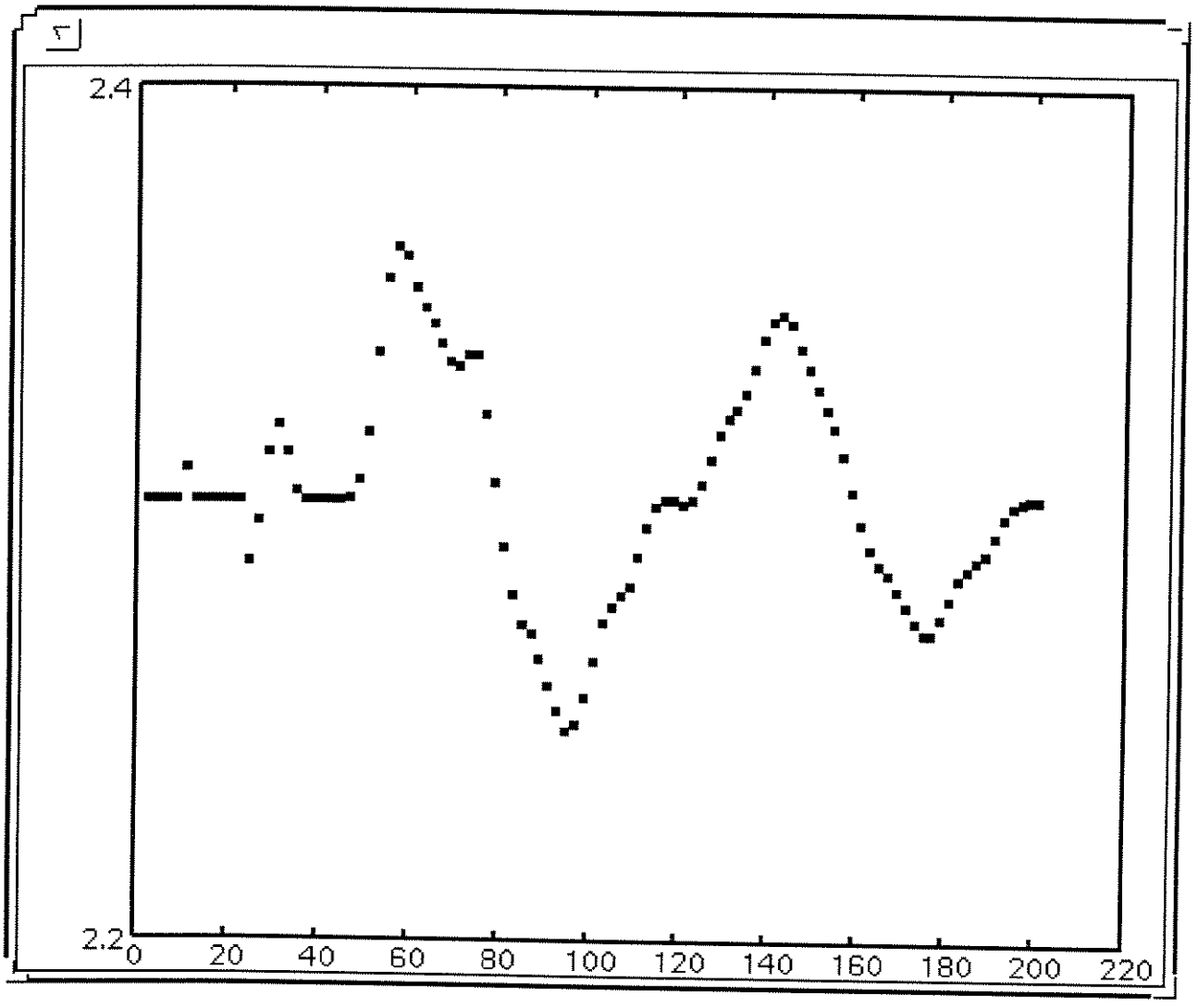

Figura 6.4: Espectro de irregularidades com $n = 2$, para um *pixel* no topo de uma pirâmide da figura 6.3. Os valores do eixo das abscissas representam quantidade de *pixels*. O eixo das ordenadas é adimensional.

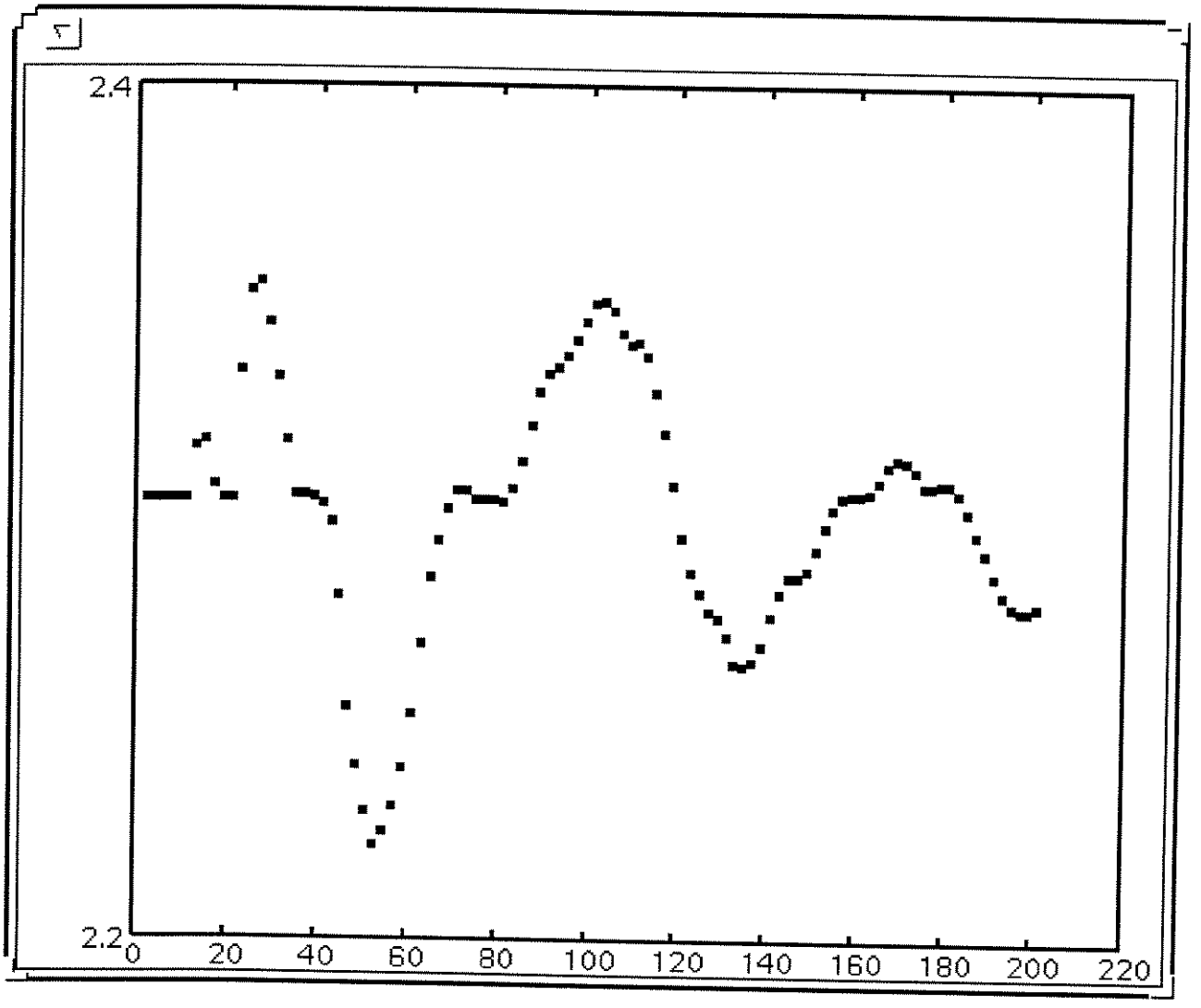

Figura 6.5: Espectro de irregularidades com $n = 2$, para um *pixel* no ponto médio entre quatro pirâmides da figura 6.3. Os valores do eixo das abscissas representam quantidade de *pixels*. O eixo das ordenadas é adimensional.

pirâmides centralizadas nos *pixels* $(k-60, l-60), (k-60, l-20), (k-60, l+20), (k-60, l+60), (k-20, l-60), (k-20, l+60), (k+20, l-60), (k+20, l+60), (k+60, l-60), (k+60, l-20), (k+60, l+20), (k+60, l+60)$ e assim sucessivamente. No espectro de irregularidades, o pico que ocorre por volta de 15 é atingido quando a vizinhança atinge as quinas das pirâmides do primeiro anel. O próximo pico ocorre por volta do valor de escala igual a 25. Usando a função que calcula distâncias entre *pixels* de imagens sob o sistema Fraktal, descobrimos que esse valor é aproximadamente igual à distância entre as quinas mais próximas dos quadrados $(k-20, l-20)$ e $(k+20, l+20)$ ou entre as quinas mais próximas dos quadrados $(k-20, l+20)$ e $(k+20, l-20)$. Se considerarmos um círculo centralizado no *pixel* (k, l), quando seu diâmetro fosse aproximadamente igual a 25, ele atingiria as pirâmides do primeiro anel. Ou seja, se olharmos igualmente em todas as direções horizontais ao redor do *pixel* (k, l), vemos que num diâmetro igual a 25 ao redor deste *pixel* temos um aumento considerável do grau de oscilação da imagem. Assim, seria de se esperar que o espectro de irregularidades apresentasse variações para uma escala igual a esse valor. O primeiro vale do espectro ocorre por volta do valor 55, que é aproximadamente igual à distância entre os topos das pirâmides $(k-20, l-20)$ e $(k+20, l+20)$ ou das pirâmides $(k-20, l+20)$ e $(k+20, l-20)$. Novamente, se considerarmos um círculo centrado no *pixel* (k, l), vemos que quando seu diâmetro é aproximadamente igual a 55, ele atinge os topos das pirâmides do primeiro anel. Considere agora o valor de escala 40. Por volta desse valor, os valores das irregularidades calculadas começam a cair bruscamente. Veja que na imagem, esse valor é igual ao lado do quadrado centralizado no *pixel* (k, l) quando ele atinge os topos as pirâmides do primeiro anel. Por fim, existem vários outros picos e vales que poderiam sofrer análises semelhantes.

Em todas essas análises, encontramos diversas interpretações para os picos ou vales que ocorrem no espectro de dimensões. Na verdade, é de se esperar que existam várias delas quando olhamos através de diferentes prismas para a vizinhança ao redor do *pixel* tomado como base de cálculo. Em todas essas análises, entretanto, os picos, vales ou alterações bruscas no espectro corresponderam a mudanças significativas na superfície da imagem: em alguns casos a vizinhança ao redor do *pixel* de cálculo atingia um conjunto de vales, em outros atingia um conjunto de picos, etc. A explicação que me parece mais correta para esse comportamento advém, entretanto, dos comentários expostos no início desta seção : quando a vizinhança de cálculo da irregularidade (de diâmetro d) atinge e passa por alterações significativas na superfície da imagem, $\mu_P^{\delta}(d)$ começa a crescer ou diminuir significaticamente com o valor de d, resultando nas alterações observadas nos espectros de irregularidades. Em resumo, essas alterações bruscas do espectro permitem identificar, essencialmente, as variações da superfície da imagem na vizinhança no *pixel* tomado para cálculo, permitindo extrair informações como o tamanho dos grãos que ocorrem na superfície e

o espaçamento entre eles, entre várias outras, associadas a alterações das alturas de conjuntos com número significativo de *pixels*. Por fim, se a imagem é suficientemente homogênea em suas irregularidades ao longo de seu mapa de *pixels*, as alterações do espectro devem ser aproximadamente as mesmas para qualquer ponto considerado, permitindo identificar os valores médios das irregularidades da superfície.

Cabe aqui um comentário com relação aos objetivos expostos na seção 2. Naquela seção observamos que as irregularidades calculadas deveriam crescer quando a escala considerada fosse da ordem de grandeza das irregularidades que ocorrem na superfície da imagem. Existem casos em que isso é verdade. Se essa equivalência de tamanhos ocorre, foi observado que $\mu_P^\delta(\delta)$ tente a apresentar valores relativamente altos comparados com o volume δ^3 do cubo de lado δ. Como estamos usando uma seqüência de valores de diâmetro d de vizinhança que varia pouco ao redor do valor δ, (veja seqüência 4.3), é de se esperar que os valores $\mu_P^\delta(d)$ sejam também relativamente altos comparados com os volumes da vizinhanças de diâmetro d. Bem, ao aumentarmos d de seu valor mínimo até seu valor máximo na seqüência 4.3, aumentamos o tamanho da vizinhança considerada. Como os valores das massas são altos, temos alguma chance de que, aumentando um pouco d, a superfície continue a apresentar o mesmo padrão alto de irregularidade, fazendo com que as massas $\mu_P^\delta(d)$ creçam devido a dois fatores: o aumento do diâmetro d e a manutenção desse padrão, uma vez que a vizinhança não altera muito. Com isso, temos também uma chance de que $\log(\mu_P^\delta(d))$ cresça razoavelmente com $\log(d)$. Veja que isso não ocorre no caso da vizinhança V_3 na figura 2.3. Nesse caso, a escala é muito maior que as irregularidades. Ao aumentarmos o tamanho da vizinhança, a massa interior aumenta simplesmente porque temos uma vizinhança maior e somente no sentido horizontal, uma vez que as oscilações verticais encontram-se totalmente dentro do cubo. Como isso, a massa $\mu_P^\delta(d)$ deve crescer com uma potência quadrática do valor de d (d^2) e a superfície comporta-se como um plano. Em resumo, matematicamente o que interessa é que a massa $\mu_P^\delta(d)$ cresça com expoentes grandes com o aumento de d, devido ao alto grau de irregularidades nas vizinhanças de um ponto. Isso pode não ocorrer em casos como o da vizinhança V_3 da figura 2.3 ou em casos em que, apesar de a massa $\mu_P^\delta(d)$ ser alta, ela não cresce com um expoente alto de d. Por isso, os objetivos expostos na seção 2 não são antigidos da forma como esperávamos. O método que propusemos mede efetivamente o expoente de crescimento da massa $\mu_P^\delta(d)$ com o valor de d. Por isso, fornece uma estimativa válida de irregularidade. Entretanto, a determinação dos tamanhos dos grãos das irregularidades dá-se pela observação das diversas oscilações do espectro de irregularidades e não apenas de seus picos, conforme discussão dos parágrafos anteriores.

Vejamos agora o comportamento do espectro de irregularidades na determinação dos tamanhos das oscilações verticais das imagens topográficas. Para tal, usamos para as análises a

serem efetuadas uma imagem resultante da adição de duas imagens básicas :

- Um mapa de pirâmides como o da figura 6.3, mas com pirâmides de bases de lado 5, alturas iguais a 50 e com espaçamentos horizontal e vertical iguais a 5.
- Uma senóide que para o *pixel* localizado na coluna i e linha j fornece o valor $\frac{40}{2}\sin(2\pi j/100)$. Ou seja, uma senóide calculada apenas com base no valor da linha do *pixel*, com amplitude igual 40 e com período igual a 100.

Com isso, produziu-se a imagem mostrada na figura 6.6. A senóide produz um conjunto de oscilações de grande período horizontal, mas com pequena amplitude e as pirâmides produzem um conjunto de pequenas oscilações na superfície da senóide.

Na verdade, para os resultados que iremos obter bastaria o uso do mapa de pirâmides. O uso da senóide tem apenas o objetivo de ampliar a amplitude máxima vertical $\mathcal{A}$ da imagem, de acordo com a discussão exposta no começo da seção 6 e evitar o efeito mostrado na figura 6.2. Pretendemos, em última análise, verificar se o espectro de irregularidades é capaz de determinar as diversas oscilações verticais da imagem. No caso em estudo, uma das oscilações é dada pelas alturas das pirâmides e possui valor igual a 50. Ao usarmos uma senóide de período igual a 100, pretendemos evitar que as oscilações que devem aparecer no espectro por volta desse valor interfiram com as oscilações que devem ser observadas por volta do valor correspondente às alturas das pirâmides. Mais ainda, ao usarmos uma senóide com pequena amplitude procuramos evitar também que suas oscilações verticais interfiram com as oscilações verticais da superfície da imagem causadas pelas pirâmides. Em resumo, usamos uma imagem com grande número de oscilações verticais de mesma amplitude localizadas sobre uma superfície de oscilação suave.

O espectro de irregularidades calculado para um *pixel* da imagem aleatoriamente selecionado e para valores $n = 2$ e $f = 0.1$ da seqüência 4.3 é mostrado na figura 6.7. A seleção do *pixel* tomado como base de cálculo foi aleatória porque o grau e tipo de irregularidades são homogêneos ao longo do plano base da imagem. Dessa forma, os espectros gerados para diferentes *pixels* mostraram-se muito parecidos entre si, apresentando os mesmos valores de escala que são significativos, ou seja, valores onde ocorrem picos ou variações bruscas do grau de irregularidade calculado.

Veja que realmente ocorre um pico bastante visível no espectro de irregularidades por volta do valor 50, correspondente ao tamanho das irregularidades que ocorrem na superfície da senóide, ou seja, à altura das pirâmides. Veja também que por volta da escala 100, o valor das

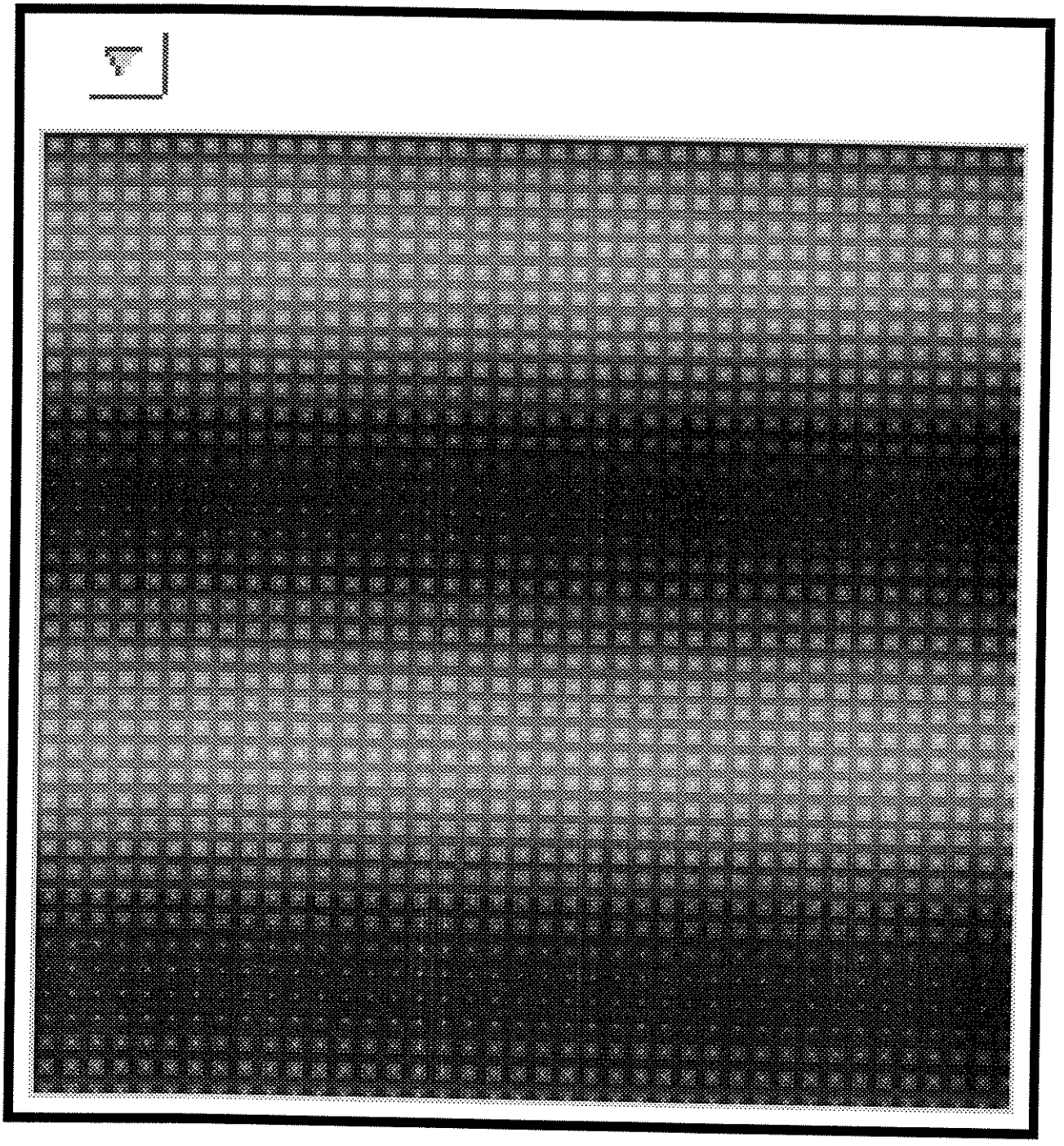

Figura 6.6: Imagem resultante da adição de uma senóide com um mapa de pirâmides.

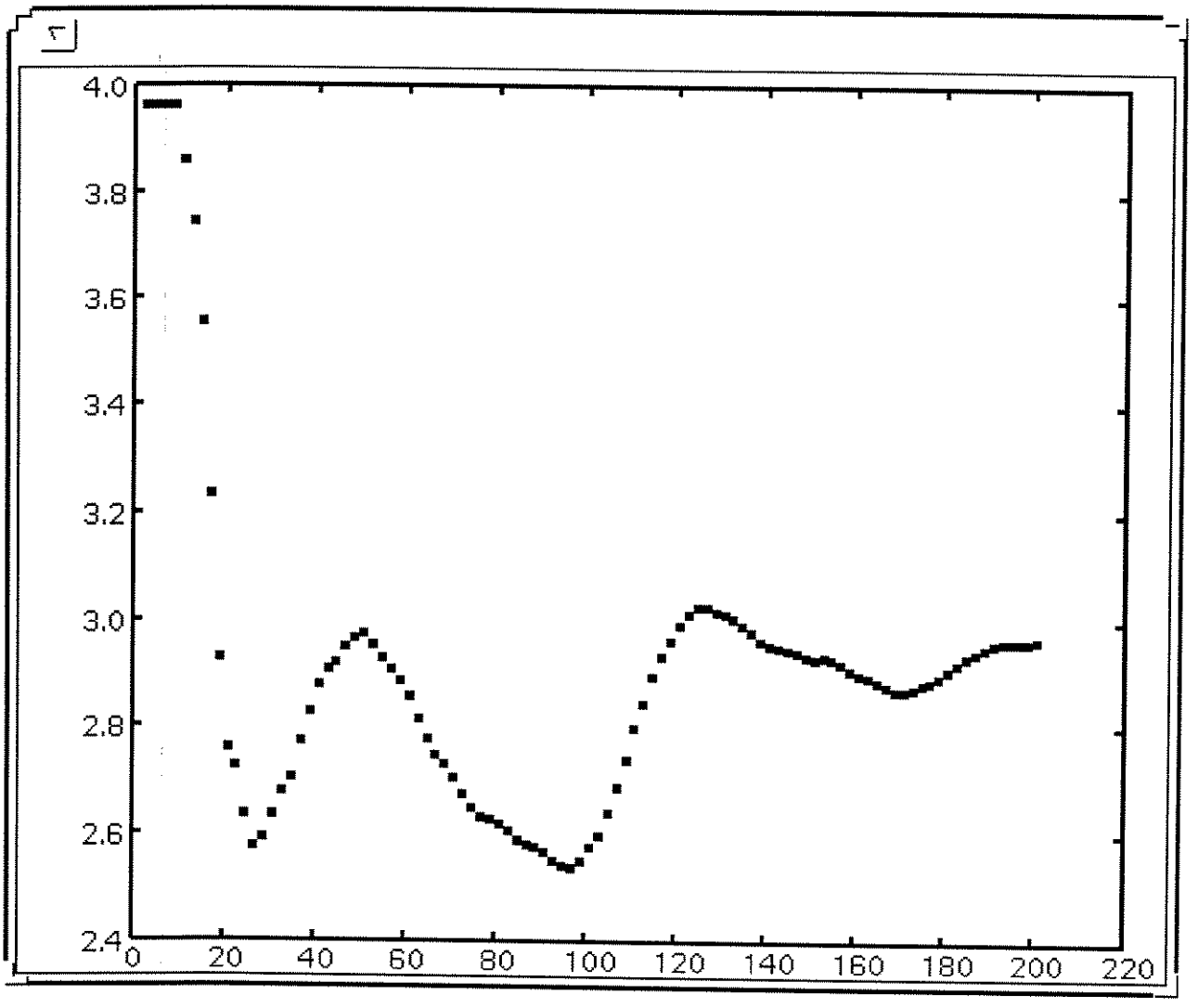

Figura 6.7: O espectro de irregularidades para a imagem da figura 6.6. Os valores do eixo das abscissas representam quantidades de *pixels*. O eixo das ordenadas á adimensional.

irregularidades começa a aumentar significativamente, correspondendo às distâncias das oscilações horizontais da imagem, ou seja, ao período da senóide.

Vários outros exemplos de imagens existem que permitem evidenciar a eficácia do espectro de irregularidades na determinação dos tamanhos das irregularidades verticais de imagens topográficas.

Vejamos agora o uso do mapa de irregularidades na segmentação de imagens. Como exemplo, selecionamos uma imagem dividida verticalmente ao meio em duas regiões de diferentes graus de oscilação vertical, mostrada na figura 6.8. Na parte da esquerda, o valor de cada *pixel* é dado por um número aleatório gerado a partir de uma seqüência com distribuição homogênea entre os valores 0 e 20. Na parte da direita, usamos a mesma distribuição de probabilidade com valores mínimo e máximo iguais a 0 e 40. No processo de captura da imagem, como os valores da parte esquerda oscilavam pouco, toda a região foi mapeada para o nível de cinza 0 (preto). A parte da direita, entretanto, ilustra muito bem o aspecto oscilatório vertical da imagem. Intuitivamente, fica claro que a parte da direita apresenta maior grau de irregularidade que a parte da esquerda.

Para a geração do mapa de irregularidades usamos novamente a seqüência 4.3 com valores $n = 2$ e $f = 0.1$ e uma escala igual a 11. O mapa gerado é mostrado na figura 6.9. Novamente, o processo de captura reduziu os níveis de cinza da imagem a exatamente 2, fazendo com que a região da esquerda aparacesse com o nível 0 e a da direita com o nível 255. O importante é que os *pixels* da parte direita apresentam em média um nível de cinza maior que os níveis de cinza dos *pixels* da parte esquerda, o que significa que apresentam também, em média, graus de irregularidade calculados maiores.

Como último exemplo, vejamos a atuação do espectro de irregularidades sobre uma imagem de um filme de diamante obtida com o sistema *Nanoscope II*, mostrada na figura 6.10. Se usarmos a funcionalidade do sistema Fraktal, veremos que os grãos apresentam larguras em torno de 4000 nm. Ao calcularmos espectros de dimensões para uma série aleatória de *pixels*, obteremos picos por volta desse valor, como mostrado na figura 6.11. A imagem 6.10 mostra grãos bastante próximos uns dos outros e com uma distribuição bastante homogênea sobre o substrato, fazendo com que as larguras dos grãos sejam aproximadamente iguais aos espaçamentos entre eles. Dessa forma, os picos dos espectros de irregularidade fornecem-nos uma idéia das larguras médias dos grãos do filme. Se *pixels* diferentes forem escolhidos, o espectro obtido é diferente, mas todos eles apresentam um pico bastante visível por volta do valor de escala 4000 nm.

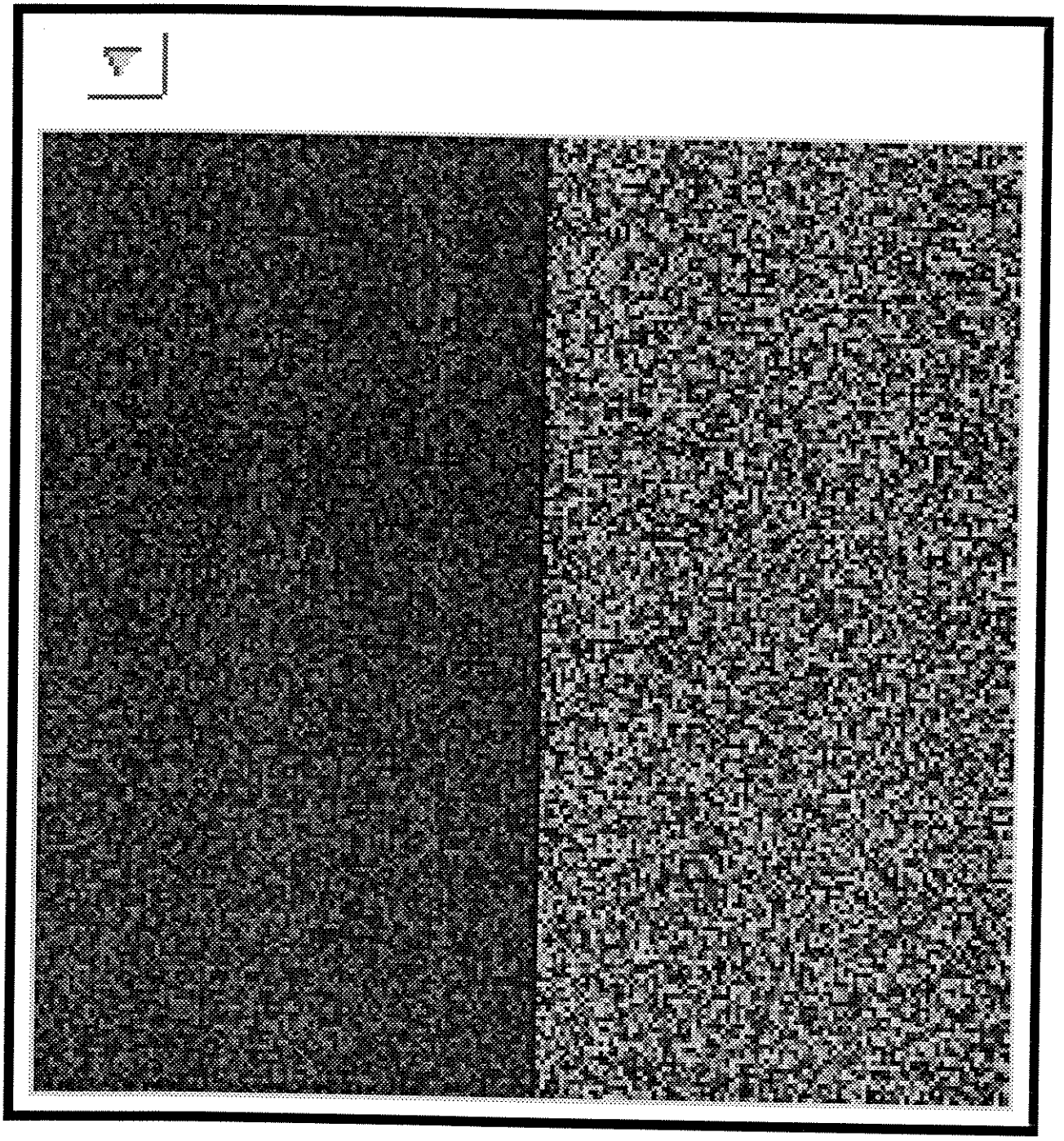

Figura 6.8: Imagem com duas regiões com diferentes níveis de oscilação vertical.

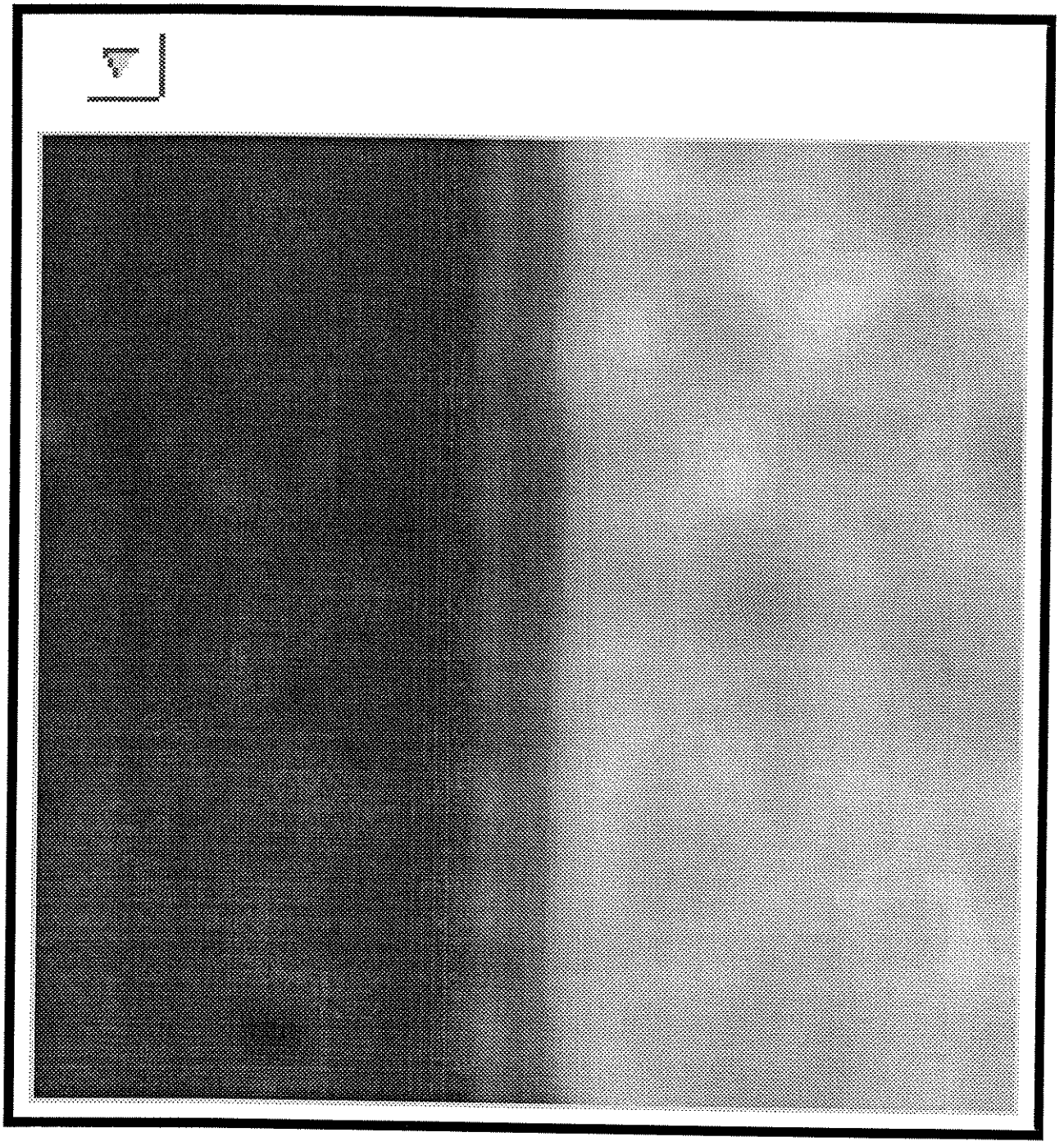

Figura 6.9: Mapa de irregularidades para a imagem da figura 6.8.

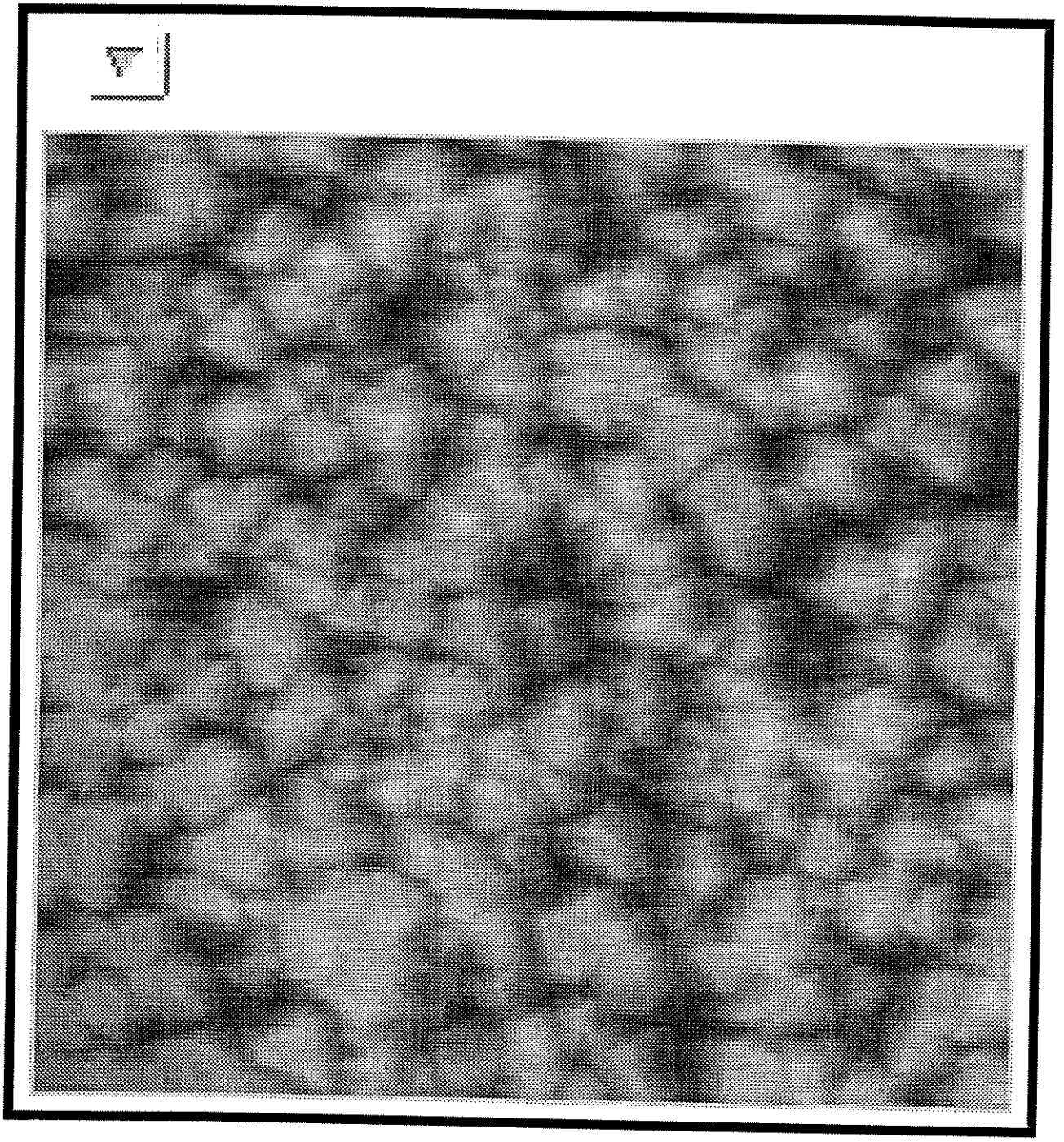

Figura 6.10: Imagem de um filme de diamante obtida com o sistema Fraktal.

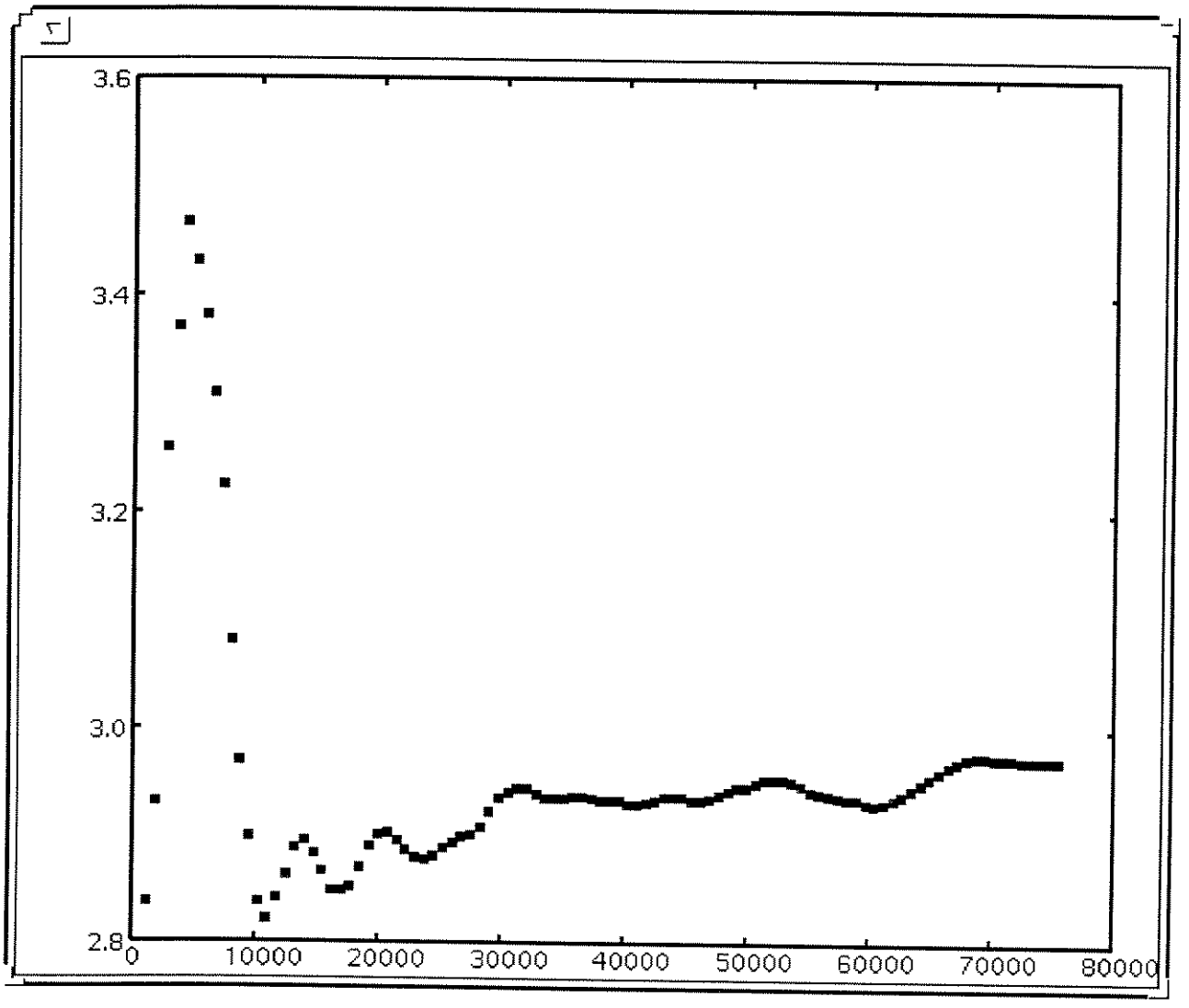

Figura 6.11: O espectro de irregularidades para um *pixel* da imagem 6.10. Os valores do eixo das abscissas estão em nanômetros. O eixo das ordenadas é adimensional.

Capítulo 7

Conclusão

O estudo do método que propusemos está ainda muito longe de ser exaurido. Durante as análises de imagens que fizemos pudemos comprovar sua eficácia em vários casos e para algumas de suas variáveis. Na seqüência 4.3 para valores de vizinhanças fizemos f igual a 0.1 e usamos, em toda a seção 6 o valor n igual a 2. As vizinhanças produzidas com esses fatores são boas por serem pouco extensas. Entretanto, as outras vizinhanças geradas aos variarmos esses fatores produziram resultados semelhantes, mas poderiam ser melhor exploradas na tentativa de identificar a qualidade do mapa e espectro de irregularidades que produzem.

Propusemos um método de distribuição de massa para superfícies de imagens topográficas baseado no cáculo da variância dos valores de seus *pixels*, o que levou à definição do operador $\mu_P^\delta(d)$. Apesar de apresentar várias vantagens, não exploramos outras possibilidades, o que poderia resultar em métodos que fossem tão ou mais eficientes e que pudessem estimar verdadeiramente a dimensão fractal da imagem em um dado ponto e numa determinada escala.

O ambiente Fraktal que elaboramos para estudo do algoritmo mostrou-se muito útil e fácil de usar, permitindo elaborar diversos algoritmos tanto para geração de imagens fictícias como para cálculo de irregularidades. Estudos futuros na linha de pesquisa desta Tese sem dúvida poderão beneficiar-se muito de seu uso, ainda mais se alguns de seus problemas forem resolvidos, entre eles a falta de ligação dinâmica na implementação atual dos módulos de novos algoritmos e de novos objetos gráficos. Se isso for consertado, evitando compilações e ligações desnecessárias, o trabalhos dos desenvolvedores tornar-se-á ainda mais fácil.

Finalmente, podemos dizer pelas análises que realizamos que o método desenvolvido é bastante eficaz para estudos de granulometria de imagens topográficas, permitindo identificar os tamanhos e separação dos grãos e vales que ocorrem em suas superfícies. O processo de segmentação das imagens em regiões de maior ou menor suavidade comportou-se muito bem, apesar de alguns inconvenientes causados pela grande freqüência de oscilações locais dos valores calculados pelo algoritmo ao longo do mapa de *pixels*. Entretanto, quando temos regiões com diferenças significativas em suas irregularidades, esse incoveniente desaparece, permitindo realizar a segmentação.

BIBLIOGRAFIA

[ABM94] A. Arneodo, E. Bacry, and J. F. Muzy. Solving the inverse fractal problem from wavelet analysis. *Europhys. Lett.*, 25(7):479–484, March 1994.

[BDM+88] Michael F. Barnsley, Robert L. Devaney, Benoit B. Mandelbrot, Heinz-Otto Peitgen, Dietmar Saupe, and Richard F. Voss. *The Science of Fractal Images.* Springer-Verlag, 1988.

[BGSZ94] R. Barberi, M. Giocondo, G. V. Sayko, and A.K. Zvedin. AFM experimental observation and fractal characterization of an SiO-coated plate for nematic bistable anchoring. *J. Phy.: Condensed Matter*, 6:A275–A278, 1994.

[Có93] Cesar Augusto Rosales Córdova. Caracterização de rugosidades de superfícies atômicas por fractais. Master's thesis, DSIF - FEEC - UNICAMP, 1993.

[CC88] S. C. Chapra and R. P. Canale. *Numerical Methods for Engineers.* McGraw-Hill Book Company, 1988.

[CS95] B. B. Chaudhuri and N. Sarkar. Texture segmentation using fractal dimension. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(1):72–77, January 1995.

[CWLK89] S. Chesters, H. Y. Wen, M. Lundin, and G. Kasper. Fractal-based characterization of surface texture. *Applied Surface Science*, 40:185–192, 1989.

[Dav94] E. B. Davies. Two-dimensional riemannian manifolds with fractal boundaries. *J. London Math. Soc.*, 49(2):343–356, 1994.

[DM76] B. P. Demidovich and I. A. Maron. *Computational Mathematics.* Mir Publishers - Moscow, 1976.

[DZT+89] B. Dubuc, S. W. Zucker, C. Tricot, J. F. Quiniou, and D. Wehbi. Evaluating the fractal dimension of surfaces. *Proc. R. Soc. Lond. A*, 425:113–127, 1989.

[Edg49] Gerald A. Edgar. *Measure, Topology, and Fractal Geometry.* Springer-Verlag New York Inc., 1949.

[Fal90] Kenneth Falconer. *Fractal Geometry - Mathematical Foundations and Applications.* Johm Wiley & Sons, 1990.

[GMHG94] J. C. Gallant, I. D. Moore, M. F. Hutchinson, and P. Gessler. Estimating fractal dimension of profiles: A comparison of methods. *Mathematical Geology*, 26(4):455–481, 1994.

[Han94] G. Handke. Fractal dimensions in the phase space of two-electron atoms. *Physical Review A*, 50(5):R3561–R3564, November 1994.

[HD95] P. Hall and S. Davies. On direction-invariance of fractal dimension on a surface. *Applied Physics A*, 60:271–274, 1995.

[Hel90] D. Heller. *XView Programming Manual*, volume 7. O'Reilly & Associates, Inc., Sebastapol, CA, 2 edition, 1990.

[HLD94] Q. Huang, J. R. Lorch, and R. C. Dubes. Can the fractal dimension of images be measured? *Pattern Recognition*, 27(3):339–348, 1994.

[Jon89] J. Jones. *Introduction to the X Window System.* Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[JT94] R. Jullien and R. Thouy. Numerical investigation of two-dimensional projections of random fractal aggregates. *Physical Review E*, 50(5):3878–3882, November 1994.

[Kan94] H. Kantz. Quantifying the closeness of fractal measures. *Physical Review E*, 49(6):5091–5097, June 1994.

[KR78] B. W. Kerningham and D. M. Ritchie. *The C Programming Language.* Prentice Hall, Englewood Cliffs, New Jersey, 1978.

[Lam86] Leslie Lamport. *A Document Preparation System.* Addison-Wesley, 1986.

[LFHB96] M. C. V. Lopes, S. G. S. Francisco, C. M. Hasenack, and V. Baranauskas. Si/SiO_2 electronic roughness: A consequence of Si/SiO_2 interface roughness. *Journal of The Electrochemical Society*, March 1996.

[Lin94] A. Linde. Scientific American, November 1994.

[LKK+94] H. P. Lang, J. Karpinski, E. Kaldis, J.-P. Ramseyer, and H.-J. Güntherodt. Atomic force microscopy study of the fractal shape of residual barium Cuprate/Copper oxide flux on the surface of $Y_2Ba_4Cu_7O_{15}$ and $YBa_2Cu_4O_8$ single crystals. *J. Vac. Sci. Technol. B*, 3(12):1614–1617, May/Jun 1994.

[Lu95] Chunsheng Lu. On the validity of the slit islands analysis in the measure of fractal dimension of fracture surfaces. *International Journal of Fracture*, 69:R77–R80, 1995.

[Lut94] S. M. Luthi. Textural segmentation of digital rock images into bedding units using texture energy and cluster labels. *Mathematical Geology*, 26(2):181–196, 1994.

[Mak94] A. Z. Maksymowicz. Fractal properties of a one-dimensional film surface. *Physical Review B*, 50(12):8901–8902, September 1994.

[Man82] Benoit B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freeman and Company, 1982.

[Mic90] Sun Microsystems. *OpenLook – Graphical Users Interface - Aplication Style Guidelines*. Addison-Wesley, Reading, MA, 1990.

[NNNS94] V. I. Nikora, O. I. Nikora, D. A. Noever, and G. M. Smart. New method of structural functions for analysing fractal scaling properties of natural processes. *J. Phys. A*, 27:L403–L409, 1994.

[Nye89] A. Nye. *XLib Programming Manual*, volume 1. O'Reilly & Associates, Inc., Sebastapol, CA, first edition, 1989.

[Pfe84] P. Pfeifer. Fractal dimension as working tool for surface-roughness problems. *Applications of Surface Science*, 18:147–164, 1984.

[PFTV89] W. H. Press, B. P. Flannery, S. A. Tenkolsky, and W. T. Vetterling. *Numerical Recipes in C - The Art of Scientific Computing*. Cambridge University Press, 1989.

[RMC94] M. V. H. Rao, B. K. Mathur, and K. L. Chopra. Evaluation of the scalling exponent of self-affine fractal surface from a single scanning probe microscope image. *Appl. Phys. Lett.*, 65(1):124–126, July 1994.

[SC94] N. Sarkar and B. B. Chaudhuri. An efficient differential box-counting approach to compute fractal dimension of image. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(1):115–120, January 1994.

[SI94] L. Spanos and E. A. Irene. Investigation of roughened silicon surfaces using fractal analysis i: Two-dimensional variation method. *J. Vac. Sci. Technol. A*, 5(12):2646–2652, Sep/Oct 1994.

[SL95] S. J. Sze and T. Y. Lee. Fractal structure of porous solids characterized by adsorption. *Physical Review B*, 51(14):8709–8714, April 1995.

[SLI+94] L. Spanos, Q. Liu, E. A. Irene, T. Zettler, B. Hornung, and J.J. Wortman. Investigation of roughened silicon surfaces using fractal analysis ii: Chemical etching, rapid thermal chemical vapor deposition, and thermal oxidation. *J. Vac. Sci. Technol. A*, 5(12):2653–2661, Sep/Oct 1994.

[Str86] B. Stroustrup. *The C++ Programing Language.* Addison-Wesley Publishing Company, 1986.

[TKW+94] D. P. Tsai, J. Kovacs, Z. Wang, M. Moskovits, V. M. Shalaev, J. S. Suh, and R. Botet. Photon scanning tunneling microscopy images of optical excitations of fractal metal colloid clusters. *Physical Review Letters*, 72(26):4149–4154, June 1994.

[VSO+94] L. Vázquez, R. C. Salvarezza, P. Ocón, P. Herrasti, J. M. Vara, and A. J. Arvia. Self-affine fractal electrodeposited gold surfaces: Characterizations by scanning tunneling microscopy. *Physical Review E*, 49(2):1507–1511, February 1994.