Universidade do Estado do Rio de Janeiro

Instituto Politécnico

Dalmo Stutz

ESTRATÉGIAS DE COMPUTAÇÃO PARALELA PARA A RESTAURAÇÃO DE IMAGENS COM O FUNCIONAL DE REGULARIZAÇÃO DE TIKHONOV

Nova Friburgo
2009

Dalmo Stutz

ESTRATÉGIAS DE COMPUTAÇÃO PARALELA PARA A RESTAURAÇÃO DE IMAGENS COM O FUNCIONAL DE REGULARIZAÇÃO DE TIKHONOV

Tese apresentada como requisito parcial para obtenção do título de Doutor, ao Programa de Pós-Graduação em Modelagem Computacional, do Instituto Politécnico, da Universidade do Estado do Rio de Janeiro.

Orientadores: Prof. Antônio José da Silva Neto
Prof. Ricardo Cordeiro de Farias

Nova Friburgo
2009

CATALOGAÇÃO NA FONTE
UERJ / REDE SIRIUS / BIBLIOTECA CTC/E

S586 Stutz, Dalmo
Estratégias de computação paralela para a restauração de imagens com o funcional de regularização de Tikhonov / Dalmo Stutz. - 2009.
203 f.: il.

Orientadores: Antônio José da Silva Neto e Ricardo Cordeiro de Farias
Tese (Doutorado em Modelagem Computacional) – Universidade do Estado do Rio de Janeiro, Instituto Politécnico.

1. Processamento de imagens – Teses. 2. Processamento paralelo (Computadores) - Teses. 3. Algoritmos – Teses. 4. Tikhonov, A. N. (Andrei Nikolaievich), 1906- – Teses. I. Silva Neto, Antônio José da. II. Farias, Ricardo Cordeiro de. III. Universidade do Estado do Rio de Janeiro. Instituto Politécnico. IV. Título.

CDU 004.932

Dalmo Stutz

Estratégias de Computação Paralela Para a Restauração de Imagens com o Funcional de Regularização de Tikhonov

Tese submetida ao corpo docente do Programa de Pós-Graduação em Modelagem Computacional do Instituto Politécnico, Campus Regional da UERJ em Nova Friburgo, como parte dos requisitos necessários à obtenção do grau de Doutor em Modelagem Computacional. Linha de Pesquisa: Matemática Aplicada e Computação Científica.

Orientadores: Prof. Antônio José da Silva Neto, Ph.D.

Prof. Ricardo Cordeiro de Farias, Ph.D,

Aprovado em 26 de outubro de 2009

Banca Examinadora:

Prof. Antônio José da Silva Neto, Ph.D.
IPRJ/UERJ

Prof. Ricardo Cordeiro de Farias, Ph.D.
UFRJ

Profª Cristiana Bentes, D.Sc.
UERJ

Prof. Geraldo Antônio Guerrera Cidade, D.Sc.
UFRJ

Prof. Luiz Marcos Garcia Gonçalves, D.Sc.
UFRN

Prof. Roberto Aizik Tenenbaum, D.Sc
IPRJ/UERJ

Dedico este trabalho à minha esposa

Kátia pelo seu amor, compreensão, carinho e apoio

e à minha filha

Danielle, razão da minha vida e que muito me orgulha.

“O fogo acende,
ele apaga,
ele esquenta,
ele é a paz e
outras coisas mais.”

Danielle C. Stutz

RESUMO

STUTZ, Dalmo. Estratégias de computação paralela para a restauração de imagens com o funcional de regularização de Tikhonov. 2009. 203 f. Tese (Doutorado em Modelagem Computacional) - Instituto Politécnico, Universidade do Estado do Rio de Janeiro, Nova Friburgo, 2009.

A Microscopia de Força Atômica é uma técnica que permite a aquisição de imagens em escalas nanométricas da superfície de quase todo tipo de material. Nessa escala, porém, as imagens podem apresentar uma relação sinal/ruído pobre, causado por efeitos degenerativos em sua qualidade. Para recuperar essas imagens ou minimizar os efeitos da degradação, técnicas de restauração de imagens são empregadas.

Nas últimas décadas, diversas técnicas têm sido desenvolvidas e aplicadas com essa finalidade. Dentre elas, uma técnica de restauração, descrita aqui nesta tese, baseada na minimização de um funcional de Tikhonov com termos de regularização a um parâmetro, tem sido usada há alguns anos com resultados bastante satisfatórios no tratamento de imagens obtidas com o Microscópio de Força Atômica.

O uso dessa técnica, entretanto, exige um grande esforço computacional que resulta em um tempo de execução elevado quando o programa que implementa o algoritmo de restauração é processado serialmente. Além disso, à medida que os equipamentos eletrônicos aumentam as suas capacidades, as imagens obtidas por esses equipamentos aumentam de resolução, assim como o esforço computacional e o tempo gasto para analisá-las e restaurá-las. Assim, com o passar do tempo, o aumento da velocidade de processamento e do desempenho do programa de restauração tem-se tornado um problema cada vez mais crítico.

Com o intuito de obter uma velocidade maior de processamento, nesta tese é descrita uma estratégia de implementação do algoritmo de restauração que faz uso de técnicas de computação paralela para se desenvolver uma nova versão paralela do programa de restauração.

Os resultados obtidos com essa nova versão do programa mostram que a estratégia paralela adotada reduziu os tempos de execução e produziu bons desempenhos computacionais quando comparado com outras implementações feitas do mesmo algoritmo. Além disso, a nova estratégia apresenta níveis de desempenho maiores à medida que as resoluções das imagens restauradas aumentam, possibilitando a restauração de imagens maiores num tempo proporcionalmente mais curto.

Palavras-chave: Processamento de imagens. Processamento paralelo (Computadores). Algoritmos. Tikhonov, A. N. (Andrei Nikolaievich), 1906-.

ABSTRACT

The Atomic Force Microscopy technique allows the nanoscale imaging of almost any type of surface. However, in this scale, the images obtained with this technique can present either poor signal/noise ratios or blurred contents. In order to restore or, at least, decrease the effects of the degradations, image restoration techniques are employed.

In the last decades, many techniques have been developed and applied for such purpose, including the restoration technique based on the minimization of Tikhonov's regularization functional, described in this thesis. In recent years, this technique has been used to restore AFM´s image with very satisfactory results.

The application of this technique requires, however, a lot of effort and computational power that result on long run-times when processed serially. Moreover, as the electronic equipments have increased their capabilities, the images have increased in resolution as well as the time and the effort for their processing. In the course of time, the processing speedup and the performance improvement have become a critical problem.

To solve this, techniques of parallel computation are applied to obtain greater processing speedups. In this thesis, a parallel implementation strategy for the restoration algorithm is described and used to develop a new parallel version of the restoration computational program.

The results obtained so far show that the parallel strategy adopted reduced the run times and produced better performances in comparison to the other previous implementations of the same algorithm. Moreover, the new strategy yields higher levels of performance as the restored image resolution increase, making possible the restoration of larger images in time periods relatively shorter.

Keywords: Image processing. Parallel processing (Electronic computers). Algorithms. Tikhonov, A. N. (Andrei Nikolaievich), 1906-.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Apresentação

A Microscopia de Força Atômica (Binnig et al., 1986) é uma tecnologia que permite a aquisição de imagens em escala nanométrica de quase todo tipo de superfície e materiais. Atualmente, diversas aplicações em engenharia e medicina têm mostrado que o estado de arte nessa área encontra-se na aquisição e no tratamento dessas imagens (Cidade et al., 2003).

Nesta escala, os efeitos degenerativos na qualidade das imagens são bastante significativos. Normalmente, as imagens obtidas por um Microscópio de Força Atômica (AFM – *Atomic Force Microscope*) apresentam uma relação sinal/ruído pobre, causados por efeitos degenerativos em sua qualidade (degradação), ocasionados tanto por borramento quanto pelo ruído aditivo proveniente da interação entre a ponteira do microscópio e a amostra que está sendo analisada, bem como da eletrônica do próprio equipamento.

Para tratar ou minimizar os efeitos degradantes, faz-se necessário a aplicação de um pós-processamento dessas imagens. Nos últimos anos, diversas técnicas têm sido desenvolvidas e aplicadas com o objetivo de melhorar algum aspecto e/ou recuperar uma imagem, que tenha sofrido algum tipo de degradação: borramento, ruído aditivo etc. Dentre essas técnicas, podemos citar, por exemplo, a transformada rápida de Fourier (FFT) e a morfologia matemática. No primeiro, faz-se um corte espectral que pode levar a uma excessiva suavização da imagem obtida pelo AFM, eliminando parte da informação relevante junto com o ruído de alta frequência. Enquanto que, no segundo, podem se ressaltar os efeitos do ruído experimental presente na imagem obtida com o microscópio.

Visando recuperar e/ou minimizar os efeitos de degradação das imagens obtidas pelo AFM, a aplicação de uma técnica usando uma abordagem com problema inverso, na qual se emprega o funcional de Tikhonov com os termos de regularização formados por uma família de funções de regularização a um parâmetro, vem sendo usada já há algum tempo no tratamento de imagens obtidas com AFM, com resultados bastante satisfatórios (Cidade et al., 2000, 2003).

Entretanto, o processo de restauração de imagens usando essa técnica exige um esforço computacional muito grande que acaba resultando em um tempo de execução elevado quando o programa que implementa o algoritmo de restauração é processado serialmente. Isso pode tornar o seu uso não muito indicado em aplicações comerciais ou em processos cujas

atividades demandem respostas rápidas (p.ex. geração de vídeos produzidos a partir de imagens capturadas pelo AFM).

Um outro aspecto importante que também deve ser ressaltado é que à medida que os equipamentos eletrônicos vão aumentando as suas capacidades, as imagens obtidas por eles vão aumentam de resolução, assim como o esforço computacional para analisá-las e restaurá-las.

Com o passar do tempo, o aumento da velocidade de processamento e do desempenho do programa de restauração tem se tornado cada vez mais crítica. Buscar soluções para o problema que reduzam os tempos computacionais e o desenvolvimento de uma estratégia de implementação usando computação paralela é o caminho natural para se resolver este problema.

Com o intuito de obter uma velocidade maior de processamento, Stutz (2004) em seu trabalho de dissertação de mestrado usou técnicas de computação paralela no desenvolvimento de uma estratégia para a implementação de um programa paralelo do algoritmo de restauração de imagens baseado no funcional de regularização de Tikhonov (Cidade, 2000).

Apesar da eficácia dessa estratégia e do programa paralelo de restauração ter alcançado tempos computacionais menores do que aqueles obtidos com o programa serial, descobriu-se mais tarde que o desempenho e a eficiência do programa decaíam e que os custos operacionais aumentavam significativamente à medida que novas unidades de processamento iam sendo acrescentados ao processamento em paralelo. Em outras palavras, a estratégia proposta não era escalável.

Investigando-se melhor as causas desse problema, descobriu-se que a quantidade total de dados distribuída às unidades de processamento aumentava consideravelmente a cada processador incluído no processo de restauração em paralelo. Esse aumento atribuía uma sobrecarga adicional de trabalho a cada processador e ao processamento em paralelo como um todo, que interferia negativamente nos índices de desempenho nessa versão do programa paralelo.

Além disso, as imagens restauradas por esse programa apresentavam algumas diferenças em relação à mesma imagem restaurada serialmente. Essa diferença aparecia na forma de artefatos (efeito-borda) presentes na imagem processada que surgiam após uma restauração em paralelo. Uma solução para esse problema foi criada de modo a reduzir a

incidência do efeito-borda, porém a sua aplicação não elimina completamente esse efeito, apenas a minimiza.

Com o objetivo de melhorar e aprimorar o programa de restauração em paralelo, estudou-se melhor o problema e desenvolveu-se uma nova estratégia de implementação paralela do algoritmo de restauração. Como resultado desse processo, criou-se uma nova versão paralela do programa que resolveu os problemas detectados na versão paralela anterior, alcançou melhores desempenhos computacionais e produziu tempos de execução menores do que aqueles obtidos até então.

- ***Trabalhos recentes***

Os Microscópios de Varredura por Sonda são uma família de microscópios, utilizados para se observar localmente diversas propriedades da superfície de uma amostra em uma escala nanométrica. Dessa família de equipamentos, os mais utilizados são o Microscópio de Força Atômica e o Microscópio de Tunelamento (Chinaglia, 2002). Desde a sua criação (Binnig et al., 1986), a tecnologia do AFM tem sido objeto de estudos em vários grupos de pesquisa espalhados por todo o mundo, com os mais variados interesses e nas mais variadas áreas do conhecimento.

No campo de processamento de imagens, Ribeiro et al. (2001) e Ribeiro e Silva (2003) investigaram a aplicação de Algoritmos Genéticos (GA), enquanto, Bonnet et al. (1994) e Wilson et al. (1995) pesquisaram a Morfologia Matemática (MM), buscando através da aplicação dessas abordagens no processo de restauração, melhorias na qualidade das imagens obtidas pelo AFM. Vários grupos de pesquisa do Instituto Politécnico/UERJ (IPRJ/UERJ), em parceria com a equipe do Instituto de Biofísica Carlos Chagas Filho da Universidade Federal do Rio de Janeiro (IBCCF/UFRJ), têm trabalhando na melhoria do algoritmo de restauração de imagens de AFM, desenvolvido por Cidade (2000). Dentre eles, podemos citar: Furtado (2002) que propôs uma técnica de realimentação na restauração das imagens de AFM, que acelerava a convergência do método de restauração; Stutz (2004) que acelerou o processamento com a implementação de um programa paralelo do algoritmo de restauração, usando técnicas de decomposição do domínio; Fonseca (2004) que implementou um método de otimização, utilizando algoritmos genéticos; Gil (2005) que implementou um método de restauração de imagens de AFM, utilizando redes neurais artificiais; Terração (2005) que implementou um programa de restauração de imagens de AFM, usando morfologia

matemática; Romualdo (2006) que avaliou diferentes indicadores de qualidade para algoritmo de restauração e Santos (2008) que avaliou a aplicação da técnica de restauração na análise de superfícies metálicas; Recentemente, Almeida et al. (2009) apresentaram uma nova proposta, dentro da mesma linha de pesquisa, onde placas gráficas ou GPUs (*Graphics Processing Unit*) são empregadas no processo de restauração em paralelo de imagens de AFM.

Na área de processamento paralelo aplicado a restaurações de imagens, Li et al. (2000) apresentam um algoritmo paralelo para a restauração de imagens e vídeos que sofreram perdas ao serem comprimidos pelos padrões JPEG e MPEG; Fernández et al. (2000) fazem a paralelização de um algoritmo de restauração, baseado no método máximo a posteriori (MAP - *Maximum a Posteriori*) para estimar a imagem original, a partir de imagens que tenham sofrido ruído aditivo do tipo gaussiano; Bevilacqua e Picollomini (2000) apresentam um algoritmo baseado em paralelismo de dados, usando decomposição do domínio adaptativa, para a restauração de uma classe de degradações de imagens feitas em paralelo num cluster; Picollomini e Zama (2001) descrevem um algoritmo paralelo de restauração de imagens quando a função de espalhamento pontual é variante para ser aplicado no processamento de imagens astronômicas ou aéreas que tenham sofrido aberrações óticas, um movimento relativo de câmera ou sofrido degradações devido a turbulências atmosféricas; Almeida et al. (2006) fazem referência a novo método, ainda em desenvolvimento, para ser aplicado na restauração imagens astronômicas em paralelo, usando decomposição do domínio numa solução híbrida que mescla memória compartilhada (OpenMP) e passagem de mensagem (MPI); Gordon (2006) faz uso de processadores matriciais para a reconstrução em paralelo de imagens de tomografia computadorizada empregando uma técnica de reconstrução algébrica conhecida com ART (*Algebraic Reconstruction Technique*); Dongdong et al. (2007) desenvolvem pesquisas em arquiteturas paralelas de computador que melhor se aplicam as três categorias de sistemas de processamento paralelo de imagens; entre outros trabalhos.

1.2 Objetivos

Com o intuito de resolver os problemas detectados na versão anterior do programa paralelo de restauração e dar continuidade ao trabalho que já foi realizado na dissertação (Stutz, 2004), esta tese tem como objetivo principal empregar técnicas de computação

paralela no desenvolvimento de uma estratégia de implementação de um novo programa paralelo de restauração que:

a) Reduza as sobrecargas (*overheads*) da estratégia paralela anterior;
b) Reduza os tempos de execução do programa paralelo de restauração;
c) Melhore as medidas de desempenho computacional em relação às estratégias que já foram empregadas;
d) Elimine o efeito-borda que aparece nas imagens restauradas pela versão paralela anterior do programa de restauração;
e) Permita que imagens de resoluções maiores possam ser restauradas em paralelo usando mais processadores, sem com isso introduzir sobrecargas extras e desnecessárias no processamento paralelo que prejudique e/ou inviabilize o uso do programa de restauração com mais de um processador.

Para verificar se a nova estratégia computacional paralela atinge os requisitos expostos acima (validação), este trabalho também tem como objetivo implementar e testar uma nova versão do programa paralelo de restauração baseado na nova estratégia.

Como propósito secundário, busca-se uma nova versão paralela do programa de restauração que possa algum dia ser embutido em um pacote de softwares para ser embarcado em um equipamento dedicado de processamento de imagens em tempo real (ou cujo processamento seja o mais rápido possível), de modo a torná-lo viável comercialmente.

Além disso, busca-se nesta tese produzir um material escrito que reúna em um mesmo lugar assuntos relacionados ao tema abordado para que possa servir de base de estudos para trabalhos futuros que se seguirão após a conclusão deste trabalho.

1.3 Organização deste material

Essa tese está dividida em sete partes. Na primeira parte, **Capítulo 1**, é feita uma introdução sobre os assuntos abordados por todo o material, os objetivos do trabalho, outras contribuições e um resumo de alguns trabalhos e estudos feitos recentemente dentro da mesma linha de pesquisa aqui abordada.

Na segunda parte, composta pelos Capítulos 2 e 3, são apresentados alguns conceitos básicos que serão empregados em quase todo este material. No **Capítulo 2**, é descrito as características de uma imagem digital monocromática, como ocorre o processo de formação de imagens, o que é microscopia de força atômica, como uma imagem é obtida através de um microscópio de Força Atômica (AFM), a iteração ponteira-amostra e as características das imagens obtidas por esse equipamento etc. No **Capítulo 3**, são apresentados alguns conceitos, paradigmas e terminologias da computação paralela, decomposição de domínio, fatores críticos para a queda de desempenhos dos programas paralelos etc.

Na terceira parte, **Capítulo 4**, o problema de restauração de imagens é formulado. Nele são descritos: o modelo de degradação, o problema de restauração de imagens, o algoritmo de convolução discreta, o operador de borramento, funções de regularização, a técnica de restauração baseado na minimização do funcional de regularização de Tikhonov, o fluxograma do algoritmo de restauração, critérios de parada, o parâmetro de regularização α e algumas técnicas para obtê-lo, o tratamento dos pontos próximos aos limites da imagem, a técnica de realimentação etc.

Na quarta parte, no **Capítulo 5**, são apresentadas as métricas de avaliação que serão usadas para se medir o desempenho computacional em paralelo e a qualidade das restaurações. Nesse capítulo é apresentada uma nova métrica de avaliação da qualidade de imagens: o IWMSE, que difere dos métodos convencionais baseados em critérios objetivos apoiados em erros estatísticos (p.ex. MSE, SNR etc.) e que fornece uma medida de diferenças, usando critérios subjetivos baseados nas características do sistema visual, que representam a forma como observamos e percebemos as diferenças entre duas imagens.

Na quinta parte, que abrange os Capítulos 6 e 7, são descritas as estratégias de implementação aplicadas ao algoritmo de restauração. No **Capítulo 6**, são descritas as estratégias anteriores de implementação do programa de restauração. Nele são feitos comentários a respeito do programa serial, descreve-se resumidamente a estratégia paralela do programa de restauração desenvolvido anteriormente (Stutz, 2004), o efeito-borda, a solução adotada para minimizá-lo e o problema da queda de desempenho apresentado por esta estratégia.

No **Capítulo 7** é descrita uma nova estratégia de computação paralela usada na implementação do programa paralelo de restauração, o processo de otimização e os refinamentos aplicados a esta nova estratégia etc. Nesse capítulo, descreve-se os termos e convenções adotadas, o detalhamento da estratégia de execução da sequência de cálculos dos

termos em paralelo, o problema de conflito de dados causado pelo uso do método de Gauss-Seidel, a solução do problema usando o método iterativo de Jacobi, a avaliação dos resultados dessa solução, redução dos tempos de comunicação, a aplicação da técnica *message prefetching* etc.

O **Capítulo 8**, sexta parte, consiste na avaliação de desempenho e da qualidade dos resultados obtidos com novo o programa paralelo de restauração baseado na nova estratégia de implementação e discussões finais. Aqui se encontram a metodologia de avaliação, a arquitetura empregada, gráficos com curvas de desempenho dos programas serial e paralelos (versão nova e anterior), tabelas comparativas de qualidades de restauração e comentários a respeito desses resultados.

Na sétima e última parte, o material é finalizado com as conclusões (**Capítulo 9**), referências bibliográficas, glossário e apêndices.

1.4 Contribuições desta tese

A principal contribuição desta tese é a elaboração de uma estratégia de implementação, que faz uso de técnicas de computacionais, para o desenvolvimento de uma versão paralela do programa que implementa o algoritmo de restauração de imagens de AFM apresentado por Cidade (2000) e atinge um rendimento computacional maior do que àqueles obtidos, até então, com outras implementações do mesmo programa.

Como resultado disso, foi desenvolvida uma nova versão paralela do programa de restauração que alcança tempos de execução menores, apresenta níveis de desempenho melhores do que de outras implementações e, ainda, resolve o problema de sobrecarga (*overhead*) e o aparecimento de artefatos na imagem restaurada que foram detectados em uma implementação anterior do programa paralelo de restauração (Stutz, 2004).

Além da estratégia de implementação e a nova versão paralela do programa de restauração, outras contribuições, algumas delas inovadoras, relacionadas com outros assuntos discutidos neste material, são descritas ou referenciadas ao longo desta tese e são apresentadas resumidamente, a seguir:

a) **IWMSE (*Information Weighted Mean Square Error*)** – uma nova métrica de comparação de imagens que fornece uma medida de diferenças, usando critérios

subjetivos baseados nas características do sistema visual e que representam a forma como percebemos as diferenças entre duas imagens (Stutz et al., 2007);

b) **Processo iterativo de busca do parâmetro de regularização** α - um novo método iterativo de busca do parâmetro de regularização ótimo em um problema de restauração de imagens (Stutz et al., 2008, 2009). Método desenvolvido para ser empregado quando certos limites do problema de restauração não são conhecidos;

c) Uso do método iterativo de Jacobi na solução do problema de conflito de dados causado pelo método de Gauss-Seidel - modificação na técnica original de restauração para resolver o problema de conflito de dados que ocorre quando o método iterativo de Gauss-Seidel é empregado na solução em paralelo de um sistema linear. Apesar de ser um procedimento muito comum em computação paralela, descobriu-se que o método iterativo de Jacobi, quando aplicado à técnica de restauração, produz restaurações qualitativamente melhores do que quando se emprega Gauss-Seidel. Como resultado, foi desenvolvido uma nova versão serial do programa de restauração onde o método iterativo de Jacobi é aplicado e testado;

d) Adaptações na técnica descrita por Cidade (2000) de forma a incluir o parâmetro de relaxação (SOR/JOR) no algoritmo de restauração – essa modificação amplia a técnica e abre possibilidades para que novos trabalhos explorem o uso da relaxação no processo de restauração.

2 CONCEITOS FUNDAMENTAIS

2.1 Imagens digitais monocromáticas

Uma imagem digital monocromática é uma função bidimensional $f(u,v)$ de intensidades discretas de luz, onde o valor ou amplitude de f nas coordenadas espaciais discretas (u,v) representa a intensidade (ou brilho) de uma pequena região contínua da imagem naquele ponto (Fig. 2.1) (Gomes e Velho, 1994).

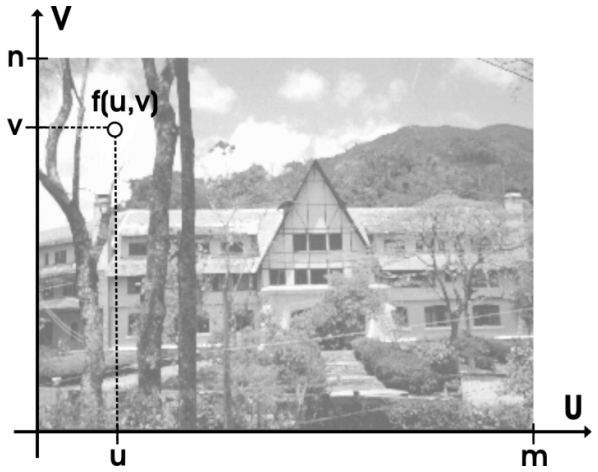


Figura 2.1 - Imagem digital formada pela variação de intensidades discretas de luz, mostrando em destaque uma representação generalizada do ponto (u,v).

A luz, entretanto, não é o único parâmetro usado na formação de imagens digitais. Dependendo da aplicação e/ou da área de interesse, uma imagem digital pode ser formada, também, pela variação de intensidades de temperatura, pressão, topologia, emissão de raios-X, absorção de energia etc. Normalmente, essas imagens são convertidas em intensidades de luz impressas sobre um plano bidimensional, tal como numa foto, para que possam ser visualizadas.

Assim, para se gerar uma imagem digital, f deve ser digitalizada, fazendo-se uma amostragem de $f(u,v)$ nas direções U e V e na amplitude $z = f(u,v)$. Isso resulta em uma matriz de $m \times n$ amostras ou pontos, Eq. (1), também conhecidos como pixels ou pels (do inglês, *picture elements*), convertidos em intensidades de luz e quantizados em $L_{\max}$ níveis de cores ou conjunto de cores, denominado espaço de cores.

$$f(u,v) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,m-1) \\ f(1,0) & f(1,1) & \cdots & f(1,m-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(n-1,0) & f(n-1,1) & \cdots & f(n-1,m-1) \end{bmatrix}_{m \times n} \quad (1)$$

Quando estes valores variam numa escala de inteiros de 0 a 255 ($L_{\max} = 256$), diz-se que a imagem é monocromática (escala de cinza), onde o valor 0 (zero) representa a menor intensidade de luz (ou a cor preta), 255 a maior intensidade (ou o branco) e os demais valores intermediários as diversas tonalidades de cinza (Fig. 2.2b).

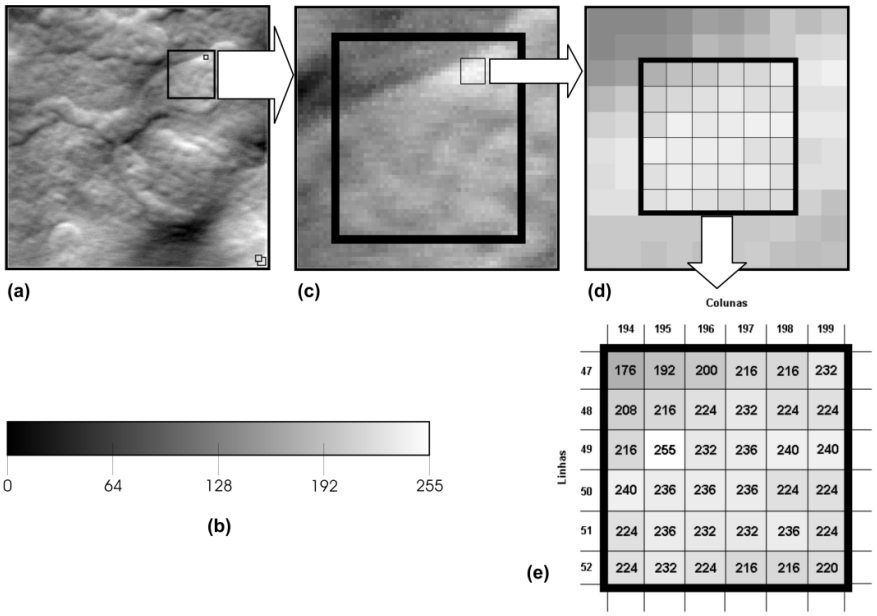


Figura 2.2 - Estrutura de uma imagem digital monocromática. (a) Imagem biológica de AFM de 256x256 pixels de um eritroblasto em estado leucêmico de 1000 nm x 1000 nm numa (b) escala de cor monocromática. Projeções (c) e (d) das áreas demarcadas na imagem, (e) a representação matricial dos pontos (pixels) de uma sessão da imagem representada em (d).

2.2 Processo de formação de imagens

Conforme ilustrado na Fig. 2.3, assumindo-se que exista um objeto x num sistema de coordenadas (k,l), denominado **plano objeto**, iluminado por uma fonte de energia radiativa e/ou sendo ele próprio a fonte dessa radiação, tem-se que a energia irradiada por esse objeto se propaga através do espaço até ser captada por um sistema de formação de imagens que

transforma essa energia em uma imagem y num sistema de coordenadas (i, j), conhecido como **plano imagem**.

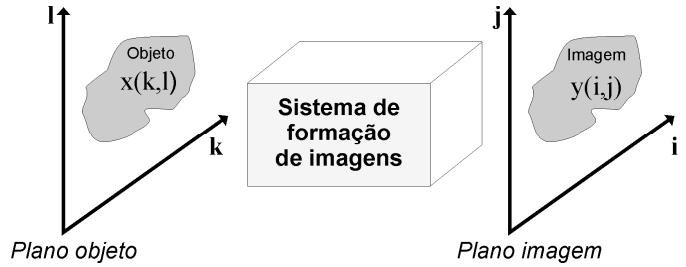


Figura 2.3 - Diagrama esquemático do sistema de formação de imagens (Andrews e Hunt, 1977).

- ***Função de espalhamento pontual***

Fazendo um paralelo entre os pontos (k,l) no plano objeto e (i, j) no plano imagem, tem-se que o sistema de formação de imagens gera o ponto (i, j) através da energia irradiada e captada do objeto x. No entanto, esse sistema recebe componentes de energia não só de (k,l), mas também, de todos os pontos do plano objeto. Visto que, normalmente, nos processos de transporte, a energia irradiada é acumulativa (ou seja, aditiva), isto significa que no processo de formação de imagens a composição do ponto (i, j) sofre influências tanto do ponto (k,l) como, também, dos outros pontos do objeto situados ao redor de (k,l). A intensidade dessa influência, entretanto, varia de acordo com a distância entre o ponto (k,l) e os demais pontos no plano objeto. À medida que essa distância aumenta, a contribuição desses pontos do objeto para a formação de (i, j) diminui. Essa superposição de distribuições de energia pode ser representada por uma função h, conhecida como Função de Espalhamento Pontual (*PSF - Point Spread Function*), que descreve a transformação da energia do plano objeto para o plano imagem, dada por

$$y(i, j) = h(i, j, k, l, x(k,l)) \ . \tag{2}$$

Dado que o menor nível de energia é zero (não-negatividade), tem-se também que

$$x(k,l) \geq 0 \tag{3}$$

e

$$y(i, j) \geq 0 . \tag{4}$$

- ***Equação geral***

Considerando que os pontos no objeto sejam contínuos e que a distribuição de energia no plano imagem seja aditiva, somando-se as devidas contribuições infinitesimais de todos os pontos do objeto, obtém-se a equação geral de formação de imagem dada por (Andrews e Hunt, 1977)

$$y(i, j) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(i, j, k, l, x(k, l))\, dk\, dl . \tag{5}$$

- ***Sistemas lineares e não-lineares***

Apesar da distribuição de energia em ambos os planos (objeto e imagem) ser cumulativa, o comportamento do sistema de formação de imagens pode não necessariamente relacionar os componentes aditivos de um plano ao outro. Sistemas com essa característica são classificados como não-lineares e apresentam a seguinte relação

$$\begin{aligned} y_1(i, j) + y_2(i, j) &= h(i, j, k, l, x_1(k, l)) + h(i, j, k, l, x_2(k, l)) \\ &\neq h(i, j, k, l, x_1(k, l) + x_2(k, l)) \end{aligned} \tag{6}$$

onde $x_1(k,l)$ e $x_2(k,l)$ são dois pontos quaisquer no plano objeto e $y_1(i, j)$ e $y_2(i, j)$ os pontos correspondentes no plano imagem.

Entretanto, se a igualdade se mantém na última relação da Eq. (6), então o sistema de formação de imagens passa a ser linear. Isto significa que certas propriedades matemáticas se aplicam e que o sistema obedece a dois princípios básicos: aditividade e homogeneidade.

Pela aditividade, tem-se que se a função h é linear, então a resposta do sistema a uma soma de duas entradas é igual à soma das duas respostas (Gonzalez e Woods, 2000)

$$h(i, j, k, l, x_1(k,l) + x_2(k,l)) = h(i, j, k, l, x_1(k,l)) + h(i, j, k, l, x_2(k,l)). \quad (7)$$

Já pela homogeneidade, diz-se que a resposta a um múltiplo constante de uma entrada é igual à resposta daquela entrada multiplicada por essa constante (Gonzalez e Woods, 2000), ou seja,

$$h(i, j, k, l, c_1\, x(k,l)) = c_1\, h(i, j, k, l, x(k,l)) \quad (8)$$

onde c_1 é uma constante qualquer.

Aplicando-se a propriedade da homogeneidade, para o caso em que o sistema de formação é linear, reescreve-se a Eq. (5) como

$$y(i, j) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(i, j, k, l)\, x(k,l)\, dk\, dl \quad (9)$$

- ***Sistemas espaço-variante e invariante***

A Eq. (9) é conhecida como integral de Fredholm com quatro dimensões e o emprego do operador h com quatro as coordenadas, Eqs. (5) e (9), é a descrição mais geral da PSF, já que ela permite variar a posição em ambos os planos (objeto e imagem). Quando descrita dessa forma, a função é conhecida como Função de Espalhamento Pontual Espaço-Variante (*SVPSF – Space-Variante PSF*).

Entretanto, se o sistema de formação de imagens agir uniformemente em ambos os planos, então o operador h é conhecido como Função de Espalhamento Pontual Espaço-Invariante (*SIPSF – Space-Invariante PSF*). Nesse caso, como a função é independente do ponto de visão, ou seja, independente das coordenadas em ambos os planos, a saída do sistema é dada em função das diferenças entre os sistemas de coordenadas e a equação do sistema de formação de imagens espaço-invariante, para o caso não-linear, é dada por

$$y(i, j) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(i-k, j-l, x(k,l))\, dk\, dl \quad (10)$$

e, para o caso linear,

$$y(i,j) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(i-k, j-l)x(k,l)\,dk\,dl\,. \tag{11}$$

A Eq. (11) é conhecida na literatura como convolução bidimensional e emprega uma notação do tipo contínuo-contínuo. Apesar desse modelo ser útil na análise de muitas técnicas de restauração, provavelmente, o modelo mais significativo e mais próximo da realidade é o discreto-contínuo.

- ***Sistema discreto-contínuo***

O modelo discreto-contínuo é de interesse prático, já que o objeto original é, usualmente, contínuo e os elementos que compõem o sistema de formação de imagens (sensores, processos de amostragem, operações matriciais, processos de exibição etc.) são discretos. Para o modelo discreto-contínuo, a equação de formação de imagem similar à Eq. (11) é dada por

$$y_{ij} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_{ij}(k,l)x(k,l)\,dk\,dl \tag{12}$$

onde $h_{ij}(k,l)$ é uma função dependente das variáveis (k,l) do plano objeto contínuo e (i, j) do plano imagem discreto.

Usando uma notação matricial, reescreve-se a Eq. (12) da seguinte forma:

$$Y = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} H(k,l)x(k,l)\,dk\,dl \tag{13}$$

onde $H(k,l)$ é uma matriz cujos elementos são funções a duas variáveis de coordenadas (k,l).

Admitindo-se que a matriz Y possui dimensão quadrada $n \times n$, reordenada lexicograficamente em um vetor $n^2 \times 1$, reescreve-se a Eq. (13) como

$$y = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(k,l)x(k,l)\,dk\,dl\,, \tag{14}$$

onde y e $h(k,l)$ são vetores de dimensão $n^2 \times 1$, sendo o primeiro composto por valores escalares e o segundo por funções do plano objeto de coordenadas (k,l).

- ***Sistemas discretos***

Apesar de ser mais realístico, o modelo discreto-contínuo não é o mais apropriado para ser implementado em num computador, uma vez que somente aproximações do objeto são possíveis. Além do mais, se precisarmos empregar álgebra linear e/ou alguma técnica de análise numérica, o modelo discreto-contínuo não seria muito útil. Para tal, seria necessário que o modelo fosse dado em um sistema discreto-discreto. Nesse caso, usando uma notação discreta, a Eq. (11) reduz-se a

$$y_{ij} = \sum_{k=0}^{n-1}\sum_{l=0}^{n-1} h_{i-k,j-l}x_{k,l}\,, \tag{15}$$

ou, simplesmente,

$$y = h * x\,, \tag{16}$$

onde o símbolo (*) representa o operador de convolução (discreta, no caso), y e x são respectivamente as imagens do plano objeto e do plano imagem, armazenados em vetores $n^2 \times 1$, lexicograficamente ordenados.

2.3 Microscopia de Força Atômica

A Microscopia por Varredura de Sonda (*SPM – Scanning Probe Microscopy*) é um termo genérico para descrever uma família de microscópios (Fig. 2.4) capazes de obter imagens com resoluções nanométricas ($1\text{nm} = 1\times10^{-9}\,\text{m}$), próximas da escala atômica. De um modo geral, esses equipamentos empregam uma sonda, que varre a superfície de uma amostra, e o sinal resultante da iteração sonda-amostra é percebido por um sensor e interpretado de modo a formar uma imagem do objeto analisado (Manheimer, 2002. Chinaglia, 2002).

A microscopia de Força Atômica (*AFM - Atomic Force Microscopy*), descrita por Binnig et al. (1986), também conhecida como Microscopia de Varredura de Força (*SFM - Scanning Force Microscopy*), é uma derivação da técnica da Microscopia de Varredura por Tunelamento (*STM - Scanning Tunneling Microscopy*) (Hansma et al., 1988) capaz de obter imagens, em escala nanométrica, da superfície de quase todo tipo de material, inclusive de amostras não-condutoras (polímeros, cerâmicas, amostras biológicas etc.) (Binnig et al., 1986), com grande utilidade e aplicação em estudos citológicos e histológicos (Kasas et al., 1993; Mariani et al., 1994). O AFM e técnicas correlatas formam uma família de instrumentos de varredura por sonda.

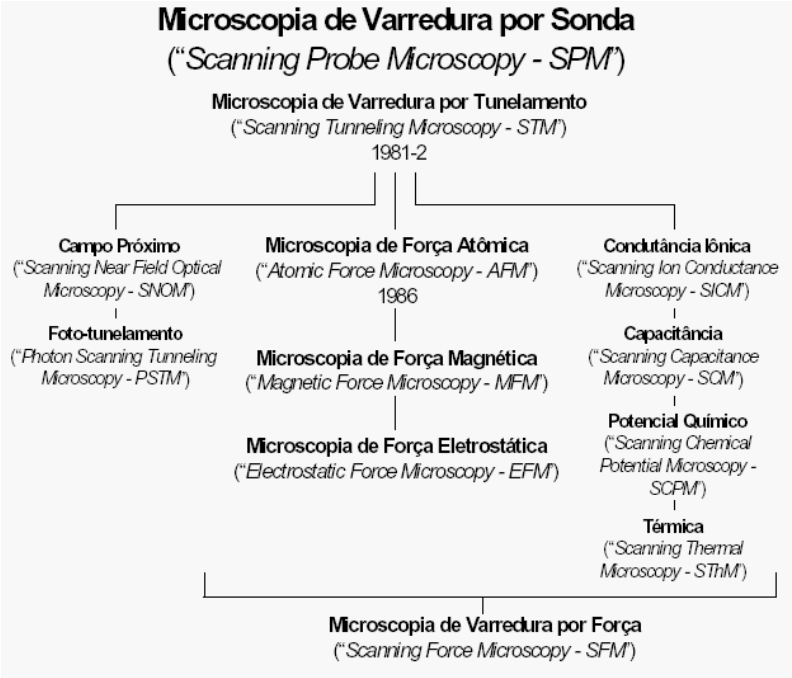


Figura 2.4 - Família de microscópios que utilizam a técnica de varredura por sonda (*Scanning Probe Microscope* - SPM) (Cidade et al., 2000)

- ***Processo de obtenção de imagens através de AFM***

De forma simplificada, a Fig. 2.5 ilustra esquematicamente o princípio de funcionamento de um AFM. Uma sonda de dimensões reduzidas (*cantilever*), com uma ponta em formato piramidal (*tip*, Fig. 2.6) em sua extremidade, varre a superfície da amostra que está sendo analisada, produzindo, ao final do processo, uma imagem que se compõe do registro sucessivo de diversos elementos individuais da amostra.

Durante o processo de varredura, a sonda exerce sobre a amostra uma força de ordem interatômica. Devido, predominantemente, às forças de van der Waals, a ponteira interage com a superfície da amostra, gerando, em função do relevo, uma resposta mecânica que flexiona a sonda. Um feixe de laser, incidindo sobre ela e refletido sobre um fotodetector, produz uma resposta elétrica que registra as deflexões, gerando um sinal captado e processado por um computador que cria uma representação gráfica da imagem da superfície da amostra naquele ponto. Uma plataforma piezoelétrica, que realiza movimentos simultâneos nas direções x e y, varre toda a amostra e, a cada registro, o sistema a reposiciona na direção z quase que de imediato, protegendo a superfície contra quaisquer riscos de ser atingida pela sonda. Um circuito integrador amortiza a velocidade de reposicionamento da plataforma, de forma a evitar a ocorrência de oscilações indesejáveis que possam prejudicar a qualidade das imagens produzidas pelo AFM. (Mannheimer, 2002; Cidade et al., 2003).

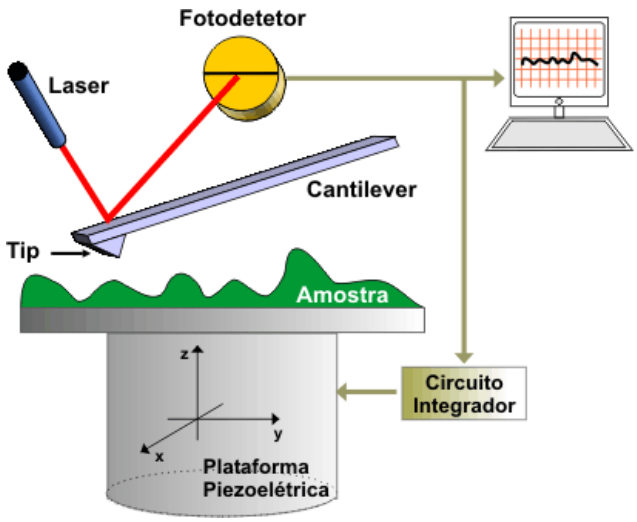


Figura 2.5 - Esquema simplificado de funcionamento do AFM.

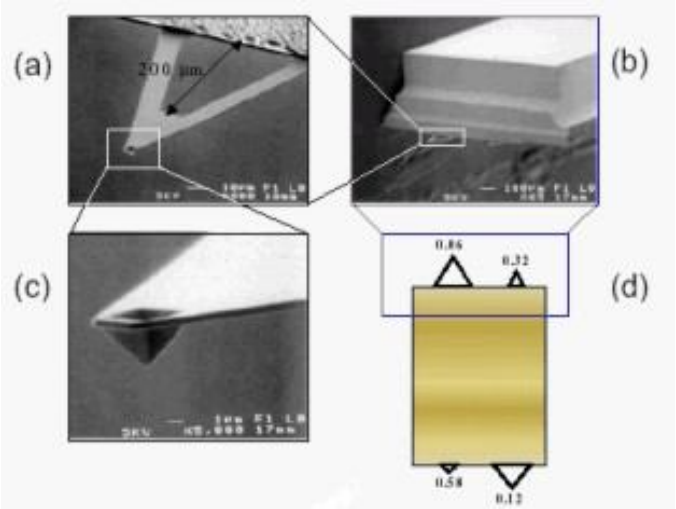


Figura 2.6 - (a) Cantilever convencional empregado em AFM, (d e b) parte de um conjunto de ponteiras com diferentes constantes de mola, (c) contendo em sua extremidade uma ponteira de dimensões nanométricas. (Cidade et al., 2000).

A forma de iteração ponteira-amostra depende do modo de funcionamento do microscópio, que pode operar em três regimes diferentes:

a) Contato: neste modo, a sonda é mantida a uma certa distância de décimos de nm da superfície da amostra com a ponteira, essencialmente, em contato físico com a amostra. A força atuante é fortemente repulsiva;
b) Não-contato: a ponteira se mantém distanciada da amostra (dezenas de nm) com o objetivo de preservá-la e, devido a este distanciamento, predominam as forças de van der Waals com caráter atrativo;
c) Intermitente (*tapping mode*): o cantilever vibra numa frequência de ressonância (normalmente, algumas dezenas de kHz), reunindo vantagens dos modos anteriores.

Na Fig. 2.7 pode-se observar o comportamento da interação ponteira-amostra e a localização das regiões de operação para cada regime.

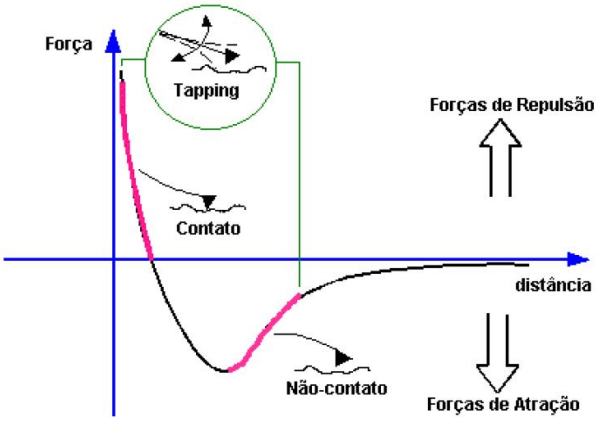


Figura 2.7 - Curva força versus distância, obtida a partir da interação ponteira-amostra (Cidade et al., 2003).

- ***Interação ponteira-amostra***

A interpretação do sinal resultante da interação ponteira-amostra para a formação da imagem é muito dependente das características da ponteira utilizada no microscópio e influi diretamente na percepção dos detalhes do objeto, podendo até ocorrer o surgimento de artefatos na imagem. Conforme ilustrado na Fig. 2.8, diferentes geometrias da ponteira produzem alterações significativas nas imagens obtidas pelo AFM para uma mesma amostra.

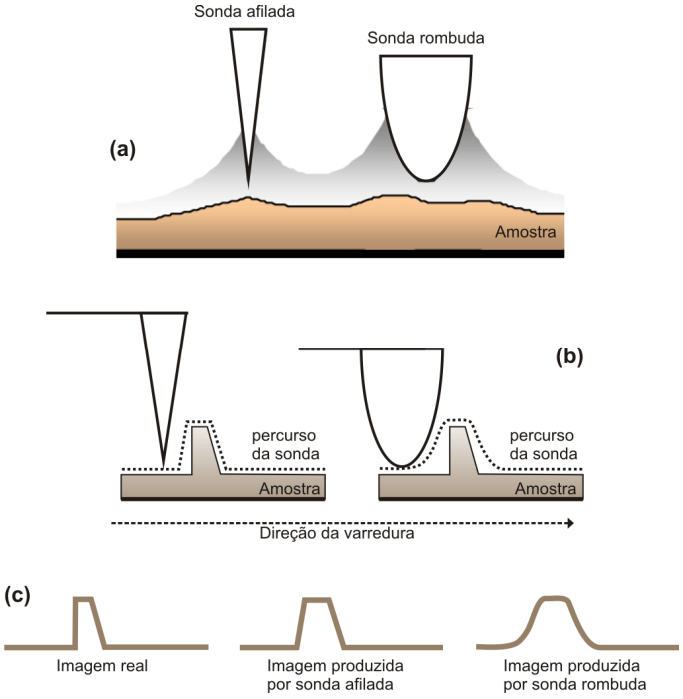
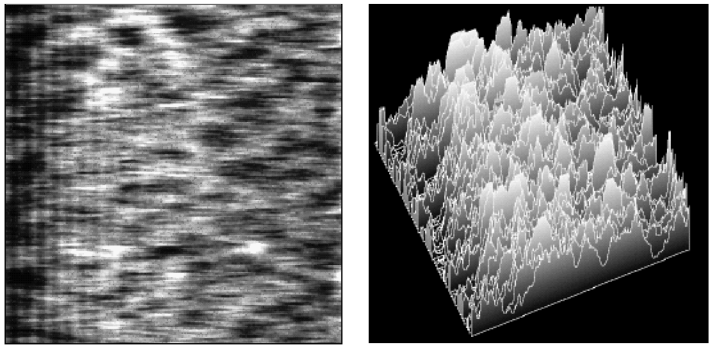


Figura 2.8 - Influência da geometria da ponteira na formação de uma imagem de AFM. Ilustrações dos (a) efeitos da média de forças (equivalente à convolução de forças) de uma ponteira afilada e outra mais rombuda sobre a amostra; (b) interpretação da imagem, dependente da geometria da ponteira; e a (c) presença de artefatos nas imagens de uma mesma amostra, gerados por diferentes formatos de ponteira: uma afilada e a outra rombuda. (Mannheimer, 2002; Cidade et al., 2003).

- ***Características das imagens obtidas pelo AFM***

As imagens características de um AFM são normalmente imagens digitais monocromáticas, variando em 256 gradações de cinza, compostas por 256 linhas e 256 colunas, num total de 65536 pixels. Entretanto, dependendo do equipamento utilizado, essas imagens podem ter dimensões maiores (p.ex. 512x512, 1024x1024 etc.).

O valor da intensidade de cinza, em cada pixel da imagem, representa uma dada altura acima do plano, uma vez que as imagens de AFM representam a superfície do objeto amostrado no espaço euclidiano tridimensional (3D), projetadas sobre um plano bidimensional (2D) (Fig. 2.9).



(a) (b)

Figura 2.9 - (a) Imagem de AFM de filamentos de DNA em alta concentração, medindo 200nm x 200nm, visualizado em 2D e (b) a sua representação tridimensional (3D) (Cidade et al., 2003).

Devido às características de construção do equipamento, as imagens obtidas pelo AFM, usualmente, apresentam uma relação sinal/ruído pobre (Cidade et al., 2000). Isso ocorre por causa de efeitos degenerativos na qualidade da imagem, provocados por borramentos resultantes da interação ponteira-amostra (degradação espacial), Fig. 2.8, e por ruídos aditivos originados do próprio equipamento (degradação pontual), causados por ruídos elétricos, eletrônicos, vibrações etc.

Dadas as características do processo de obtenção de imagens descritas neste tópico, o sistema de formação de imagens do AFM pode ser classificado como sendo um sistema linear espaço-invariante (SIPSF).

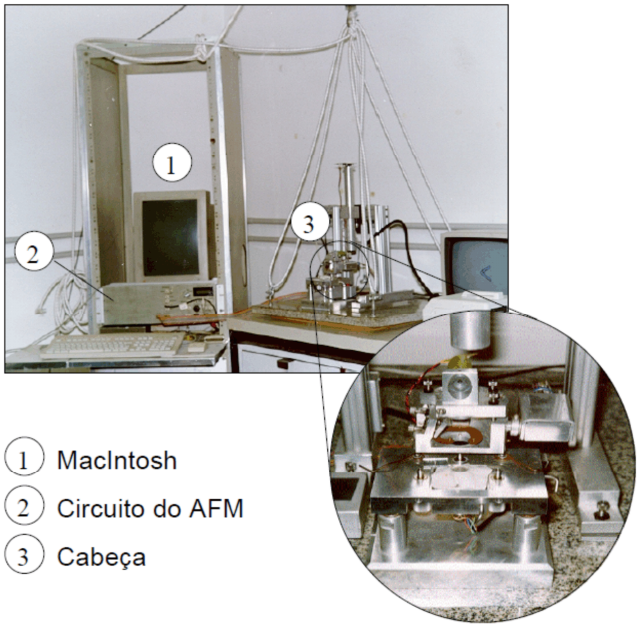


Figura 2.10 - Foto do Microscópio de Força Atômica do IBCCF, Instituto de Biofísica Carlos Chagas Filho da UFRJ, destacando os seus componentes principais (Cidade et al., 2003).

3 COMPUTAÇÃO PARALELA

3.1 Programação paralela

Há muito tempo se sabe que a tecnologia de fabricação de microprocessadores vem se aproximando do seu limite físico e que o desempenho e a velocidade de processamento têm melhorado significativamente. Apesar disso, entretanto, os usuários e as diversas áreas de conhecimento e pesquisa têm exigido cada vez mais poder computacional com o passar dos anos. Isso ocorre, em parte, devido ao fato de que muitos problemas, que alguns anos atrás demandariam demasiado tempo computacional para a execução nas máquinas disponíveis naquela época, hoje são factíveis e podem ser processados usando os equipamentos atuais.

O uso de equipamentos mais sofisticados e/ou dedicados é, muitas vezes, empregado para resolver a necessidade de mais poder computacional. Porém, a aquisição e manutenção desses equipamentos têm, normalmente, um custo muito alto. Como alternativa, podem ser empregadas técnicas de paralelismo e computação paralela na busca de melhores desempenhos de hardware e software a um custo bem menor. Muitas vezes, sem que saibamos, essas técnicas (*pipeline*, processamento superescalar, DMA etc.) estão sendo utilizadas e se encontram presentes nos processadores que usamos atualmente, ou estão fora deles: máquinas multiprocessadas, processamento distribuído, arquiteturas em clusters, *grid computing* etc.

Para que seja efetiva e alcance o máximo de desempenho, a programação em um ambiente paralelo deve ser específica e otimizada para a arquitetura que está sendo utilizada e isto pode ser feito de duas maneiras (Schlemer, 2003):

a) **Implícita** - implementa-se um programa serial normalmente usando uma linguagem de programação convencional qualquer (p.ex. Fortran, C etc.) e, através de um compilador paralelizador que analisa o código e paraleliza partes dele, gera-se um código executável paralelo ou parcialmente paralelo;

b) **Explícita** – cabe ao programador toda a responsabilidade da geração de códigos paralelos eficientes que explorem da melhor forma o problema proposto e a arquitetura utilizada.

3.2 MPI - Message Passing Interface

O MPI (*Message Passing Interface*) foi desenvolvido por um fórum internacional aberto (MPI Forum, 1994), composto por grandes empresas, centros acadêmicos e governos, para ser um padrão de desenvolvimento de sistemas paralelos, disponível para uma grande variedade de arquiteturas e sistemas (multiplataforma), acessível a vários estudantes e profissionais dos mais variados ramos da ciência e engenharia (Snir et al., 1996).

O MPI constitui uma biblioteca de definições e funções, para ser utilizada em programas C ou Fortran, e que torna possível o desenvolvimento de sistemas paralelos, usando memória distribuída. Por memória distribuída, aqui se entende que não existe uma memória global compartilhada que possa ser acessada por todos os processadores, que cada unidade de processamento possui a sua própria memória local e que a troca de dados entre as unidades é feita por meio de troca de mensagens (*message passing*), usando uma rede de comunicação (p.ex. rede local).

Os programas desenvolvidos com o MPI são implementados de forma explícita e o modelo de programação utilizado é o SPMD (*Single Program Multiple Data*). Nessa abordagem, diversas unidades de processadores são reunidas através de um canal de comunicação físico qualquer (barramento, cabo de rede etc.) e cada uma delas recebe uma cópia de um mesmo programa e os dados necessários para o seu processamento. Tão logo podem, iniciam a execução do programa paralelo independente dos demais processadores. Ao longo da execução, as unidades podem processar diferentes partes do programa, tomando como base um número de processo (ou *rank*) que identifica cada uma das unidades de processamento. Sempre que necessário, param a execução normal do programa para fazerem trocas de dados/controles com os demais processadores. Na maioria dos casos, o processo-0 (também conhecido como *root*) é normalmente escolhido pelo programador MPI para fazer a distribuição e coleta dos resultados em paralelo.

3.3 Paradigmas da programação paralela

Em programação, existem dois paradigmas importantes, usados no desenvolvimento e implementação de aplicações paralelas (Censor e Zenios, 1997):

a) **Paralelismo de dados** - técnica empregada quando um mesmo conjunto de instruções pode ser aplicado sobre vários elementos de dados, em vários processadores, simultaneamente (Fig. 3.1).

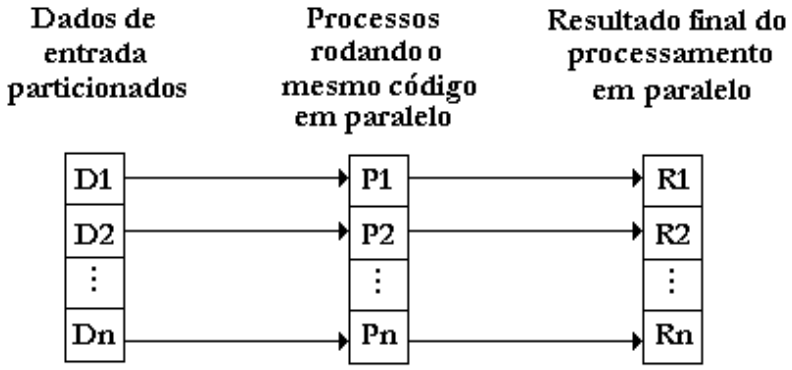


Figura 3.1 - Exemplo de paralelismo de dados.

b) **Paralelismo de controle** - técnica empregada quando as instruções de um programa podem ser particionadas e agrupadas em conjuntos de instruções independentes (estágios), de tal maneira que os conjuntos possam ser executados de forma paralela (Fig. 3.2).

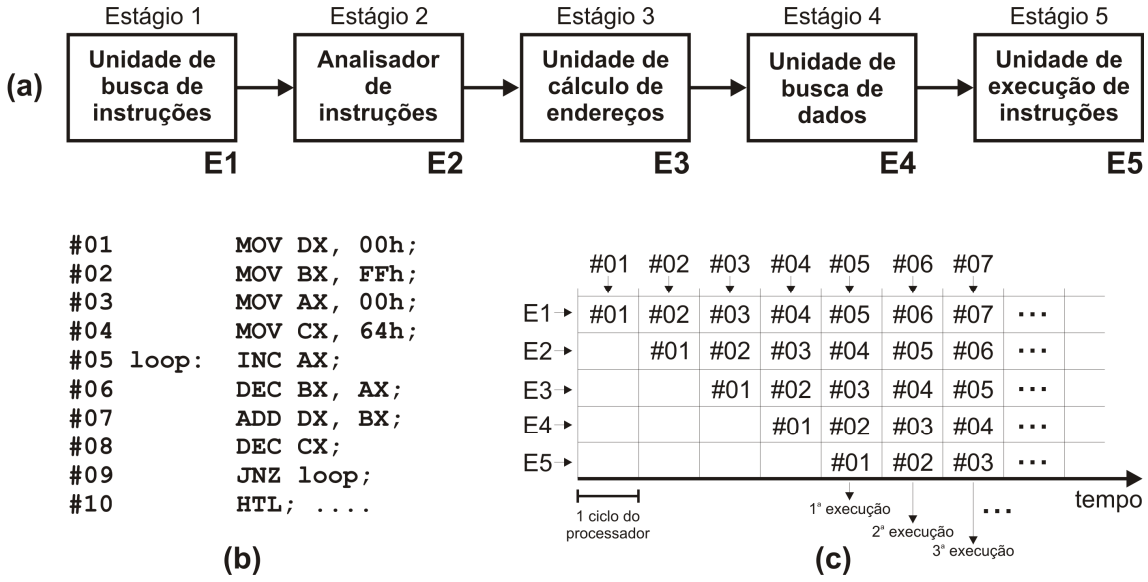


Figura 3.2 - Exemplo de paralelismo de controle, usando um (c) *pipeline* com (a) 5 estágios, executando (b) um pequeno programa assembly.

3.4 Paralelismo de dados: decomposição de domínio

A decomposição de domínio (*domain decomposition*) é uma técnica de paralelismo de dados cujo objetivo é particionar os dados do problema em unidades de dados menores (partições), de modo que possam ser distribuídas as múltiplas unidades de processamento para serem processadas de forma paralela.

Existem algumas maneiras de se particionar os dados, dentre as quais, podemos citar duas abordagens clássicas:

a) ***Striped Partition*** – é a abordagem mais simples para se trabalhar, onde uma matriz de dados é dividida em blocos de dados disjuntos menores (partições) e cada um destes blocos é enviado, separadamente, a uma unidade de processamento diferente.

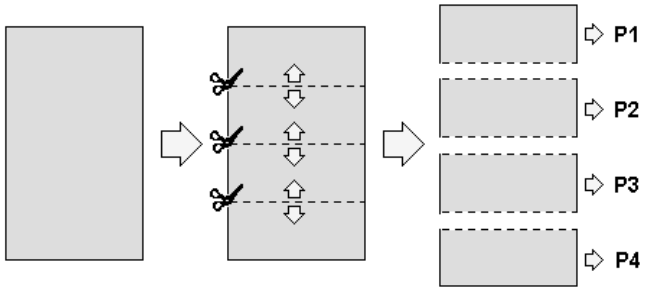


Figura 3.3 - *Striped partition*. A matriz de dados é dividida em blocos disjuntos (partições).

b) ***Overlapping Blocks* ou *Multisplittings*** – nessa abordagem, a matriz de dados é dividida em blocos sobrepostos.

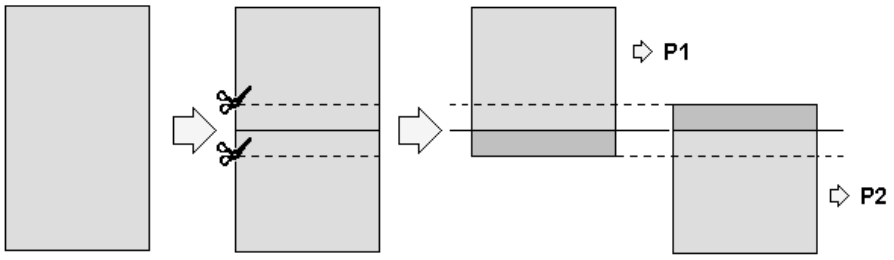


Figura 3.4 - *Overlapping partition.* Uma matriz de dados dividida em dois blocos sobrepostos.

- ***Formas de particionamento***

A forma como os blocos de dados são divididos, ou seja, como as partições se formam, pode variar muito de acordo com o problema a ser tratado. Conforme ilustrado na Fig. 3.5, as partições mais comuns são:

a) **partição 1D ou bloco** – partição formada pela divisão da matriz de dados em uma dimensão apenas. Ex.: partição bloco-linha ou bloco-coluna;

b) **partição 2D ou bloco-bloco** – partição formada pela divisão da matriz de dados em duas dimensões. Ex.: partição tabuleiro ou bloco-linha-coluna (*checkerboard*).

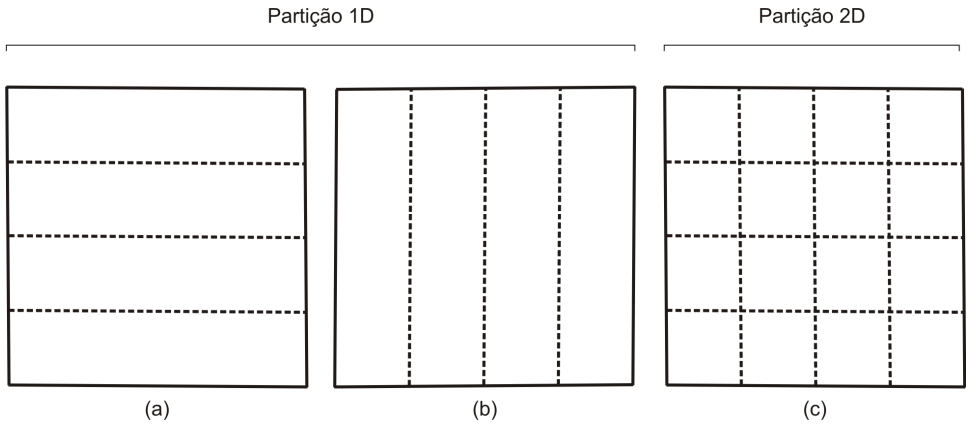


Figura 3.5 - Exemplo de partições de domínio: (a) bloco-linha (b) bloco-coluna e (c) tabuleiro.

O tamanho dos blocos não precisa ser necessariamente igual. Se as unidades de processamento forem mais ou menos homogêneas é importante que se utilizem blocos de mesmo tamanho. Porém, se as características dessas unidades forem diferentes (heterogêneas), aconselha-se então o emprego de blocos de tamanhos variados, onde os blocos menores podem ser enviados, por exemplo, para unidades de menor poder de processamento, que estiverem sobrecarregadas ou que possuírem pouca capacidade de memória e as partições maiores podem ser enviadas para unidades que não apresentam restrições para o tamanho do bloco de dados recebido.

3.5 Fatores críticos para a queda do desempenho de programas paralelos

O desejo de se implementar programas em paralelo surge da necessidade de se obter um poder computacional maior, de modo a alcançar resultados de processamento cada vez mais rápidos. Ao incluirmos mais de uma unidade de processamento na execução de uma aplicação, desejamos que a velocidade de execução e o desempenho computacional cresçam na mesma proporção. Entretanto, alguns fatores limitam o uso dos sistemas paralelos, fazendo com que a velocidade e o desempenho sejam menores do que desejamos. Isso se deve a alguns obstáculos existentes na paralelização que sobrecarregam os programas paralelos, reduzindo a sua eficiência e, consequentemente, aumentando os tempos de execução das aplicações em paralelo. Essa sobrecarga é conhecida na literatura como ***overhead*** e pode se originar de diversas formas (Pacheco, 1997; Kumar et al., 1994).

- ***Quantidade de trabalho ou computação extra***

A quantidade de trabalho executada por um programa paralelo é muitas vezes diferente daquela que é executada por um programa serial. Para se implementar um algoritmo em paralelo, são necessárias instruções extras na programação que não existem quando o mesmo algoritmo é implementado de modo serial. Por conta disso, a carga total de trabalho executada pelos processadores rodando uma aplicação em paralelo pode, na maioria dos casos, ser maior que a carga de um único processador rodando a mesma aplicação serialmente.

- ***Complexidades dos programas paralelos***

Nem sempre, a solução de um problema em paralelo é resolvida de forma simples e fácil. Na maioria das vezes, a solução em paralelo envolve uma complexidade muito maior do que quando o mesmo problema é resolvido serialmente. Por exemplo: (a) execução de tarefas em paralelo que necessitam de comunicação entre os processadores, (b) sincronização entre os processos, (c) conversão do algoritmo serial em paralelo, (d) nem tudo é paralelizável etc. Esse aumento na complexidade dos programas paralelos acaba resultando em codificações extras (*overheads*) no programa em paralelo que não existem no programa serial.

- ***Comunicação interprocessadores***

Qualquer sistema paralelo não-trivial necessita que os processadores se comuniquem e troquem dados entre si. O tempo de transferência de dados entre os processadores é normalmente o principal obstáculo dos programas paralelos e origem da maioria dos *overheads.*

- ***Desbalanceamento de carga***

Em muitas aplicações paralelas, dependendo do problema (p.ex.: busca e otimização), é impossível se determinar a quantidade trabalho e/ou subtarefas associadas aos vários processadores. Já em outros sistemas paralelos, podemos encontrar arquiteturas heterogêneas com diferentes velocidades de processamento. Se diferentes processadores têm diferentes cargas de trabalho, pode ocorrer em algum momento que algum processador tenha que ficar aguardando enquanto que os demais processadores ainda estão trabalhando no problema. Além disso, em certos momentos, pode ocorrer que os processadores (todos ou apenas um grupos deles) tenham que se sincronizar. Se nem todos os processadores estiverem prontos para a sincronização ao mesmo tempo, ocorrerá que alguns deles terão que ficar aguardando até que os demais estejam prontos.

- ***Causas comuns de overheads***

Diversos fatores podem introduzir *overheads* na programação paralela, entretanto, as causas mais comuns são:

a) Tempo de processamento para execução e controle dos programas em paralelo (inicialização, finalização, sincronização etc.);
b) Tempo de comunicação gasto na troca de dados e/ou controles entre as unidades de processamento;
c) Período de tempo em que uma unidade de processamento fica aguardando por uma comunicação (modo de espera ou ***idle-time***);
d) Tempo gasto em disputas por um mesmo recurso computacional (disco, memória etc.). Também conhecido como **conflito por recursos**;

e) Tempo de espera em que uma unidade de processamento fica aguardando pelo resultado de outra para continuar o seu processamento (**conflito de dados**);
f) Tempo consumido após uma tomada de decisão de parada do sistema (definitiva ou temporária), enquanto as outras ainda estão processando (p.ex.: desvios condicionais, interrupções etc.), conhecido como **conflito de controle**;
g) Sobrecarga imposta pelos compiladores, bibliotecas, ferramentas, sistemas operacionais etc.; usados na implementação e execução dos programas paralelos etc.

- ***Efeitos da comunicação no processamento paralelo***

O objetivo principal do processamento paralelo é o emprego de várias unidades de processamento na busca de uma solução mais rápida de um problema específico qualquer. Para que as unidades trabalhem de forma conjunta, entretanto, é preciso que haja um canal de comunicação que possibilite a troca de informações entre elas. Nos programas desenvolvidos com o MPI, essa comunicação é feita via troca de mensagens (*message passing*), utilizando uma rede de comunicação (rede local, internet etc.) que é significativamente mais lenta que os acessos que uma unidade de processamento faz ao seu próprio hardware como, por exemplo, a memória. É por isso que, sempre que as unidades enviam e/ou recebem mensagens, elas estarão consumindo com essa comunicação um tempo computacional bem maior. Um outro problema, que reduz o desempenho e que ocorre com uma certa frequência, é o tempo que uma unidade de processamento fica ociosa (*idle-time*), aguardando pelo término de uma comunicação.

4 FORMULAÇÃO DO PROBLEMA DE RESTAURAÇÃO DE IMAGENS

4.1 Modelo de degradação

Em restauração, o processo de degradação de uma imagem pode ser modelado da seguinte forma (Andrew e Hunt, 1977; Gonzalez e Woods, 2000; Gonzalez e Wintz, 1987; Kang e Katsaggelos, 1995)

$$y = Hx + \eta\,, \tag{17}$$

onde x e y representam, respectivamente, as imagens original e degradada, lexicograficamente ordenadas, a matriz H um operador que descreve a função de borramento que juntamente com um termo de ruído aditivo η operam sobre a imagem x (plano objeto), produzindo uma imagem observada y (plano imagem), obtida através de dados experimentais.

Na Fig. 4.1 é apresentado o modelo do processo de degradação de imagens. Nele, o operador degenerativo H (ou borramento) opera sobre uma imagem de entrada x (imagem real), que acrescida do ruído aditivo η, produz a imagem degradada y observada experimentalmente.

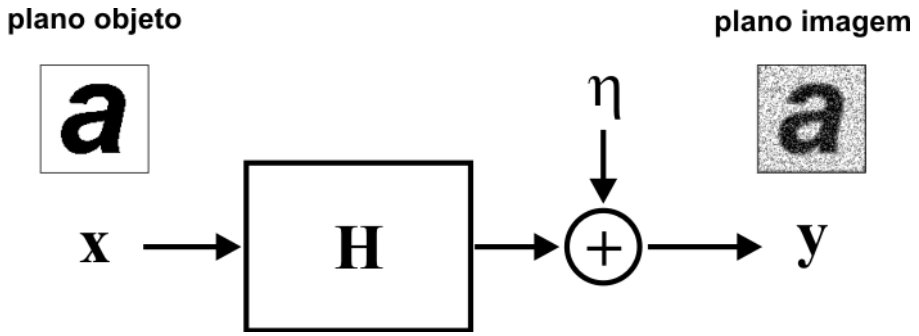


Figura 4.1 - Modelo do processo de degradação de imagens.

4.2 Problema de restauração de imagens

A aplicação de técnicas de restauração de imagens surge da necessidade de melhorar um aspecto e/ou recuperar uma informação qualquer de uma imagem, que tenha sofrido algum tipo de degradação: borramento, ruído aditivo etc. Basicamente, o problema consiste em, usando um conhecimento *a priori* do fenômeno de degradação, tentar reconstruir e/ou recuperar uma imagem degradada, buscando-se a imagem real do objeto x (ou uma estimativa $\hat{x}$ mais próxima dela), partindo de uma imagem y conhecida obtida através de um experimento qualquer (p.ex. AFM) e que apresenta algum tipo de degradação.

Ao longo do tempo, diversas técnicas de restauração têm sido empregadas. Dentre elas, só para citar algumas, temos: morfologia matemática (Gonzalez e Woods, 2000), deconvolução, FFT, filtros (pseudo-inversa, Wiener etc.), Wavelet, Pixon (Puetter et al., 2005) etc.

- ***Técnicas no domínio da frequência***

Tradicionalmente, a maioria das técnicas que operam no domínio da frequência utiliza, principalmente, a Transformada Rápida de Fourier (FFT) e a sua inversa (FFT^{-1}). Caso o ruído possa ser desprezado, a imagem real pode ser determinada e calculada eficazmente usando esse método que apresenta uma boa velocidade de processamento. Entretanto, a técnica de deconvolução usando Fourier não é eficaz quando o ruído não pode ser desprezado (Puetter, Gosnell e Yahill, 2005). Além disso, quando esta técnica é empregada em restaurações de imagens de AFM, ela penaliza tanto o ruído quanto o sinal de alta frequência, prejudicando o resultado da restauração e impedindo a recuperação de características importantes da imagem, tais como bordas e texturas, que estejam na mesma faixa de frequência do ruído aditivo (Cidade et al., 2000; Weissman et al., 1992).

- ***Técnicas no domínio espacial***

No domínio espacial, a solução do problema de restauração resume-se em encontrar o operador inverso de H ou uma estimativa $\hat{x}$, que seja a mais próxima possível da imagem original, sujeito a um critério de otimização ou grau ótimo.

Assumindo-se a Eq. (17) sem o ruído aditivo ($\eta = 0$) e admitindo que H seja inversível, uma solução possível para o problema poderia ser obtida, fazendo-se

$$x = H^{-1} y . \tag{18}$$

Entretanto, na maioria dos problemas de interesse (p.ex. desfocamento) a matriz H é singular e, portanto, não possui inversa (Noonan e Natarajan, 1997). Além disso, soluções diretas em restaurações de imagem são notoriamente sensíveis a pequenas variações em y. Isso significa que a busca de uma solução direta do problema através da Eq. (18), além de ser ineficiente sob o ponto de vista computacional, devido o custo operacional para se encontrar a matriz inversa H^{-1}, o mal-condicionamento da matriz de borramento H faz com que H^{-1} atue como um amplificador, significando que uma pequena perturbação nos dados experimentais traduz-se em uma grande instabilidade na solução do problema, tornando-a pouco exequível (Kang e Katsaggelos, 1995; Katsaggelos, 1989). Assim, quando não há garantia da existência, única e estável de uma solução para o problema com base na inversão direta, tais problemas são conhecidos na literatura como problemas mal-postos (Katsaggelos, 1989; Paik e Katsaggelos, 1989).

Considerando-se H fixo, uma estabilização poderia ser alcançada através do emprego de mínimos quadrados na determinação da solução, revertendo o problema para bem-posto, ou seja,

$$L(x) = \min_{x} \left\| y - Hx \right\|^2 . \tag{19}$$

Porém, erros numéricos provenientes do processo computacional iterativo para determinação da solução do problema (Karayiannis e Venetsanopoulos, 1989; Cidade et al., 1998), fazem com que o operador de borramento H implicitamente comporte-se como se estivesse sujeito a variações, como que incorporasse tais perturbações, tornando o problema mal-posto novamente.

Uma maneira de solucionar este problema seria o uso de um termo de regularização (Tikhonov e Arsenin, 1977). Nesse caso, uma solução para o problema pode ser encontrada minimizando-se o funcional de regularização de Tikhonov (Cidade et al., 2000, 2003; Paik e Katsaggelos, 1989), dado por

$$Q(x) = \left\| y - Hx \right\|^2 + \alpha S \,,\ \alpha > 0\,, \tag{20}$$

onde αS é o termo de regularização, que representa o tratamento dado à porção ruidosa da imagem, α o parâmetro de regularização que determina a estabilidade do funcional Q e S uma função de regularização ou funcional de estabilização do processo de restauração.

Fazendo-se $\alpha = 0$, a solução da Eq. (20) tenderia para $H^{-1}y$, Eq. (18), enquanto que para $\alpha \to \infty$, a solução tenderia para o funcional de estabilização S que, normalmente, representa uma possível solução suavizada do problema (Paik e Katsaggelos, 1990). A escolha entre a deconvolução e a suavização é determinada pelo valor de α, que irá estabelecer um compromisso entre a acurácia e a estabilidade da solução do problema.

Jeng e Woods (1998) afirmam que a restauração usando técnicas do domínio espacial possui uma grande vantagem sobre o domínio da frequência, pois o método utilizado para se estimar a imagem tem um caráter recursivo, onde os pontos da imagem (pixels) são tratados em função de uma vizinhança local (tratamento em base local), sem com isso penalizar significativamente os componentes do sinal (contraste) e ruído.

4.3 Algoritmo de convolução discreta

Dado que as imagens digitais são expressas em coordenadas espaciais discretas, onde o valor em cada ponto representa uma pequena região de imagem contínua e digitalizada (ou pixel), e dada a característica linear espaço-invariante do processo de obtenção de imagens do AFM, a Eq. (17) pode ser expressa em termos de uma convolução discreta, dada por

$$y = H * x + \eta\,. \tag{21}$$

Assumido $M = (n-1)$ e admitindo-se que x e y possuam uma dimensão $n \times n$, desprezando-se, por ora, o termo do ruído aditivo ($\eta = 0$) e redefinindo a Eq. (21) ponto a ponto, tem-se

$$y_{ij} = \sum_{k=0}^{M}\sum_{l=0}^{M} h_{i-k,j-l}\, x_{kl} \quad , \; i, j = 0,1,...,M \; . \tag{22}$$

Pela propriedade comutativa da convolução, onde

$$H * x = x * H \quad , \tag{23}$$

A Eq. (22), também pode ser escrita como

$$y_{ij} = \sum_{k=0}^{M}\sum_{l=0}^{M} x_{i-k,j-l}\, h_{kl} \; , \; i, j = 0,1,...,M \; . \tag{24}$$

Em processamento digital de sinais, o cálculo da convolução pode ser expresso de duas formas diferentes (Smith, 1999):

a) sob o ponto de vista do sinal de entrada; ou
b) sob o ponto de vista do sinal de saída.

O cálculo apresentado nas Eqs. (15), (22) e (24) é dado sob o ponto de vista do sinal de entrada e é a forma mais geral usada para expressar a convolução. Dado dessa forma, a expressão analisa como cada ponto do plano objeto x (sinal de entrada) afeta (ou contribui para) a formação dos pontos no plano imagem y (sinal de saída).

Na segunda maneira, reverte-se o ponto de vista, analisando o processo de formação de cada um dos sinais de saída. Examinando-se, então, o processo de formação de imagens do AFM sob o ponto de vista do sinal de saída, observa-se que cada elemento y_{ij} do plano imagem é formado pela convolução de elementos de x que estão ao redor do ponto (i, j) no plano objeto. Quanto mais distante esse ponto estiver de (i, j), menor é a sua influência e, portanto, menor a sua contribuição para a formação do sinal de saída y_{ij}. Dado sob essa ótica, reescreve-se a Eq. (22) (Cidade et al., 2000) como

$$y_{ij} = \sum_{k=-N}^{N}\sum_{l=-N}^{N} b_{kl}\, x_{i+k,j+l} \quad , \; i, j = 0,1,...,M \; , \tag{25}$$

sendo $B = [b_{kl}]_{2N+1 \times 2N+1}$ o operador de borramento, descrito sob o ponto de vista do sinal de saída, e N um valor arbitrário que define de forma discreta as dimensões de B ($\dim B = 2N+1$) e limita uma área em torno do ponto (i, j) no plano objeto, descrevendo o grau de influência dos pontos dessa área para a formação do elemento y_{ij} no plano imagem.

Sabendo-se que (a) quanto mais distante o ponto $x_{i+k,j+l}$ estiver de (i, j) no plano objeto, menor é a sua contribuição para a formação do elemento y_{ij} no plano imagem e (b) dada a imprecisão e erros de arredondamento computacionais, pode-se admitir que a partir de uma determinada distância N de (i, j), o valor de contribuição do ponto para a formação do ponto imagem y_{ij} seja nulo (zero).

4.4 Operador de borramento B

Na Eq. (25), a matriz B é um operador que determina a função de distribuição de energias observada na imagem y e que representa o modelo de interação de forças da ponteira do microscópio e a amostra analisada, descrevendo a função de borramento (função de espalhamento pontual ou PSF) que ocorre no processo de aquisição de imagens do AFM.

Nas abordagens clássicas de restauração, normalmente assume-se que o modelo de borramento e os seus parâmetros são conhecidos *a priori*. Entretanto, no mundo real, essa função geralmente não é conhecida, podendo até existir dificuldade em se obter uma representação de um modelo que expresse fielmente a interação ponteira-amostra. Na falta de uma estimativa, assume-se, então, que a função segue um modelo de distribuição estatística conhecida. Em processamento de imagens, os modelos de distribuições mais usados são (Telatar, 2004; Banham e Katsaggelos, 1997):

a) Poisson: normalmente usado em processamento de imagens astronômicas, raios-x, mamogramas, angiografias etc.

$$B_{R(1)}(n1, n2; \lambda 1, \lambda 2) = \frac{\lambda_1^{n1}\, \lambda_2^{n2}}{(n1)!(n2)!} e^{-(\lambda 1 + \lambda 2)}; \tag{26}$$

b) Exponencial: modelo encontrado em algumas aplicações de geração de imagens baseados em lasers ou que possam ser caracterizados como.

$$B_{R(2)}(n1,n2) = e^{-(n1^2+n2^2)}; \tag{27}$$

c) Gaussiana: modelo mais comum usado em restauração de imagens, bastante empregado em aplicações que sofrem turbulências atmosféricas, desfocamento e movimento de câmeras etc.

$$B_{R(3)}(n1,n2;\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(n1^2+n2^2)/2\sigma^2}; \tag{28}$$

d) Rayleigh: similar à distribuição Gaussiana, o modelo é encontrado em alguns sistemas de formação de imagens médicas e sensoriamento.

$$B_{R(4)}(n1,n2;\sigma^2) = \frac{n1\,n2}{\sigma^4} e^{-(n1^2+n2^2)/2\sigma^2}, \tag{29}$$

onde $n1$ e $n2$ são as dimensões da matriz de borramento, σ^2 é a variância da função de borramento, λ_1 e λ_2 são as taxas de ocorrência relacionadas as intensidades de luz da imagem e $B_{R(.)}$ os modelos de distribuição acima relacionados.

Para representar o modelo de iteração ponteira-amostra do AFM, Kokaram et al. (1995) sugerem o emprego da distribuição gaussiana, enquanto Dongmo et al. (1996) e Weisman et al. (1992) sugerem um parabolóide de revolução. Neste trabalho, entretanto, o operador B usado assumirá uma dependência espacial do tipo gaussiana, conforme ilustrado nas Fig. 4.3 e Fig. 4.2, dada por (Kokaram et al., 1995; Cidade et al., 2000):

$$B \propto e^{-\frac{d^2}{\sigma^2}}, \tag{30}$$

onde d é a distância euclidiana dos pontos em relação a um dado ponto P da amostra e σ^2 é a variância que simula os diferentes tipos de ponteiras, mais finas ou mais grossas (Furtado, 2002).

$$B_{5\times5} = \begin{bmatrix} 3{,}613\times10^{-2} & 3{,}894\times10^{-2} & 3{,}993\times10^{-2} & 3{,}894\times10^{-2} & 3{,}613\times10^{-2} \\ 3{,}894\times10^{-2} & 4{,}198\times10^{-2} & 4{,}304\times10^{-2} & 4{,}198\times10^{-2} & 3{,}894\times10^{-2} \\ 3{,}993\times10^{-2} & 4{,}304\times10^{-2} & 4{,}413\times10^{-2} & 4{,}304\times10^{-2} & 3{,}993\times10^{-2} \\ 3{,}894\times10^{-2} & 4{,}198\times10^{-2} & 4{,}304\times10^{-2} & 4{,}198\times10^{-2} & 3{,}894\times10^{-2} \\ 3{,}613\times10^{-2} & 3{,}894\times10^{-2} & 3{,}993\times10^{-2} & 3{,}894\times10^{-2} & 3{,}613\times10^{-2} \end{bmatrix}$$

Figura 4.2 - Representação matricial de um operador de borramento B normalizado, do tipo gaussiano, de dimensão 5x5, com $\sigma^2 = 20$.

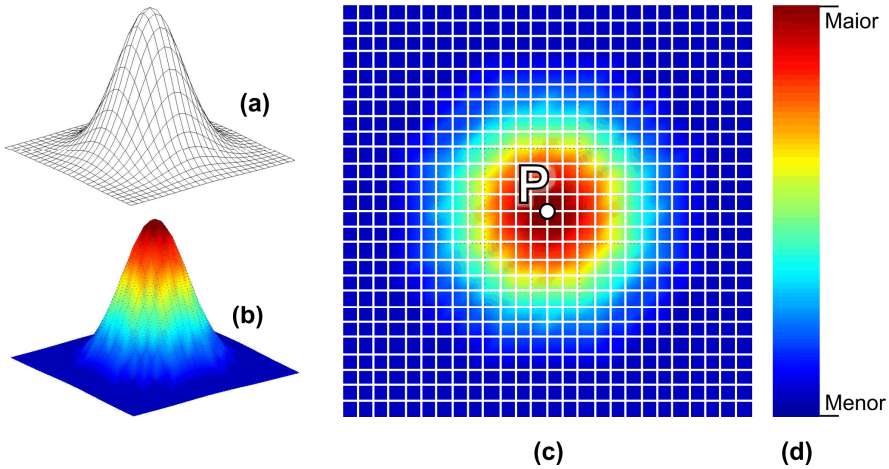


Figura 4.3 - Visualização do operador de borramento do tipo gaussiano (a) e (b), mostrando em detalhes a distribuição das forças de iteração normalizadas (b), projetado sobre um plano (c), numa escala de valores (d) que varia do menor ao maior grau de contribuição dos pontos periféricos, para a formação do ponto central, na medida em que se aproximam de P.

Independente do modelo adotado, entretanto, uma vez que o sistema de formação de imagens normalmente não absorve e nem gera energia, o operador B deve satisfazer, então, a seguinte condição inicial:

$$\sum_{k,l=-N}^{N} b_{kl} = 1 \tag{31}$$

Para garantir que a condição dada na Eq. (31) seja satisfeita, divide-se todos os elementos de B pelo somatório desses elementos, normalizando-o.

Além disso, como as intensidades (energias) envolvidas são não-negativas, tem-se também que

$$b_{kl} \geq 0\,,\ k,l = -N,...0,...,N\,. \tag{32}$$

E, para os índices que estiverem fora desta faixa, tem-se que

$$b_{kl} = 0\,,\ \forall |k,l| > N\,. \tag{33}$$

4.5 Funções de regularização

Diferentes funcionais de regularização S podem ser empregados como critério de estabilização do funcional Q, Eq. (20), usado no processo de restauração de imagens (Csiszár, 1991; Zervakis et al., 1995). Dentre eles, os mais utilizados são:

a) Mínima energia

$$S = \frac{1}{2}\sum_{i=0}^{M}\sum_{j=0}^{M}\left(\hat{x}_{ij} - \overline{x}_{ij}\right)^2; \tag{34}$$

b) Entropia-cruzada

$$S = -\sum_{i=0}^{M}\sum_{j=0}^{M}\left[\hat{x}_{ij} - \overline{x}_{ij} - \hat{x}_{ij}\ln\frac{\hat{x}_{ij}}{\overline{x}_{ij}}\right]; \tag{35}$$

onde $\hat{x}$ representa o valor esperado da solução (ou estimativa de x) e $\overline{x}$, um valor de referência, para tratar o ruído aditivo, que pode ser: (a) a própria imagem borrada, obtida experimentalmente com o AFM, (b) um valor constante qualquer como, por exemplo, a média das intensidades dos pixels da imagem etc.

No sentido de encontrar uma possível adequação ao tratamento de imagens, Cidade et al. (2000) propõem um método pouco tradicional na área de restauração de imagens, em que se emprega um funcional geral de regularização S que corresponde a uma família de funções de regularização, construído com distâncias de Bregman (1967), baseados no funcional q-discrepância (Carita et al., 1999)

$$S = D_q(\hat{x}, \bar{x}) = \frac{1}{1+q} \sum_{i=0}^{M} \sum_{j=0}^{M} \left\{ \hat{x}_{ij} \left[\frac{\hat{x}_{ij}^q - \bar{x}_{ij}^q}{q} \right] - \bar{x}_{ij}^q (\hat{x}_{ij} - \bar{x}_{ij}) \right\}, \tag{36}$$

onde q é um parâmetro ajustável a partir do qual, além das Eqs. (34) ($q=1$) e (35) ($q \to 0$), outras normas podem ser obtidas.

4.6 Técnica de restauração baseado na minimização do Funcional de Regularização de Tikhonov

Aplicando-se as Eq. (25) e (36) na Eq. (20) e substituindo x (imagem real) por $\hat{x}$ (estimativa), obtém-se o funcional de regularização de Tikhonov (Cidade et al., 2000) e, na busca de uma estimativa da imagem real, procura-se minimizar o funcional dado por

$$Q(\hat{x}) = \sum_{i=0}^{M} \sum_{j=0}^{M} \left[y_{ij} - \sum_{k=-N}^{N} \sum_{l=-N}^{N} b_{kl} \hat{x}_{i+k,\, j+l} \right]^2 + \alpha \frac{1}{1+q} \sum_{i=0}^{M} \sum_{j=0}^{M} \left\{ \hat{x}_{ij} \left[\frac{\hat{x}_{ij}^q - \bar{x}_{ij}^q}{q} \right] - \bar{x}_{ij}^q (\hat{x}_{ij} - \bar{x}_{ij}) \right\}, \tag{37}$$

Cujo valor residual pode ser obtido com

$$R(\hat{x}) = \sum_{i=0}^{M} \sum_{j=0}^{M} \left[y_{ij} - \sum_{k=-N}^{N} \sum_{l=-N}^{N} b_{kl} \hat{x}_{i+k,\, j+l} \right]. \tag{38}$$

De modo a minimizar o funcional dado pela Eq. (37), escreve-se a equação do ponto crítico

$$\vec{F}(\hat{x}) = \frac{\partial Q(\hat{x})}{\partial \hat{x}} = 0\,, \quad \text{i.e. } \frac{\partial Q(\hat{x})}{\partial \hat{x}_{00}} = 0\,,\ \frac{\partial Q(\hat{x})}{\partial \hat{x}_{01}} = 0\,,\ ...,\ \frac{\partial Q(\hat{x})}{\partial \hat{x}_{MM}} = 0\,. \tag{39}$$

Isso nos leva a um sistema de n^2 (ou $[M+1]^2$) equações não-lineares para ser resolvido, do tipo:

$$\begin{cases} F_{00}(\hat{x}_{00}, \hat{x}_{01}, ..., \hat{x}_{MM}) = \dfrac{\partial Q(\hat{x})}{\partial \hat{x}_{00}} = 0 \\ F_{01}(\hat{x}_{00}, \hat{x}_{01}, ..., \hat{x}_{MM}) = \dfrac{\partial Q(\hat{x})}{\partial \hat{x}_{01}} = 0 \\ \cdots \\ F_{MM}(\hat{x}_{00}, \hat{x}_{01}, ..., \hat{x}_{MM}) = \dfrac{\partial Q(\hat{x})}{\partial \hat{x}_{MM}} = 0 \end{cases}, \tag{40}$$

ou então:

$$F_{rs}(\vec{\hat{x}}) = \frac{\partial Q(\hat{x})}{\partial \hat{x}_{rs}} = 0\,, \quad r,s = 0,...,M\,. \tag{41}$$

Usando o método de Newton-Raphson multivariável (Lim, 1990; Press et al., 1996) para solucionar o sistema de Eqs. (41), novas estimativas podem ser obtidas, empregando-se

$$\vec{\hat{x}}^{\,t+1} = \vec{\hat{x}}^{\,t} + \Delta\vec{\hat{x}}^{\,t}\,, \quad t = 0,1,2,...\,, \tag{42}$$

onde t é o contador de iterações do método Newton-Raphson e como estimativa inicial $\vec{\hat{x}}^{\,0}$ pode-se usar a própria imagem obtida pelo microscópio.

Empregando-se uma expansão de Taylor e retendo os termos até a primeira ordem, tem-se

$$F_{rs}(\vec{\hat{x}}^{\,t+1}) = F_{rs}(\vec{\hat{x}}^{\,t} + \Delta\vec{\hat{x}}^{\,t}) = F_{rs}(\vec{\hat{x}}^{\,t}) + \sum_{m=0}^{M}\sum_{n=0}^{M} \frac{\partial F_{rs}}{\partial \hat{x}_{mn}}\bigg|_{\hat{x}^t} . \Delta\hat{x}^t_{mn} = 0\,. \tag{43}$$

Empregando-se o método iterativo de Gauss-Seidel (Press et al., 1996; Silva Neto e Moura Neto, 1999; Kreyszig, 1972) para a solução do sistema de equações lineares (43), escreve-se

$$\Delta\hat{x}_{rs}^{\,t,c+1} = -\frac{1}{\left(\partial F_{rs}\big/\partial\hat{x}_{mn}\right)\Big|_{\substack{t,c\\ m=r\\ n=s}}}\left\{\begin{array}{l} F_{rs}\Big|_{\hat{x}^{t,c}} + \displaystyle\sum_{m=0}^{r-1}\sum_{n=0}^{M}\frac{\partial F_{rs}}{\partial\hat{x}_{mn}}\Big|_{\hat{x}^{t,c}}.\Delta\hat{x}_{mn}^{t,c+1} + \sum_{n=0}^{s-1}\frac{\partial F_{rs}}{\partial\hat{x}_{mn}}\Big|_{\hat{x}^{t,c}}^{m=r}.\Delta\hat{x}_{mn}^{t,c+1}\Big|^{m=r} + \\ \quad + \displaystyle\sum_{n=s+1}^{M}\frac{\partial F_{rs}}{\partial\hat{x}_{mn}}\Big|_{\hat{x}^{t,c}}^{m=r}.\Delta\hat{x}_{mn}^{t,c}\Big|^{m=r} + \sum_{m=r+1}^{M}\sum_{n=0}^{M}\frac{\partial F_{rs}}{\partial\hat{x}_{mn}}\Big|_{\hat{x}^{t,c}}.\Delta\hat{x}_{mn}^{t,c} \end{array}\right\}, \quad (44)$$

onde $\Delta\hat{x}^{t,0} = 0$ e c é o contador de iterações do método iterativo de Gauss-Seidel.

De modo simplificado, reescreve-se a Eq. (44)

$$\Delta\hat{x}_{rs}^{\,t,c+1} = -\frac{1}{\left(\partial F_{rs}\big/\partial\hat{x}_{mn}\right)\Big|_{\substack{t,c\\ m=r\\ n=s}}}\left\{F_{rs}\Big|_{\hat{x}^{t,c}} + \sum_{\substack{m=0\\ m\neq r}}^{M}\sum_{\substack{n=0\\ n\neq s}}^{M}\frac{\partial F_{rs}}{\partial\hat{x}_{mn}}\Big|_{\hat{x}^{t,\tilde{c}}}.\Delta\hat{x}_{mn}^{t,\tilde{c}}\right\}. \quad (45)$$

onde $\tilde{c} = \begin{cases} c+1, & (m<r) \text{ ou } (m=r \;\; e \;\; n<s) \\ c \quad , & \text{caso contrário} \end{cases}$.

Acrescentando-se o próprio elemento da diagonal principal na Eq. (45) de forma que esta se mantém e empregando ao final um fator de relaxação ω, obtém-se o método iterativo SOR (*Sucessive OverRelaxation*) para a solução do sistema de equações lineares (Apêndice-A), em substituição ao método de Gauss-Seidel, usado na Eq. (45):

$$\Delta\hat{x}_{rs}^{\,t,c+1} = \Delta\hat{x}_{rs}^{\,t,c} - \frac{\omega}{\left(\partial F_{rs}\big/\partial\hat{x}_{mn}\right)\Big|_{\substack{t,c\\ m=r\\ n=s}}}\left\{F_{rs}\Big|_{\hat{x}^{t,c}} + \sum_{m=0}^{M}\sum_{n=0}^{M}\frac{\partial F_{rs}}{\partial\hat{x}_{mn}}\Big|_{\hat{x}^{t,\tilde{c}}}.\Delta\hat{x}_{mn}^{t,\tilde{c}}\right\}, \quad (46)$$

onde ω é o fator de relaxação do método SOR, $\mathrm{r,s} = 0,1,2,...,\mathrm{M}$, $\Delta\vec{\hat{x}}^{t,0} = 0$ e c é um contador de iterações. Para $\omega = 1$, a Eq. (45) se mantém e o método empregado para a solução do sistema de equação linear é de Gauss-Seidel, caso contrário ($\omega \neq 1$), o método aplicado será o SOR.

Das Eqs. (37) e (41), obtém-se (Cidade et al., 2003)

$$F_{rs} = -2\sum_{i=r-N}^{r+N}\sum_{j=s-N}^{s+N}\left[\left(y_{ij} - \sum_{k=-N}^{N}\sum_{l=-N}^{N} b_{kl}\,\hat{x}_{i+k,j+l}\right) b_{r-i,s-j}\right] + \frac{\alpha}{q}\left(\hat{x}_{rs}^{q} - \bar{x}_{rs}^{q}\right), \tag{47}$$

a partir do qual, escreve-se

$$F_{mn} = \frac{\partial F_{rs}}{\partial \hat{x}_{mn}} = 2\sum_{k=-N}^{N}\sum_{l=-N}^{N} b_{kl}\,b_{m-r-k,n-s-l} + \alpha\,\hat{x}_{rs}^{q-1}\delta(r,m)\delta(s,n)\,, \tag{48}$$

onde $\delta(.)$ representa o delta de Kronecker (veja no Apêndice-B maiores detalhes sobre o desenvolvimento das expressões F_{rs} e F_{mn}).

Calculadas as correções $\Delta\hat{x}_{rs}$, $\mathrm{r,s} = 0,1,2,...,\mathrm{M}$, as novas estimativas $\vec{\hat{x}}^{\,t+1}$ podem ser obtidas, usando-se a Eq. (42). Porém, ao invés de utilizar diretamente esta equação, pode-se empregar um fator de atenuação γ sobre os incrementos, de forma a auxiliar na convergência do algoritmo.

$$\vec{\hat{x}}^{\,t+1} = \vec{\hat{x}}^{\,t} + \gamma\,\Delta\vec{\hat{x}}^{\,t}, \quad t = 0,1,2,... \text{ e } 0 < \gamma < 1. \tag{49}$$

4.6.1 Critério de escolha da solução do problema de restauração

O critério adotado para se determinar qual a imagem restaurada é a mais provável dentro do espaço de soluções baseia-se no menor valor do funcional Q, Eq. (37). Como o funcional é de natureza convexa, sempre existirá um valor mínimo associado (Cidade et al., 2003). Isso implica dizer que, a cada iteração, é preciso verificar o valor do funcional Q da imagem recém-restaurada ($Q(\hat{x}^{t+1})$) com o valor obtido na iteração anterior ($Q(\hat{x}^{t})$). Se o novo valor do Q for menor que o anterior, aceita-se a imagem recém-restaurada como solução do problema. Caso contrário, mantém a última imagem restaurada com valor Q mínimo como sendo a solução.

4.6.2 Fluxograma do algoritmo de restauração

Na Fig. 4.4, é apresentado o fluxograma simplificado do algoritmo de restauração, que diferentemente do algoritmo desenvolvido por Cidade (2000), emprega o método iterativo SOR (*Sucessive OverRelaxation*), ao invés de Gauss-Seidel, para a solução do sistema de equações lineares, Eq. (41).

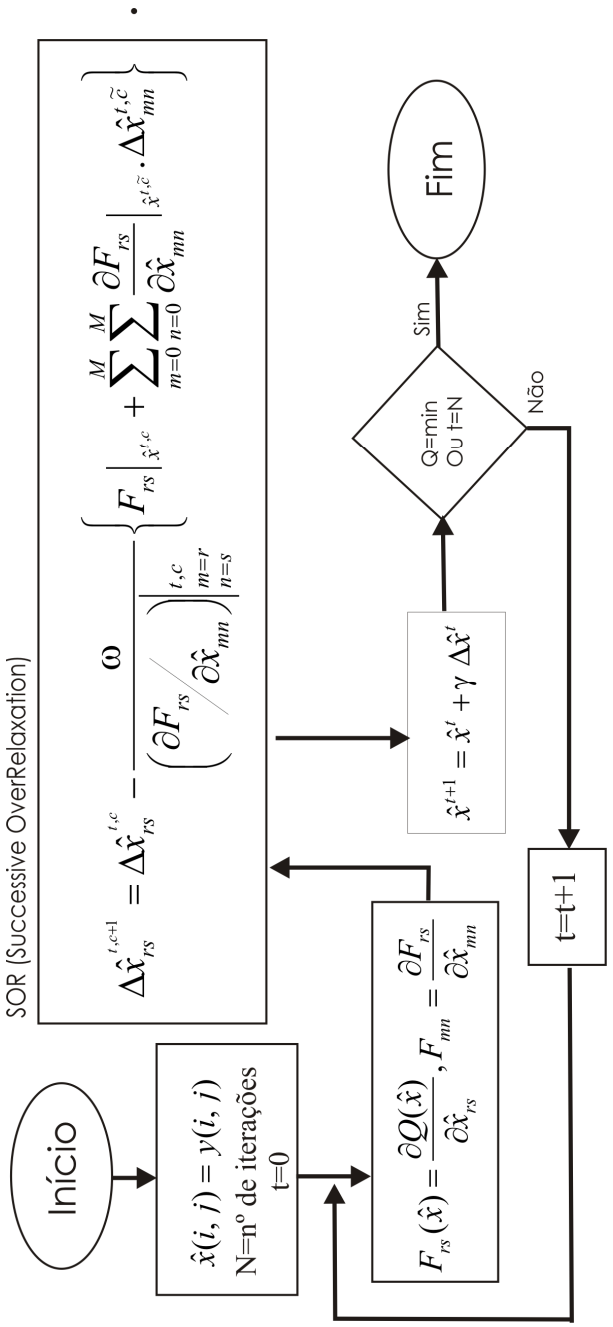


Figura 4.4 - Fluxograma do algoritmo de restauração de imagens de AFM, baseado na minimização do Funcional de Regularização de Tikhonov, empregando SOR (Adaptado de Cidade (2000)).

4.6.3 Critérios de parada

- ***Laço principal***

O laço principal do algoritmo baseado na minimização do Funcional de Regularização de Tikhonov consiste em uma restauração completa da imagem. Esse processo pode ser interrompido:

a) quando todos os pontos da imagem tiverem alcançado uma dada tolerância definida *a priori*:

$$e_1(i,j) = \left| \frac{\hat{x}_{ij}^{t+1} - \hat{x}_{ij}^{t}}{\hat{x}_{ij}^{t+1}} \right| \leq TOL\,, \quad i,j = 0,...,M \text{ , com } \hat{x}_{ij}^{t} \neq 0\,, \tag{50}$$

onde e_1 é o erro relativo das imagens restauradas nos tempos t e $t+1$ e TOL é a tolerância definida *a priori*;

b) quando o processo iterativo alcançar um dado limite máximo de iterações, de modo a evitar loops infinitos; ou

c) quando o valor do funcional Q da imagem recém-restaurada não ultrapassar mais que 5% do menor valor; o que ocorrer primeiro.

- ***Método iterativo de Gauss-Seidel***

Para o método iterativo de Gauss-Seidel usado no cálculo das correções $\Delta\hat{\vec{x}}$, o processo pode ser interrompido:

a) após um número máximo preestabelecido de iterações; ou

b) quando a diferença entre as correções, obtidas nas iterações c e $c+1$, não exceder mais uma dada tolerância, ou seja,

$$e_2 = \left| \frac{\Delta\hat{x}_{rs}^{c+1} - \Delta\hat{x}_{rs}^{c}}{\Delta\hat{x}_{rs}^{c+1}} \right| \leq TOL_{GS}\,, \text{ r,s } = 0,1,2,...,\text{M , com } \Delta\hat{x}_{rs}^{c+1} \neq 0\,, \tag{51}$$

onde e_2 é o erro relativo das correções nas iterações $c+1$ e c e TOL_{GS} é uma tolerância arbitrária, definida *a priori*.

4.7 Parâmetro de regularização α

Um fator crítico na aplicação do funcional Q, Eq. (20), é a escolha do parâmetro de regularização α (Hunt, 1973; Banham e Katsaggelos, 1997). Esse problema tem sido investigado por diversos pesquisadores em vários trabalhos e diversas técnicas têm sido propostas para determinar um valor α ótimo (Katsaggelos, 1989; Katsaggelos et al., 1991; Galatsanos e Katsaggelos, 1992; Oraintara et al., 2000).

4.7.1 Escolha de α para limites conhecidos

De acordo com Miller (1970), dado um conjunto de informações conhecidas *a priori* a respeito do ruído e do sinal, o problema de solução da Eq. (17) pode ser substituído pelo problema de busca de x, que satisfaça as seguintes condições:

$$Q_1 = \{x / \Phi(x, y) \leq \varepsilon\}; \tag{52}$$

e

$$Q_2 = \{x / \Psi(x) \leq E\}, \tag{53}$$

onde Q_1 e Q_2 são dois elipsóides definidos, respectivamente, por $\Phi(x, y) = \|y - Hx\|$ e $\Psi(x) = \|S\|$, que correspondem a primeira e segunda parte do funcional Q, ε uma estimativa da precisão dos dados (norma do ruído) e E uma constante predeterminada.

Dado que Q_1 e Q_2 contêm x, segue-se que x (solução) deve pertencer à interseção dos dois elipsóides,

$$Q_0 = \{x / Q_1 \bigcap Q_2\}, \tag{54}$$

onde Q_0 representa o espaço de soluções para o problema de restauração.

Conforme ilustrado na Fig. 4.5, x_{p1} e x_{p2} são soluções para o problema. Da mesma forma que, se variarmos α continuamente de zero a infinito, teremos a curva pontilhada que passa pelos centros dos elipsóides, x_{Q1} e x_{Q2}. Soluções para o problema podem ser encontradas nos pontos sobre esta curva, dentro da área de interseção Q_0. Observe que x_E, x_ε e x_M, também, são soluções.

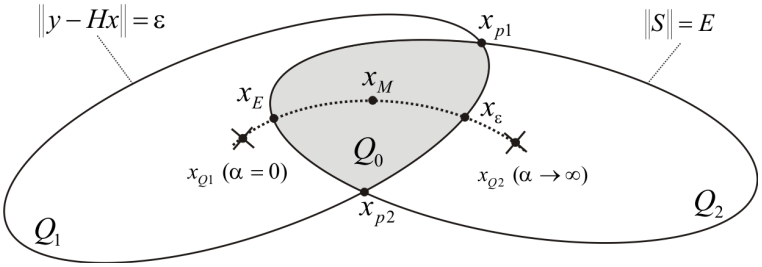


Figura 4.5 - Representação da interseção de dois elipsóides Q_1 e Q_2 (Katsaggelos, 1989).

Dado o conjunto de possíveis soluções, Katsaggelos (1989) propõe o ponto médio x_M (Fig. 4.5) como solução para o problema, aplicando-se a Eq. (20) limitada à intersecção Q_0, fazendo

$$\alpha = \left(\frac{\varepsilon}{E}\right)^2. \tag{55}$$

Para o caso das variâncias do ruído e do sinal serem conhecidas, ou puderem ser estimadas, uma outra escolha seria empregar (Banham e Katsaggelos, 1997)

$$\alpha = \frac{1}{BSNR}, \tag{56}$$

onde o BSNR (*Blurred Signal-to-Noise Ratio*) é dado por

$$BSNR = 10 \cdot \log 10 \left(\frac{\sigma_{BX}^2}{\sigma_\eta^2} \right), \tag{57}$$

onde σ_{BX}^2 e σ_η^2 são, respectivamente, o desvio-padrão da imagem borrada sem ruído e o desvio-padrão do ruído aditivo.

Obtido desta forma, usando a Eq. (56), o valor α é de extrema utilidade e importância, uma vez que ele pode ser empregado na restauração de imagens-teste, onde os limites ε e E são conhecidos.

4.7.2 Processo iterativo de busca do parâmetro α para limites desconhecidos

O conhecimento preciso dos limites de ε e E, entretanto, pode não estar disponível. De fato, esses limites na maioria das vezes não são conhecidos ou não é possível estimá-los diretamente. Para resolver esses casos, várias métodos iterativos para se estimar esses limites e/ou o valor α de têm sido apresentados: a) estimativa dos limites ε e E baseado em imagens restauradas parcialmente (Wallis, 1976; Knutsson et al., 1983; Ortega e Rheinboldt, 1970; Groetsch, 1984); b) estimativa de α associado ao valor ótimo obtido com uma curva-L (Oraintara et al., 2000) – nome do gráfico $\log \Phi(x_\alpha, y)$ versus $\log \Psi(x_\alpha)$, variando-se o parâmetro α de $\alpha \to 0$ até $\alpha \to +\infty$ que tem o formato de uma letra "L" – etc.

Durante o desenvolvimento dessa tese, Stutz et al. (2008, 2009) estudaram e desenvolveram um método iterativo de busca do parâmetro α para o processo iterativo de restauração usando o funcional de Tikhonov, onde se busca encontrar um valor α^* ótimo a cada iteração de forma a se alcançar um valor mínimo para o funcional Q, Eq. (37), na iteração seguinte, aplicando-se α^*. Esse trabalho (Stutz et al., 2009) foi apresentado no II Encontro Acadêmico de Modelagem Computacional no LNCC/Petrópolis, 2009, e recebeu o título de melhor trabalho em Matemática Aplicada e Computacional.

Esse método pode ser introduzido, facilmente, no algoritmo de restauração e usado sempre que os limites ε e E não forem conhecidos, ou quando o valor de α precisar de um certo grau de empirismo do operador que estiver trabalhando com o programa de restauração.

Como esse método foi desenvolvido inicialmente para o algoritmo serial de restauração e como ainda não foi adaptado ao algoritmo paralelo, ele não será empregado nesta tese. Maiores informações a respeito desse método podem ser encontradas em Stutz et al. (2008).

4.8 Tratamento dos pontos próximos aos limites da imagem

Toda imagem digital é limitada espacialmente. Ou seja, uma imagem digital possui um tamanho fixo, com dimensões finitas e bordas bem demarcadas. Como a matriz de borramento B define uma área de influência ao redor de um ponto na imagem (Fig. 4.3c), não é difícil perceber que os pontos próximos das bordas, diferentemente dos demais pontos mais internos, sofrem influências de elementos que extrapolam essas bordas, que existem no plano objeto, mas que não estão registradas na imagem digital (plano imagem) (Fig. 4.6).

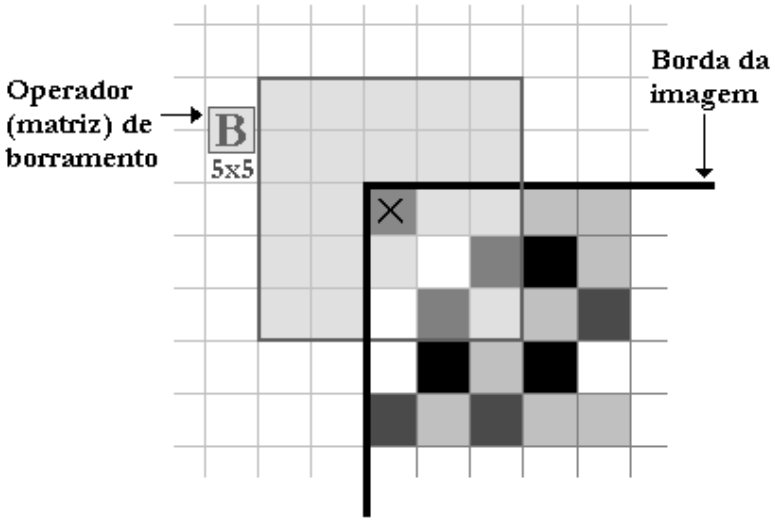


Figura 4.6 - Visualização da área de influência do operador de borramento definida por uma matriz B de dimensão 5x5 ao redor do ponto da imagem, marcado com um "X".

Na Eq. (25) é descrito, de modo geral, como uma imagem experimental é composta ponto a ponto. Entretanto, a equação não faz referência a nenhum tipo de tratamento de como o cálculo deve ser feito para os pontos próximos da borda. Para resolver este problema de

modo que os pontos que faltam sejam representados no cálculo, algumas técnicas podem ser empregadas. Dentre elas, as técnicas comumente empregadas são:

a) **Imagem periódica** – admite-se que os elementos que faltam e que extrapolam uma das margens são obtidos na margem extremamente oposta;
b) **Valor fixo** – adota-se um valor fixo em toda borda que pode ser: um valor arbitrário qualquer, zero (preto), um valor médio ou uma média ponderada dos pixels da imagem; o valor mínimo ou máximo etc.;
c) **Espelhamento** – faz-se um espelhamento dos elementos próximos da borda até o limite área de influência da matriz de borramento B;
d) **Corte** – redefine-se o cálculo de modo a desconsiderar os elementos que não estão representados na imagem.

Na dissertação de mestrado de Stutz (2004) foi feito um estudo da aplicação das técnicas (descritas acima) no processo de restauração. Nesse trabalho, verificou-se que a técnica de corte era a que apresentava os melhores resultados e é, por essa razão, que esse método continuará sendo utilizado neste trabalho.

4.8.1 Técnica de corte

Nessa técnica, os elementos da área de influência ao redor de um ponto na imagem que extrapolam os limites da borda são retirados do cálculo da Eq. (25). Entretanto, para que a condição descrita na Eq. (31) ainda seja satisfeita, a matriz de borramento B, nesses casos, precisa ser ajustada e os valores de seus elementos renormalizados, em conformidade com a quantidade de pontos pertencentes à área de influência e que estão registrados na imagem digital. Na Fig. 4.7 é apresentado um exemplo de ajuste da matriz de borramento B de dimensão 5x5 com variância $\sigma^2 = 20$ para ser usada no cálculo do ponto localizado no canto superior esquerdo da imagem, conforme ilustrado na Fig. 4.6.

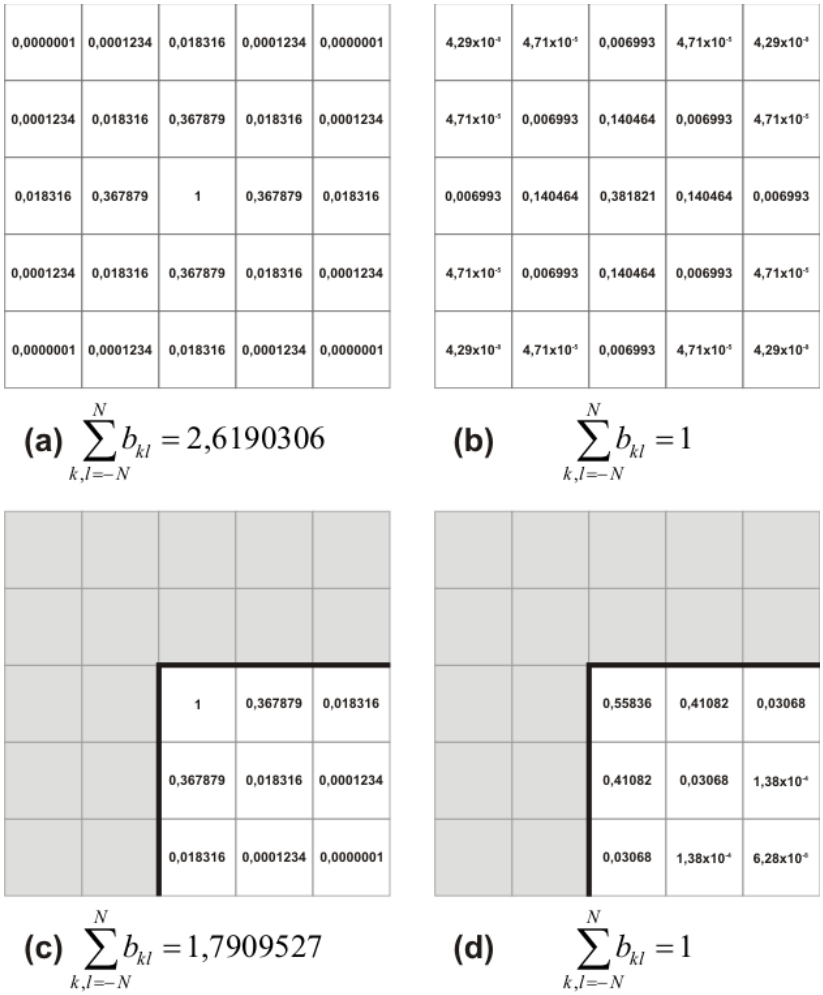


Figura 4.7 - Exemplo de uma matriz de borramento B de dimensão 5x5 com $\sigma^2 = 20$ (a) matriz completa com valores não normalizados, (b) matriz completa normalizada, (c) matriz parcial modificada e ajustada para o pixel do canto superior direito da imagem e (d) matriz parcial modificada normalizada.

4.9 Técnica de realimentação

Na Eq. (36), o termo $\bar{x}$ é um valor de referência, usado no cálculo da função de regularização S, que pode ser definido como sendo: a) a própria imagem obtida experimentalmente pelo AFM; b) uma imagem plana, onde as intensidades dos pixels assumam um valor constante qualquer ou a média das intensidades dos pixels da imagem, por exemplo. Entretanto, independente de qual seja o valor de referência empregado, este permanece fixo durante todo o processo de restauração (Fig. 4.8).

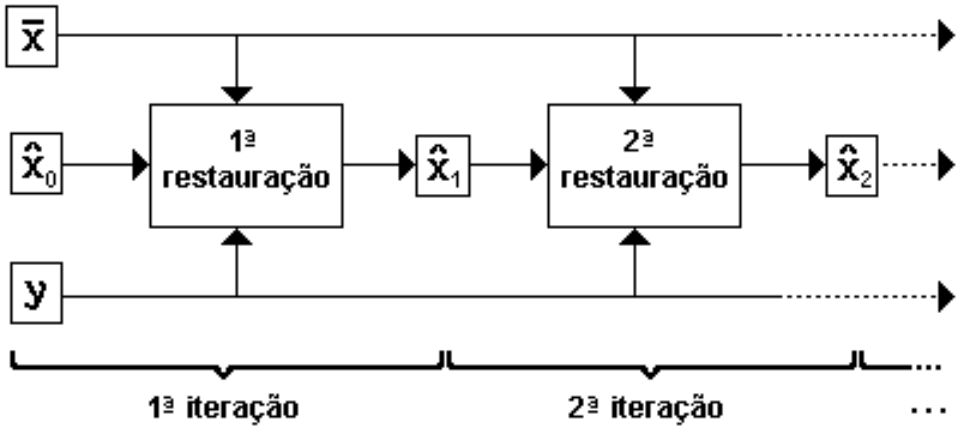


Figura 4.8 - Ciclo de iterações do processo de restauração de imagens, usando uma referência fixa.

Entretanto, Furtado (2002) em sua dissertação de mestrado, propõe uma modificação nessa técnica que ficou denominada como **técnica de realimentação**. Conforme ilustrado na Fig. 4.9, a técnica consiste basicamente em aplicar uma imagem inicial de referência qualquer em $\bar{x}_0$ (p.ex. a própria imagem que está sendo restaurada) e, a cada início de um novo ciclo de iteração, aplicar a imagem restaurada no ciclo anterior ($\hat{x}_{i-1}$) como referência para o próximo ciclo de restauração ($\bar{x}_i = \hat{x}_{i-1},\ i = 1,2,...$).

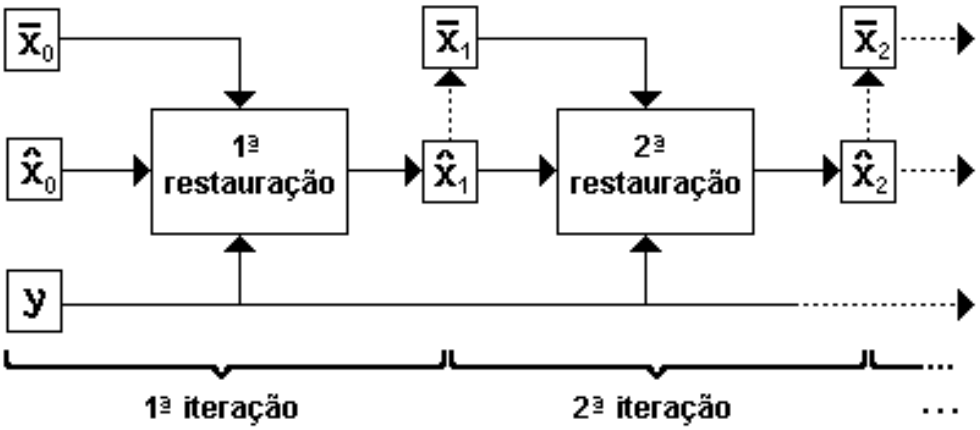


Figura 4.9 - Ciclo de iterações do processo de restauração, usando realimentação.

Furtado (2002) descobriu que a aplicação da técnica de realimentação acelerava o processo de restauração, fazendo com que a execução do programa alcançasse o valor mínimo

em um número menor de iterações. Disto resulta que o tempo de processamento do programa de restauração usando realimentação é bem menor do que o tempo de processamento gasto, usando uma imagem de referência fixa (Fig. 4.10). Apesar de não se ter ainda uma explicação matemática para esse fenômeno, o emprego da realimentação no processo de restauração de imagens tem-se mostrado muito útil, uma vez que um dos objetivos buscados nesse trabalho é exatamente alcançar uma maior eficiência na execução do algoritmo de restauração de imagens, gastando-se um tempo de processamento menor.

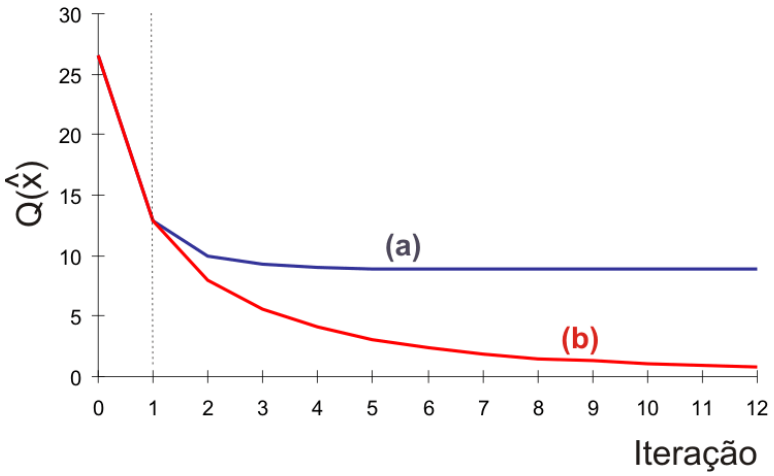


Figura 4.10 - Gráfico da evolução dos mínimos quadrados obtidos, a cada iteração do programa, restaurando-se uma mesma imagem (a) usando como referência uma imagem fixa e (b) empregando-se realimentação.

Para efeito de exemplo e comparação, na Fig. 4.11 são ilustradas algumas restaurações da imagem-padrão modificada (D5v20s40) (Fig. 4.11a) usando a técnica de realimentação (Fig. 4.11b) e outra sem aplicá-la (Fig. 4.11c). Como referência $\bar{x}$, empregou-se nas restaurações a própria imagem-padrão modificada. Como o processo de realimentação só se aplica a partir da segunda restauração, os valores apresentados ao lado das duas restaurações na primeira iteração (primeira restauração) serão exatamente iguais. Só após a segunda restauração, percebe-se uma variação no valor do funcional Q, Eq. (37), obtido com/sem a aplicação da realimentação. Junto às imagens restauradas são mostradas também algumas medidas de qualidade (apresentadas no próximo capítulo), apontando uma pequena melhora, a cada iteração, na qualidade das imagens restauradas usando a realimentação, em relação às mesmas restaurações feitas na imagem borrada, porém, sem aplicar essa técnica.

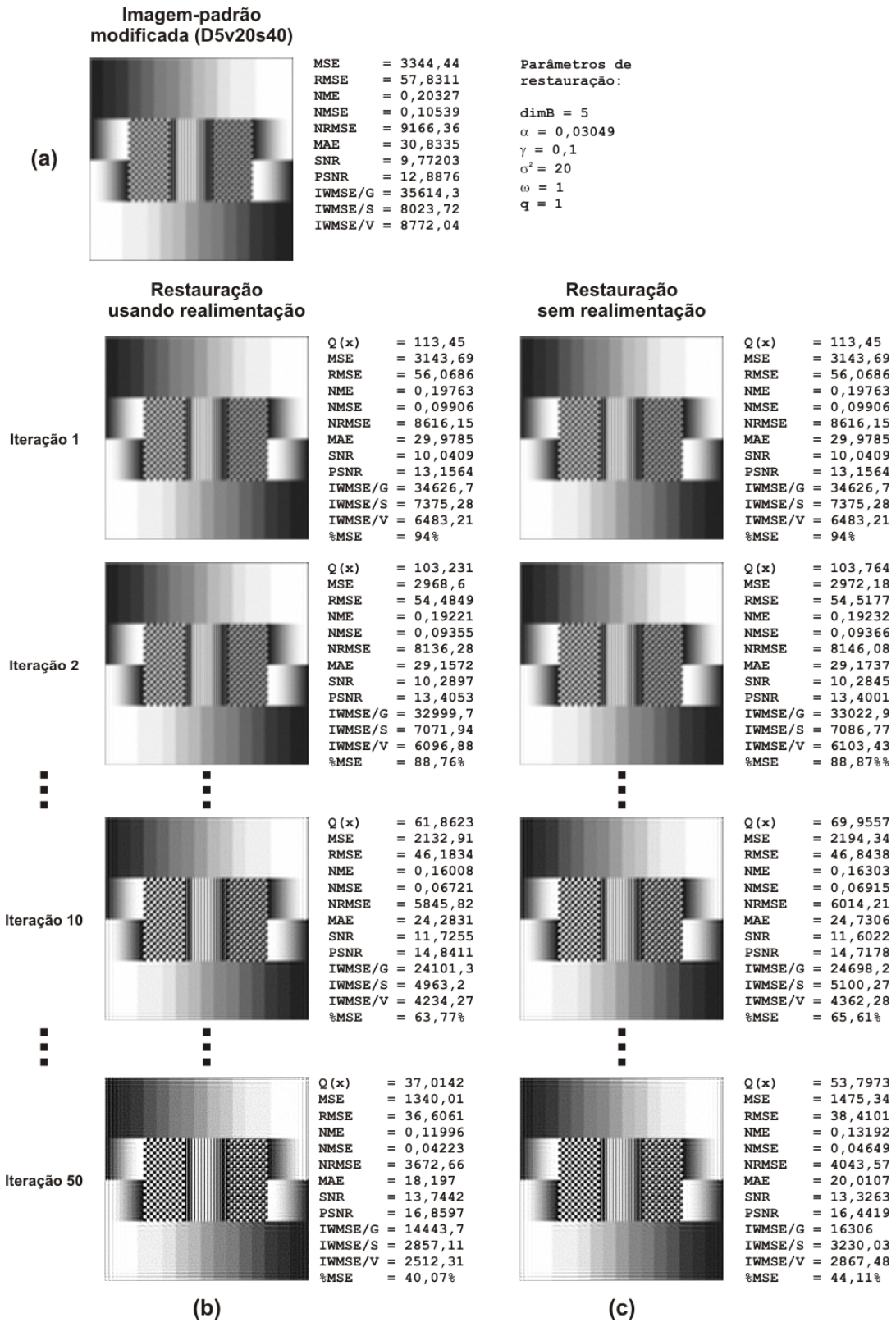


Figura 4.11 - Restaurações da imagem-padrão modificada – D5v20s40 (a), usando a técnica de realimentação (b) e sem aplicá-la (c).

5 MÉTRICAS DE AVALIAÇÃO

5.1 Medidas de avaliação de resultados e da qualidade das imagens

Em processamento de imagens (restauração, compressão, realce etc.), a existência de métricas que permitam a atribuição de um valor que expresse a qualidade de uma imagem é de grande importância. Essas medidas, de modo geral, são comumente usadas tanto para determinar a extensão ou a grandeza da diferença entre duas imagens quanto para se avaliar o desempenho e/ou a eficácia de um algoritmo computacional e a sua validação. Por exemplo, podemos comparar uma imagem distorcida e uma imagem restaurada por um processo de restauração qualquer com a imagem original para determinar a eficácia do método empregado, assim como podemos também utilizar os resultados produzidos pelo processo de restauração e compará-los com os de outros métodos, de modo a determinar qual deles apresenta o melhor resultado. Geralmente, o método mais empregado e discutido na literatura tem sido o erro médio quadrático (Weisman et al., 1992).

5.1.1 Erro Médio Quadrático - MSE (Mean Square Error)

O MSE consiste num valor simples, resultante do somatório dos quadrados das diferenças pontuais das intensidades em escala de cinza dos pontos (pixels) de cada imagem, dado pela seguinte equação:

$$MSE = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\left[f(i,j) - \tilde{f}(i,j)\right]^2}{N_t}, \tag{58}$$

onde $f(i,j)$ representa a intensidade do ponto (pixel) de coordenada (i,j) da imagem original/referência de dimensão $m \times n$, $\tilde{f}(i.j)$ a intensidade do ponto de mesma coordenada em uma outra imagem que se está comparando e $N_t = m \times n$ o número total de pixels em cada uma dessas imagens.

Além do MSE, outras medidas similares podem ser empregadas na análise de imagens (Bevilacqua e Picollomini, 2000), tais como:

a) Erro médio normalizado ou NME (*Normalized Mean Error*)

$$NME = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} \left| f(i,j) - \tilde{f}(i,j) \right| \right) \Big/ \left(\sum_{i=1}^{m} \sum_{j=1}^{n} \left| f(i,j) \right| \right); \tag{59}$$

b) Raiz do erro médio quadrático normalizado ou NRMSE (*Normalized Root Mean Square Error*)

$$NRMSE = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} \left[f(i,j) - \tilde{f}(i,j) \right]^2 \right) \Big/ \left(\sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} \left(f(i,j) - \bar{f} \right)^2} \right), \tag{60}$$

onde $\bar{f}$ é a média aritmética dos pontos da imagem original $f(i,j)$.

Uma ampla relação e uma classificação das medidas de qualidade que aparecem na literatura podem ser encontras em Eskicioglu e Fisher (1995) e em Pedrini e Schwartz (2008). No trabalho de dissertação de Romualdo (2006), alguns indicadores de desempenho para o algoritmo de restauração de imagens foram usados e avaliados por um grupo de 50 pessoas com o objetivo de investigar e determinar qual deles se aproximava mais das avaliações feitas pelos avaliadores.

- ***Erro médio quadrático relativo (%MSE)***

Nos casos em que a imagem original $f(i,j)$ é conhecida, pode-se calcular também o erro médio quadrático relativo (%MSE) das restaurações, de modo a obter um valor normalizado em relação ao MSE da imagem modificada (Cidade, 2000).

$$\%MSE = \frac{MSE_{\text{imagem restaurada}}}{MSE_{\text{imagem modificada}}}. \tag{61}$$

Essa informação é extremamente útil, pois ela nos dá uma medida do ganho da restauração com base no MSE. Quanto menor for o valor do %MSE, melhor esse ganho.

5.1.2 Erro Médio Absoluto - MAE (Mean Absolute Error)

O MAE é o valor resultante do somatório das diferenças absolutas pontuais das intensidades em escala de cinza de cada ponto (pixels) de duas imagens (p.ex. original e distorcida) dividido pelo total de pontos de cada imagem, dado pela seguinte equação:

$$MAE = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\left| f(i,j) - \tilde{f}(i,j)\right|}{N_t}, \tag{62}$$

onde $f(i,j)$ representa a intensidade do ponto (pixel) de coordenada (i,j) da imagem original (ou uma imagem de referência) de dimensão $m \times n$, $\tilde{f}(i.j)$ a intensidade do ponto de mesma coordenada em uma outra imagem a qual se quer comparar (p.ex. uma imagem resultante de um processo de restauração) e $N_t = m \times n$ o número total de pixels em cada uma dessas imagens.

Quanto menor essa métrica, mais a imagem que se está comparando ($\tilde{f}$) se aproxima da original f .

5.1.3 Relação sinal-ruído – SNR (Signal to Noise Ratio)

$$SNR = 10\log_{10}\frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\left[f(i,j)\right]^2}{\sum_{i=1}^{m}\sum_{j=1}^{n}\left[f(i,j) - \tilde{f}(i,j)\right]^2}, \tag{63}$$

onde $f(i, j)$ representa a intensidade do ponto (pixel) de coordenada (i, j) da imagem original/referência de dimensão $m \times n$, $\tilde{f}(i.j)$ a intensidade do ponto de mesma coordenada em uma outra imagem a qual se quer comparar.

5.1.4 Relação sinal-ruído de pico – (Peak Signal to Noise Ratio)

$$PSNR = 10 \log_{10} \frac{(L_{\max})^2}{MSE}, \tag{64}$$

onde $L_{\max}$ é o valor máximo de intensidade de cinza (tipicamente, $L_{\max} = 255$) e MSE é o erro médio quadrático.

A PSNR é expressa em decibel (dB) e quanto maior for o valor dessa métrica, mais a imagem que se está comparado se aproxima da original.

5.1.5 Método de avaliação baseado na percepção - IWMSE (Information Weighted Mean Square Error)

Normalmente, os métodos convencionais para avaliação da qualidade de imagens têm como base critérios objetivos obtidos essencialmente por meio de métricas apoiadas em erros estatísticos, tais com: MSE, NME, NRMSE, entre outros. Entretanto, os resultados obtidos através desses métodos (objetivos) diferem bastante de critérios subjetivos que tenham como base características do sistema visual humano (HVS - *Human Visual System*) (Ming J. et al., 2008), por exemplo.

Tompa et al. (2000) afirmam que o MSE, assim como os outros métodos convencionais, geralmente falha ao fornecer uma medida da diferenças que não representa corretamente o nível de qualidade observada entre duas imagens. Eles explicam que isso decorre em muitos casos principalmente pelo fato do MSE atribuir um mesmo peso às diferenças em todos os pontos da imagem. Ao analisar uma imagem, entretanto, percebemos que nem todos os pontos são igualmente importantes. Ou seja, nem todos os pontos em uma imagem possuem o mesmo peso para um observador humano. Diferenças perceptíveis em

regiões mais importantes da imagem irão produzir um efeito negativo muito maior do que aquelas (diferenças) encontradas em outras áreas como o fundo, por exemplo. Para um observador humano é importante que a medida de avaliação corresponda a um critério subjetivo de comparação e expresse a qualidade visual percebida.

Apesar da complexidade do sistema visual humano, entretanto, à medida que os modelos visuais vão sendo introduzidos nos métodos de avaliação objetivos, o relacionamento entre os métodos objetivos e subjetivos vão se aperfeiçoando e os resultados de avaliação tendem a melhorar, alcançando uma melhor correlação com a resposta dos observadores humanos (Eskicioglu, 2000). Atualmente, muitos trabalhos e pesquisas científicas em diversas áreas do conhecimento têm dado uma maior ênfase aos métodos que se baseiam no sistema visual humano. Recentemente, trabalhos em visão computacional e de compressão de imagens, por exemplo, têm aceitado as novas abordagens de avaliação, baseadas na percepção/visão humana, como sendo superiores às abordagens convencionais (Damera-Venkata et al., 2000).

- ***Medidas de energia de interesse local***

Apesar do sistema visual humano ser extremamente seletivo, verificou-se que as áreas de uma imagem que recebiam uma maior atenção (interesse visual) eram aquelas que continham alguma característica específica, tal como, por exemplo, uma curvatura elevada, um alto contraste, linhas e bordas, a ocorrência de algum aspecto inesperado etc. Assim, tomando-se como base alguma dessas peculiaridades, o nível de interesse visual local pode ser medido. Dentre os vários aspectos (Fig. 5.1), alguns deles foram identificados e apresentados por Topper (1991): (a) o valor do pixel em uma escala de cinza (intensidade); (b) variância local (contraste); (c) magnitude de borda, obtida com o operador de detecção de bordas de Sobel (Gonzalez & Woods, 2000) etc.

Entretanto, essas medidas de energia, por si só, não são capazes de ponderar diretamente as diferenças tal como se tem teorizado a respeito do nível de interesse visual, pois o sistema visual humano seleciona a porção do estímulo visual (ou energia) que é menos esperado ou, em termos estatísticos, aquilo que é menos provável (Mackworth & Morandi, 1967).

(a) (b)

Figura 5.1 - Medidas de energia de interesse local: (a) intensidade (escala de cinza) e (b) magnitude de borda (imagem gradiente obtida com a utilização do operador de detecção de bordas de Sobel).

- ***Mapa de informações***

Em alguns trabalhos (Tompa et al., 2000; Wong & Vogel, 1977; Topper, 1991) é mostrado que existe uma relação direta de proporcionalidade entre a medida de informação, também conhecida como auto-informação (*Shannon's self-information*) (Shannon, 1948), e o nível de interesse visual humano com relação as diferentes áreas em uma imagem. Isso foi feito através de experimentos em que se fez o mapeamento das áreas de interesse visual em uma imagem a partir do acompanhamento dos movimentos dos olhos das pessoas quando elas observam essa mesma imagem e, a partir disso, verificou-se que a as áreas de maior fixação eram justamente as áreas da imagem que continham os maiores valores de informação. Assim, ao invés de usar as energias diretamente, a medida de informação calculada com base em uma das medidas de energia de interesse local poderia ser usada no seu lugar para se ponderar as diferenças entre as imagens.

Tomando-se, por exemplo, o valor da intensidade do pixel como medida de energia de interesse local, o cálculo da medida de informação baseia-se num esquema probabilístico dado por

$$P(e) = \frac{n_e}{N_t}, \tag{65}$$

onde $P(e)$ é a probabilidade de ocorrência de um dado evento e (p.ex. valor de cinza), n_e o número de vezes que este valor aparece (histograma) e N_t o número total de pontos da imagem.

Como o nível de interesse visual aumenta à medida que a probabilidade de ocorrência de um dado evento diminui, a relação entre os diferentes eventos e a informação contida em cada um deles é dada da seguinte forma:

$$I_{si}(e) = \log \frac{1}{P(e)}, \tag{66}$$

onde $I_{si}(e)$ é a medida de informação (ou auto-informação) de um dado evento e.

Esta medida (auto-informação) fornece um valor não-negativo que representa a quantidade de informação contida em um evento e ele pode ser definido não somente em função dos valores em cada pixel (intensidade), mas também em qualquer uma das medidas de energia local citadas anteriormente.

Assim, através das medidas de informação obtidas pela a Eq. (66), constrói-se um mapeamento da imagem, substituindo-se cada um dos valores associados ao evento e pela sua respectiva medida de informação. Esse mapeamento é referido como mapa de informações, uma vez que mostra a localização e a quantidade de informação contida numa imagem.

$$I_{map}(i, j) = I_{si}(En(i, j)), \tag{67}$$

onde I_{map} é o mapa de informações, En o mapa de energias da imagem que está sendo mapeada, $I_{si}(.)$ é a medida de informação para um dado evento e (i, j) coordenadas da imagem.

- ***IMSE (Information Mean Square Error)***

Tompa et al. (2000) descrevem um novo método inspirado na percepção humana, o IMSE (*Information Mean Square Error*), em substituição aos métodos convencionais. Essa nova medida tem as diferenças ponderadas como base no mapa de informações de cada imagem, de modo a garantir que os elementos perceptíveis (pontos da imagem com valores maiores de informação e de maior interesse visual) tenham pesos maiores que os demais elementos da imagem.

$$IMSE = \frac{1}{N_t}\sum_{i=1}^{m}\sum_{j=1}^{n}\left[f_{i,j}I^{f}_{map}(i,j) - \tilde{f}_{i,j}I^{\tilde{f}}_{map}(i,j)\right]^2, \tag{68}$$

onde $f_{i,j}$ representa a intensidades no ponto de coordenada (i, j) de uma imagem (original ou de referência) com dimensão $m \times n$, $\tilde{f}_{i,j}$ o ponto de mesma coordenada de uma imagem distorcida ou resultante de um processo de restauração, por exemplo, $N_t = m \times n$ o número total de pontos (pixels) de cada uma destas imagens, e I^{f}_{map} e $I^{\tilde{f}}_{map}$ são, respectivamente, os mapas de informação das imagens original e distorcida.

A parte mais importante do uso do IMSE é selecionar uma medida de energia apropriada para se avaliar as diferenças visuais perceptíveis entre as duas imagens. Em seu artigo, Tompa et al. (2000) deixa em aberto qual medida deve ser empregada e afirma que essa escolha depende da imagem que se está sendo trabalhada e do método de distorção aplicado.

Analisando-se de forma mais rigorosa a fórmula de cálculo do IMSE, Eq. (68), entretanto, percebe-se que existem alguns problemas com relação à forma como o cálculo das diferenças é feita:

Diferenças onde existem semelhanças:

Mesmo que as intensidades dos pontos sejam iguais nas duas imagens ($f_{i,j} - \tilde{f}_{i,j} = 0$), eles poderão apresentar "diferenças" consideráveis no IMSE caso exista alguma diferença em seus mapeamentos naquele ponto ($I^{f}_{map}(i,j) - I^{\tilde{f}}_{map}(i,j) \neq 0$). Isto não faz sentido, pois se as

intensidades são iguais (perceptíveis ou não), então não existe diferença entre elas. Para esses casos, a diferença deveria ser zero.

Falsa igualdade para imagens monocromáticas diferentes:

Tomando-se, por exemplo, uma imagem totalmente branca ($W = \{W_{i,j} = 255\}$) e uma outra totalmente preta ($P = \{P_{i,j} = 0\}$), tem-se que as medidas de informação para ambas cores, branca e preta, são iguais a zero ($I_{si}^{W}(255) = I_{si}^{P}(0) = 0$), uma vez que as probabilidades de cada uma delas em cada uma das imagens é igual a 1. Disso resulta que, para imagens monocromáticas diferentes, o IMSE será igual a zero, indicando que as "imagens são iguais" quando na verdade não são.

- ***IWMSE (Information Weighted Mean Square Error)***

Para resolver os problemas descritos acima, é proposta uma alteração no cálculo do IMSE, Eq. (68), de forma a criar um novo método de calculo das diferenças baseado nas medidas de informação, corrigindo as deficiências do IMSE e que acabou resultando numa nova medida aqui denominada "Erro Médio Quadrático Ponderado pela Informação" ou simplesmente IWMSE (Stutz et al., 2007), dado por

$$IWMSE = \frac{1}{N_t}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(1 + I_{map}^{\tilde{ff}}(i,j)\right)\left[f_{i,j} - \tilde{f}_{i,j}\right]^2 \tag{69}$$

onde $I_{map}^{\tilde{ff}}(i,j) = \max(I_{map}^{f}(i,j), I_{map}^{\tilde{f}}(i,j))$ é o valor máximo dos mapas de informação das imagens original e distorcida na coordenada (i, j).

Na Eq. (69), verifica-se que os pontos iguais (perceptíveis ou não) nada acrescentam à medida uma vez que a diferença entre eles resulta em zero. Ou seja, as diferenças ocorrem somente onde não existem semelhanças.

O valor do mapeamento aplicado é dado em função do valor máximo dos mapas de informação das imagens original e distorcida de modo a refletir o nível de interesse máximo em cada um dos pontos em ambas as imagens. O peso é então calculado com base neste valor acrescido de 1 de modo a evitar que imagens monocromáticas diferentes resultem em uma

falsa igualdade. Para esses casos, o valor calculado do IWMSE será igual ao do MSE visto que para $I_{map}^{f\tilde{f}} = \{I_{map}^{f\tilde{f}}(i, j) = 0\}$ a equação do IWMSE, Eq. (69), se iguala ao MSE, Eq. (58).

Conforme ilustrado na Fig. 5.2, o processo de cálculo do IWMSE é implementado da seguinte forma:

(1) Filtram-se as imagens original $f_{i,j}$ e distorcida $\tilde{f}_{i,j}$ usando um operador ponto exponencial, Eq. (70), de modo a reduzir a sensibilidade do algoritmo a pequenas variações que por acaso existam nas regiões escuras da imagem (Tompa et al., 2000);

$$\bar{f}_{i,j} = d(b^{f_{i,j}} - 1), \tag{70}$$

onde b é uma base qualquer, d é um fator de correção de escala e $f_{i,j}$ e $\bar{f}_{i,j}$ são respectivamente os pontos de coordenadas (i, j) das imagens original e filtrada.

O valor da constante d é escolhido de modo que R, o valor de maior magnitude em $f(i, j)$, seja 255. Portanto,

$$d = \frac{255}{\log(1+|R|)}. \tag{71}$$

(2) Usando-se uma das medidas de energia local e (escolhida a priori), constroem-se os mapas de energias En^{f} e $En^{\tilde{f}}$ das imagens filtradas em (1);

(3) A partir desses mapas (2), constroem-se os seus respectivos histogramas n_e^f e $n_e^{\tilde{f}}$;

(4) Para cada um dos histogramas (3), calculam-se as funções de densidade de probabilidade $P_f(e)$ e $P_{\tilde{f}}(e)$, Eq. (65) e a seguir,

(5) calcula-se as medidas de informação $I_{si}^{f}(e)$ e $I_{si}^{\tilde{f}}(e)$ de cada um dos eventos e, Eq. (66) e, finalmente,

(6) constroem-se os mapas de informações, substituindo-se os valores associados ao evento e na medida de energia (2) pela sua medida de informação (5), para serem usados depois no cálculo do IWMSE, Eq. (69).

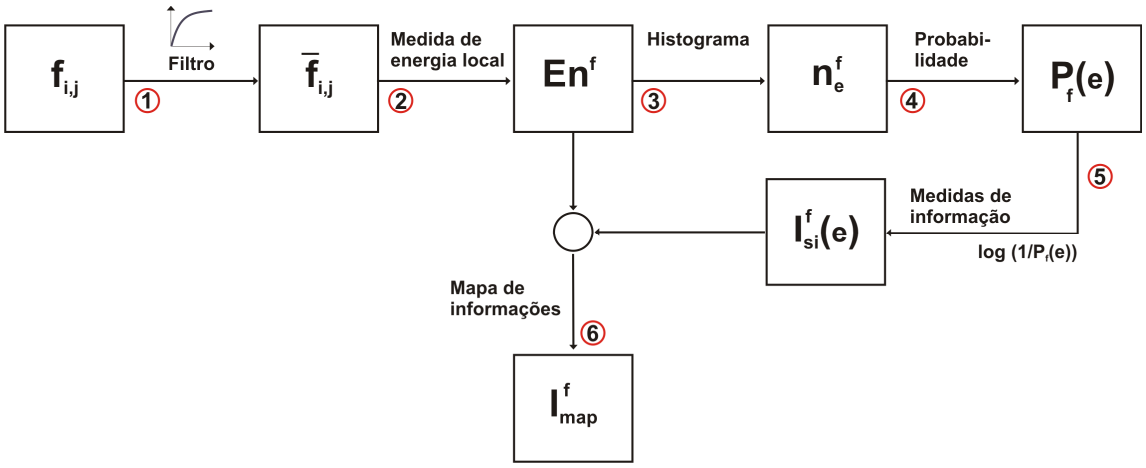


Figura 5.2 - Fluxograma do processo de construção do mapa de informações da imagem $f_{i,j}$ usado no cálculo do IWMSE.

- ***Exemplo-teste***

Devido à baixa resolução das imagens disponibilizadas no artigo de Tompa et al. (2000) não foi possível reproduzir os testes apresentados em seu artigo, nem utilizá-los de modo que o IMSE e o IWMSE pudessem ser comparados. Para se resolver esse problema, alguns testes foram feitos em que uma imagem era borrada e restaurada logo a seguir, até se obter um bom exemplo onde as diferenças entre as imagens borrada e restaurada pudessem ser notadamente percebidas e o MSE falhasse (Fig. 5.3).

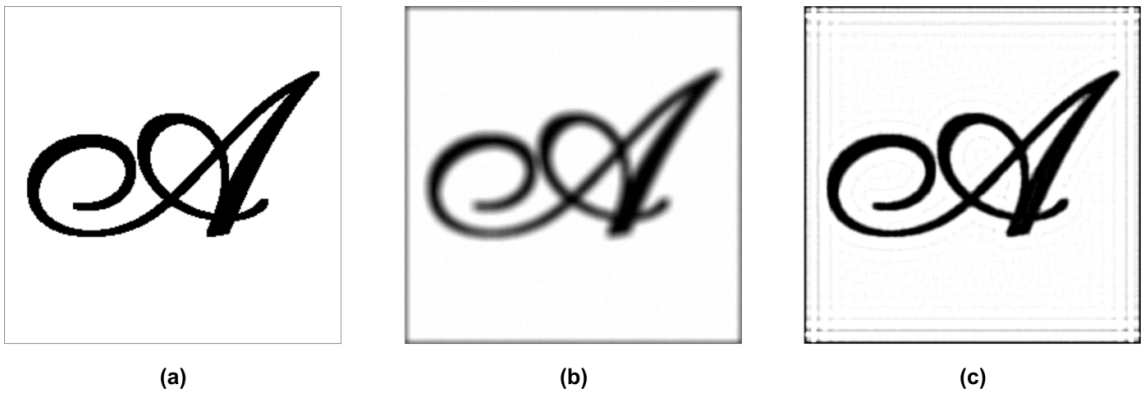


Figura 5.3 - (a) Imagem original, (b) distorcida e (c) restaurada.

Para distorcer a imagem original, foi utilizado um operador de borramento do tipo gaussiano, Eq. (30), de dimensão 5x5 e variância $\sigma^2 = 20$ e, logo a seguir, adicionados ruídos aditivos de 30 dB. Essa imagem (distorcida) foi, então, restaurada usando o funcional de regularização de Tikhonov apresentado por Cidade et al. (2000), empregando-se os mesmos parâmetros que foram usados no borramento.

Na Tabela 5.1, os resultados obtidos com as medidas MSE, IMSE e IWMSE aplicadas ao exemplo-teste (Fig. 5.3) são apresentados. As medidas de energia local usadas nos cálculos do IMSE e do IWMSE foram: a) escala de cinza (/G), b) magnitude de borda (/S) e variância local (/V). Pode-se observar que os valores de MSE para as duas imagens são praticamente iguais, apesar da imagem restaurada (Fig. 5.3c) ser visualmente (perceptivelmente) melhor do que a imagem borrada (Fig. 5.3b).

Tabela 5.1 – Diversas medidas de diferenças das imagens distorcida e restaurada em relação à imagem original.

		IMSE			IWMSE		
IMAGENS	**MSE**	**IMSE/G**	**IMSE/S**	**IMSE/V**	**IWMSE/G**	**IWMSE/S**	**IWMSE/V**
Distorcida	773,62	84284,6	3769,8	2258,4	7568,5	820,1	791,5
Restaurada	773,60	59710,1	5245,3	1589,6	6403,9	137,3	622,5

Legenda - Medidas de energias locais: intensidade (G=Greyscale), magnitude de borda (S=Sobel) e variância (V).

As medidas do IMSE não são conclusivas devido às discrepâncias entre os valores calculados: o IMSE/G e IMSE/V apontam melhoras de qualidade na imagem restaurada, enquanto que o IMSE/S indica que a imagem distorcida é a melhor. Tompa et al. (2000) citam em seu artigo que a escolha da medida de energia depende da imagem e do método de distorção aplicado. Entretanto, com relação ao IWMSE, todas as medidas apontaram uma melhora na qualidade da imagem restaurada. Porém, apesar do bom resultado apresentado pelo IWMSE, por precaução serão admitidas as mesmas considerações que foram feitas para o IMSE com relação à escolha da medida de energia.

- ***Uso do IWMSE como método alternativo ao MSE***

O MSE é o método mais comumente empregado e discutido na literatura, entretanto, dada a importância que se tem hoje do uso de métodos subjetivos na avaliação de resultados em processamento de imagens, esse trabalho irá empregar também o IWMSE como método alternativo ao MSE na avaliação dos resultados de restauração. Dessa forma, mesmo sendo um método novo, o IWMSE poderia ser exaustivamente testado e avaliado na prática.

5.2 Medidas de desempenho computacional dos programas em paralelo

5.2.1 Tempos de execução

Em termos gerais, o tempo de execução serial (T_S) é compreendido como sendo o tempo decorrido do início até o término da execução de um programa em um computador sequencial. Porém, dado a existência de arquiteturas paralelas de computadores, pode-se estender a definição de tempo serial como sendo o tempo obtido em um computador onde cada instrução de um programa precisa ser concluída primeiro, antes da próxima instrução ser iniciada, sequencialmente, usando apenas uma unidade de processamento.

O tempo de execução de um programa em paralelo (T_P) é compreendido como sendo o tempo decorrido do início em que o programa é carregado até o momento em que o último processador termina a sua execução.

Apesar de ser uma medida extremamente simples, o tempo de execução nem sempre é a métrica mais conveniente para se avaliar o desempenho de programas paralelos. Como os tempos de execução podem variar de acordo com diversas condições (o tamanho do problema a ser resolvido, volume de dados, velocidade do processador, velocidade de barramento etc.), eles precisam ser normalizados de modo que as diferentes dimensões do problema possam ser comparadas e avaliadas com relação à sua eficiência (Foster, 1994).

Quando se discute a análise de desempenho computacional de programas paralelos, não nos preocupamos apenas com os tempos de execução, frequentemente estamos mais interessamos em saber o quanto estamos ganhando com a paralelização de uma dada aplicação em relação a sua implementação sequencial. Para isso, faz-se necessário o emprego de outras medidas que determinem e quantifiquem os seus rendimentos. Dentre essas medidas, as mais comuns são: a) aceleração, b) eficiência e c) custo.

5.2.2 Aceleração (speedup)

A aceleração é uma medida que expressa a melhoria de desempenho de um programa quando é feita alguma modificação no hardware/software ou quando se aplica alguma técnica computacional diferente e/ou otimizada em sua implementação. Essa medida faz referência ao quanto um algoritmo paralelo é tão mais rápido do que o seu correspondente serial.

Portanto, define-se a aceleração como sendo a relação dos tempos obtidos por um mesmo programa antes e depois de alguma modificação, realizando o mesmo trabalho computacional sobre um mesmo conjunto de dados. Quando essa melhoria é dada somente em função do aumento do número de unidades de processamento, normalmente a aceleração é calculada da seguinte forma (1º caso):

$$S_p = \frac{T_p(v_d, 1)}{T_p(v_d, np)}, \tag{72}$$

onde S_p é a aceleração (*speedup*), $T_p(v_d, 1)$ e $T_p(v_d, np)$ são os tempos de processamento de um programa em paralelo, realizando um trabalho computacional sobre um mesmo volume de dados v_d, empregando, respectivamente, 1 e np processadores.

O valor da aceleração obtida com a Eq. (72) é também conhecido como **aceleração relativa** (*relative speedup*), pois ele é definido em função do tempo de execução do programa paralelo usando apenas um processador. Esse valor é útil quando se está analisando a escalabilidade de um algoritmo em paralelo. Entretanto, o valor obtido com a Eq. (72) nem sempre é muito bem aceito na literatura, pois ele permite que os valores sejam disfarçados, mascarando-se um "aumento" da aceleração (Schlemer, 2003). Tome-se, por exemplo, o caso em que um algoritmo paralelo seja otimizado para um número específico de processadores: oito, por exemplo. É óbvio que o desempenho do programa usando os oito processadores sempre apresentará um desempenho melhor do que quando se emprega um processador apenas. Assim, de modo a evitar subterfúgios e aumentos fictícios de aceleração, o cálculo da Eq. (72) é modificado para (2º caso):

$$S_p^{'} = \frac{T_s(v_d)}{T_p(v_d, np)}, \tag{73}$$

onde $S_p^{'}$ é a **aceleração absoluta** (*absolute speedup*) calculado, usando o tempo de processamento $T_s(v_d)$ obtido com um programa serial otimizado e que empregue o melhor algoritmo serial, e $T_p(v_d, np)$ o tempo alcançado com o programa paralelo, usando np processadores. Em muitos casos, esse "melhor" algoritmo deverá ser o algoritmo serial mais rápido conhecido.

5.2.3 Eficiência

A eficiência representa uma fração do tempo de execução que os processadores gastam para realizar o seu trabalho e fornece uma medida conveniente para se avaliar a qualidade em termos de rendimento dos programas em paralelo, independente do tamanho do problema abordado. O cálculo da eficiência de um programa paralelo pode ser feito, empregando-se as seguintes equações:

(a) para o primeiro caso, Eq. (72), ou **eficiência relativa***:*

$$E_p = \frac{S_p}{np}; \tag{74}$$

(b) para o segundo caso, Eq. (73), ou **eficiência absoluta***:*

$$E_p^{'} = \frac{S_p^{'}}{np}, \tag{75}$$

onde np corresponde ao número total de processadores usado na execução do programa em paralelo.

5.2.4 Custo de um sistema paralelo

O custo de um sistema serial (C_s) é basicamente o tempo de execução que se gasta para resolver um dado problema, usando apenas um único processador e empregando o algoritmo serial mais rápido conhecido. Para sistemas paralelos, o custo (*work* ou *processor-time product*) é definido como sendo o produto do tempo de execução em paralelo e o número de processadores.

$$C_p = T_p \times np\,, \tag{76}$$

onde C_p é o custo de um sistema em paralelo, T_p o tempo de execução em paralelo e np o número de processadores.

Essa última medida (C_p) reflete a soma dos tempos que cada um dos processadores gasta para se resolver um dado problema em paralelo e é usado, normalmente, em análises comparativas de desempenho de algoritmos paralelos e sequenciais.

A eficiência (relativa) de um sistema paralelo pode ser expressa também em função da razão:

$$E_p^{'} = O\left(\frac{C_s}{C_p}\right), \tag{77}$$

onde $O(\cdot)$ é a eficiência usando a notação de análise assintótica, também conhecida como notação “grande-O” (*big-O notation*).

Se o custo de um algoritmo é $O(C_s)$, isso significa que o algoritmo pode ser convertido em sistema serial com tempo de execução igual a $O(C_s)$. Caso o algoritmo paralelo leve o mesmo tempo ($O(C_s)$) para resolver o mesmo problema em paralelo, ele é dito ser custo-ótimo (*cost-optimal*), uma vez que a sua eficiência é igual a 1 ou $O(1)$ (Kumar et al., 1994).

5.3 Imagens-teste

5.3.1 Imagem-padrão

A imagem-padrão, ilustrada na Fig. 5.4, é uma imagem digital monocromática de dimensão 256x256 pixels, criada por Cidade (2000) e estruturada propositadamente para ser empregada nos testes de restauração e usada no cálculo de medidas de avaliação de resultados (avaliações qualitativas) de restaurações feitas em imagens modificadas da imagem-padrão.

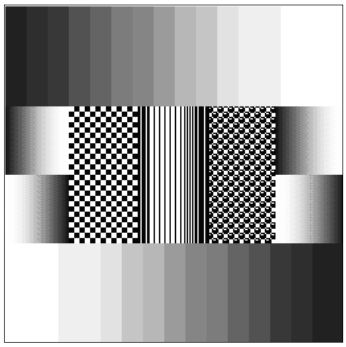


Figura 5.4 - Imagem-padrão (Cidade, 2000).

5.3.2 Simulando os efeitos degenerativos em uma imagem

Os efeitos de borramento nas imagens podem ser simulados, empregando-se uma função de espalhamento (PSF - *Point Spread Function*) dado pelo operador de borramento do tipo gaussiano com variância (σ^2) ajustável, apresentado na Eq. (30).

E, para introduzir o ruído aditivo η nessa imagem modificada (borrada), pode-se utilizar um gerador de números pseudo-randômicos (Gonzalez e Wintz, 1987; Cidade, 2000) que expressa uma relação sinal-ruído desejada em decibels (dB).

$$(SNR)_{dB} = 10.\log\left(\frac{\sum_{i=1}^{M}\sum_{j=1}^{M}\left[g(i,j)+\eta(i,j)\right]^2}{\sum_{i=1}^{M}\sum_{j=1}^{M}\left[k\overline{\eta}(i,j)\right]^2}\right), \tag{78}$$

onde g é a imagem borrada, (i, j) as coordenadas dos pontos dessa imagem e k é um fator de correção que determina a intensidade (positiva ou negativa) do ruído que será adicionado em cada ponto da imagem g, imposto sobre uma distribuição gaussiana normalizada $\overline{\eta}(i, j)$, para uma dada relação sinal-ruído $(SNR)_{dB}$ expressa em decibels.

Para se identificar as modificações impostas numa imagem (p.ex. imagem-padrão), adotou-se a mesma notação empregada por Furtado (2002), cuja sintaxe é dada por:

Dβvδsξ

onde β é a dimensão da matriz de borramento B, δ a variância e ξ é a relação sinal-ruído (SNR - *Signal Noise Ratio*) em decibels (dB).

Por exemplo, para um borramento da imagem-padrão empregando uma matriz B de dimensão 5x5, variância $\sigma^2 = 20$ e ruídos aditivos de 20 dB e 40 dB, as imagens modificadas serão identificadas, respectivamente, como D5v20s20 e D5v20s40, onde **D** representa a dimensão da matriz de borramento (associada à PSF), **v** é a variância e **s** a relação sinal-ruído em decibels (Fig. 5.5).

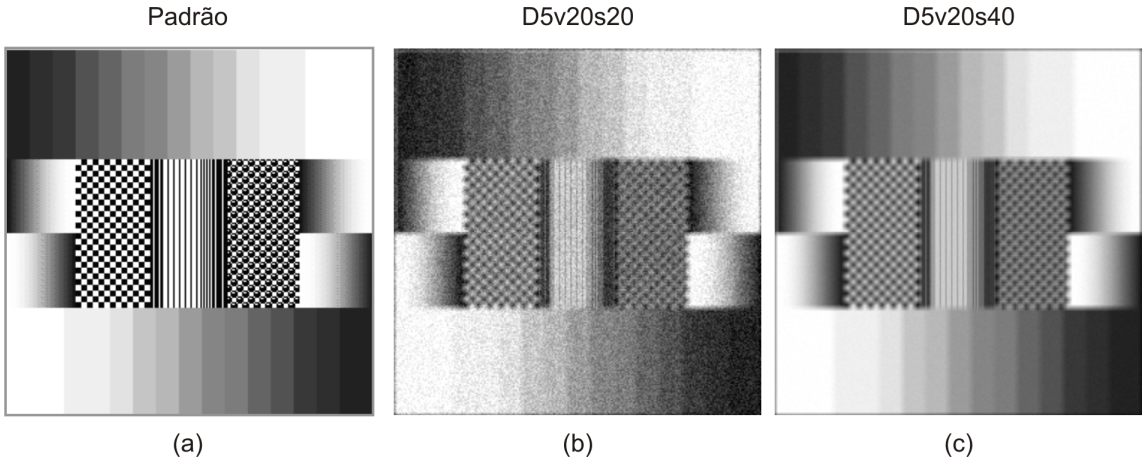


Figura 5.5 - Imagem-padrão (a) e imagens modificadas usando uma matriz de borramento B de dimensão 5x5, $\sigma^2 = 20$ e ruídos aditivos de 20 dB (b) e 40 dB (c).

6 IMPLEMENTAÇÕES ANTERIORES DO PROGRAMA DE RESTAURAÇÃO

6.1 Versões seriais do programa de restauração

A primeira versão serial do programa de restauração de imagens foi desenvolvida por Cidade (2000) e sofreu algumas modificações por Furtado (2002) em sua dissertação de mestrado. Apesar de eficaz, entretanto, a última versão do programa não apresentava um bom desempenho computacional, pois a sua codificação não empregava de maneira eficiente os recursos de programação disponibilizados pela linguagem C++/Builder®, usada para implementá-lo.

Para que o desempenho de programas paralelos seja avaliado é necessário que as métricas de avaliação sejam fiéis e que tenham como base de comparação um programa serial otimizado que resolva o mesmo problema mais rápido e de modo eficiente. Essa preocupação evita resultados irreais e suposições equivocadas, aumentando a credibilidade das conclusões apresentadas.

Para resolver isso, uma outra versão serial otimizada do programa de restauração foi desenvolvida por Stutz (2004) em sua dissertação de mestrado. Nos testes realizados em uma máquina com o processador AMD Athlon® 2600, com 516Mb de memória, rodando somente o sistema operacional Windows®, a nova versão serial do programa de restauração otimizado apresentou um desempenho computacional melhor que a codificação anterior da ordem de 70% aproximadamente.

Para que o programa de restauração pudesse ser testado no ambiente Linux®, o novo código foi implementado de modo que pudesse ser compilado e executado nessa plataforma. Além disso, outros recursos foram acrescentados à nova codificação serial do programa de restauração, dentre eles: (a) critérios de parada por alcance de tolerância; (b) emprego do método SOR na solução do sistema de equações lineares; (c) opção de aplicar ou não a realimentação; (d) restauração de apenas uma parte da imagem (janela de restauração); (e) emissão de relatórios detalhados sobre o andamento da execução do programa.

6.2 Primeira versão paralela do programa de restauração

Apesar do novo código serial ter sido implementado de forma otimizada, o esforço computacional exigido pelo método de restauração faz com que o programa de restauração serial apresente um baixo desempenho computacional, mesmo quando ele é executando em equipamentos dedicados e rápidos. Para solucionar esse problema, Stutz (2004) em seu trabalho de dissertação desenvolveu uma estratégia de implementação, empregando técnicas de computação paralela, para codificar a primeira versão paralela do programa de restauração e, como consequência disso, conseguiu reduzir os tempo de execução. Para efeito de entendimento e convenção, essa versão paralela do algoritmo de restauração será denominada aqui "**versão paralela anterior**" ou "**primeira versão paralela**".

- ***Decompondo o problema de restauração***

No desenvolvimento do código computacional em paralelo, foi necessário analisar o algoritmo de restauração de imagens e determinar qual dos paradigmas de programação ele mais se adequa (vide item 3.3).

Conforme ilustrado no fluxograma da figura 4.4, a característica serial do algoritmo de restauração, juntamente com o fato de um mesmo conjunto de operações ser aplicado em todos os pontos da imagem, faz com que a adoção da técnica de decomposição de domínio seja uma escolha natural para ser empregada na implementação do programa paralelo do problema de restauração.

- ***Método de restauração em paralelo: primeira versão paralela do programa de restauração.***

Resumidamente, a Fig. 6.1 ilustra como o processo de restauração é feito na primeira versão paralela do programa de restauração:

a) Primeiramente, uma imagem é dividida em *np* (número total de processadores) partições bloco-linha sobrepostas;
b) Cada uma dessas partições é enviada a uma unidade de processamento em particular;

c) Cada processador recebe uma partição e a processa, independente dos demais. De tempos em tempos, as unidades de processamento podem precisar se comunicar para fazer atualizações e verificações de convergência;

d) Ao final do processamento, as áreas de sobreposição são descartadas e as partições são reunidas novamente, compondo a imagem final restaurada.

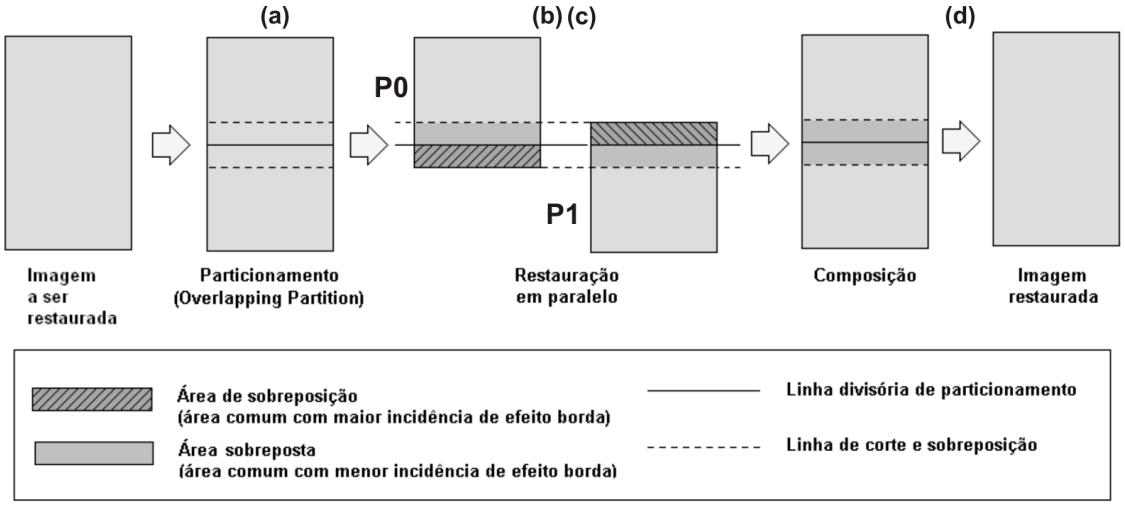


Figura 6.1 - Processo de restauração da primeira versão paralela do programa de restauração empregando dois processadores (P0 e P1) e aplicando *overlapping partition*.

Um aspecto importante desta técnica é a restauração de toda a partição, inclusive das áreas de sobreposição, ocasionando um duplo processamento de uma mesma região da imagem (áreas de sobreposição e sobreposta) feita em paralelo por duas partições vizinhas. A vantagem desta estratégia é tornar os processadores mais independentes, uns dos outros, e substituir os tempos de comunicação para troca de dados (mais lenta) por tempos de processamento (mais rápida). A desvantagem dessa abordagem é a sobrecarga (*overhead*) imposta às unidades de processamento e o surgimento de artefatos na imagem restaurada, cujo problema e solução são descritos um pouco mais adiante neste capítulo.

Melhores resultados (restaurações) podem ser obtidos com esta técnica, se em cada iteração as áreas de sobreposição forem atualizadas pela média aritmética simples, ponto a ponto, dessa área com a área sobreposta da partição vizinha (Fig. 6.2).

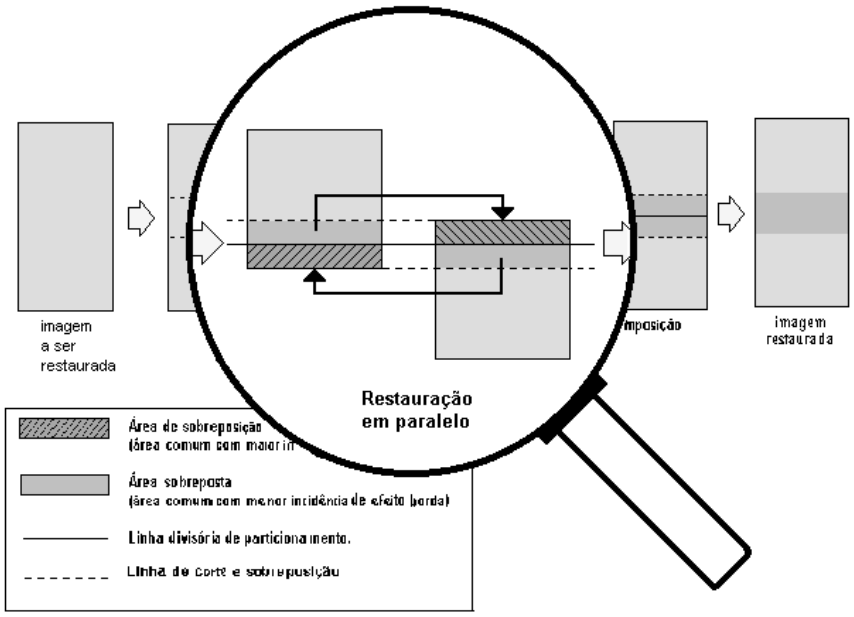


Figura 6.2 - Troca de dados entre as unidades de processamento, para atualização por média ponto a ponto das áreas de sobreposição.

Visto que a cada iteração as unidades de processamento precisam trocar informações para fazer checagens de convergência, o processo de atualização das áreas de sobreposição pode ser feito nesse momento. Um ponto negativo desse processo é que o tempo de processamento e a comunicação, gastos (custo operacional) com as atualizações e checagens, pode se tornar alto, prejudicando o desempenho do programa paralelo.

- ***O efeito-borda***

Durante o desenvolvimento do algoritmo da primeira versão paralela, diversas estratégias de implementação do programa de restauração, empregando a decomposição de domínio, foram codificadas e testadas. Nos testes realizados, verificou-se que a decomposição de domínio introduz artefatos na imagem restaurada que prejudicam a qualidade da restauração (Fig. 6.3). Esse efeito é maior, principalmente, quando a abordagem *striped partition* é empregada.

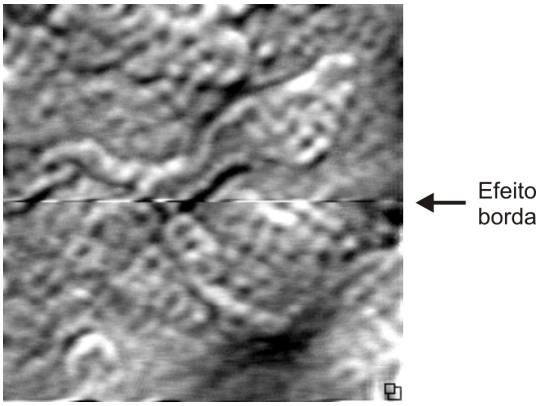


Figura 6.3 - Surgimento de artefatos (efeito-borda) em uma imagem restaurada com dois processadores, rodando a primeira versão paralela do programa de restauração empregando a técnica de decomposição de domínio sem sobreposição (*striped partition)*.

Esse fenômeno, aqui denominado "**efeito-borda**", decorre da conjunção de dois fatores: (a) o operador de borramento B que estabelece uma área de influência ao redor dos pontos em uma imagem (Fig. 4.3c e 4.6), e (b) a decomposição de domínio que cria bordas artificiais na imagem original ao particioná-la em blocos de dados menores. A associação desses fatores faz com que os pontos próximos das bordas artificiais em uma partição sofram influências de elementos da imagem que extrapolam essas bordas, que se encontram dentro das partições vizinhas (acima e abaixo) e que não estão disponíveis na própria partição (Fig. 6.4).

Como o tratamento dos pontos próximos das bordas (descrito no item 4.8), artificiais ou não, é diferente dos demais pontos internos da imagem, a restauração nessa região sofre os efeitos dessa solução (tratamento dos pontos próximos das bordas), que acabam produzindo os artefatos que são observados na imagem restaurada (Fig. 6.3).

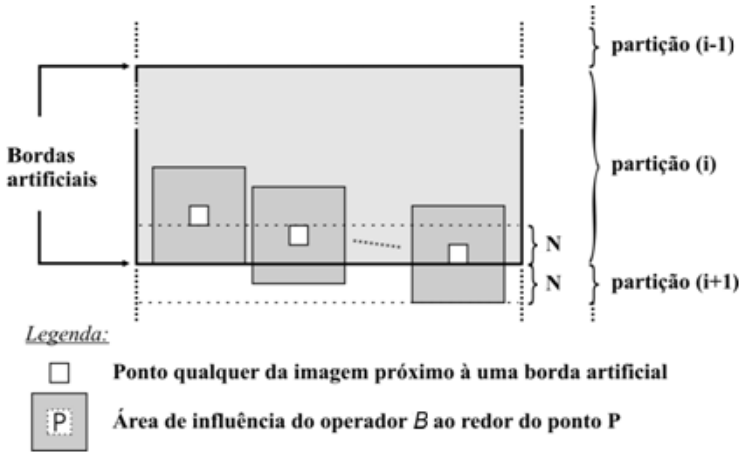


Figura 6.4 - Área de influência do operador B próxima a uma borda artificial.

- ***Solução adotada para resolver o problema do efeito-borda***

Para reduzir os artefatos observados na imagem restaurada produzidos pelo efeito-borda (Fig. 6.3), como solução, empregou-se a técnica de sobreposição (*overlapping partition)* – técnica em que uma imagem é dividida em vários blocos (ou partições) não-disjuntos e que se sobrepõem (Fig. 3.4). O uso desta técnica tem como objetivo estender as bordas artificiais para fora das linhas de particionamento – linhas que dividem uma imagem em um dado número de blocos de tamanhos aproximadamente iguais. Como o efeito-borda é mais intenso e ressaltado próximo das bordas, ao estender as bordas artificiais, estendem-se junto com elas, também, os efeitos-borda. Esta estratégia faz com que os artefatos observados na imagem restaurada apareçam com uma intensidade maior fora da área não-estendida da partição, ou seja, dentro da área de sobreposição (Fig. 6.5).

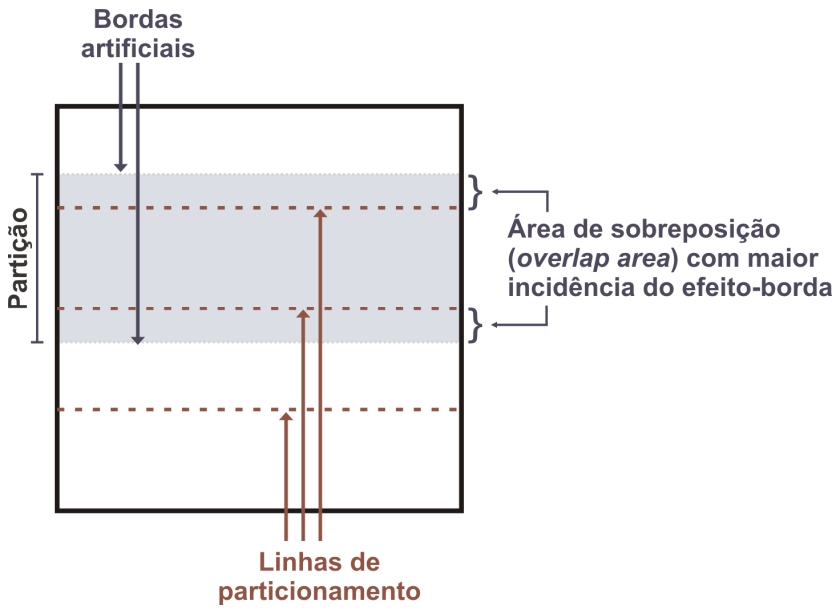


Figura 6.5 - Aplicação de sobreposição (*overlapping partition*) na restauração de imagens em paralelo.

Após o término do processo de restauração, as áreas de sobreposição são descartadas (área com maior incidência do efeito-borda) e o restante da partição (com menor incidência do efeito-borda) é reunido com as das demais partições vizinhas, formando a imagem final restaurada (Fig. 6.6).

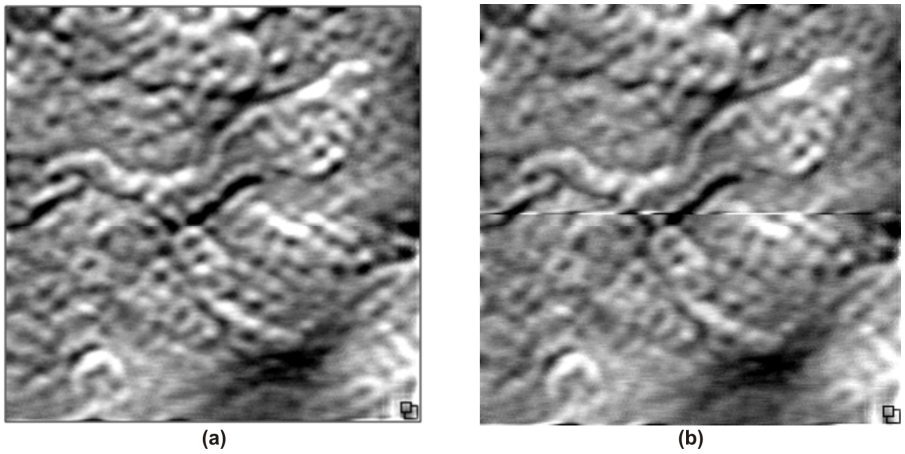


Figura 6.6 - Imagem restaurada pela primeira versão paralela do programa de restauração usando partições sobrepostas (*overlapping partition*), onde não se observa mais tão claramente o efeito-borda (a) que aparece quando a sobreposição não é empregada (b).

Essa estratégia faz com que a imagem restaurada em paralelo sofra menos com os efeitos-borda e o resultado final (restauração) fique bem próximo da imagem restaurada serialmente (Fig. 6.7).

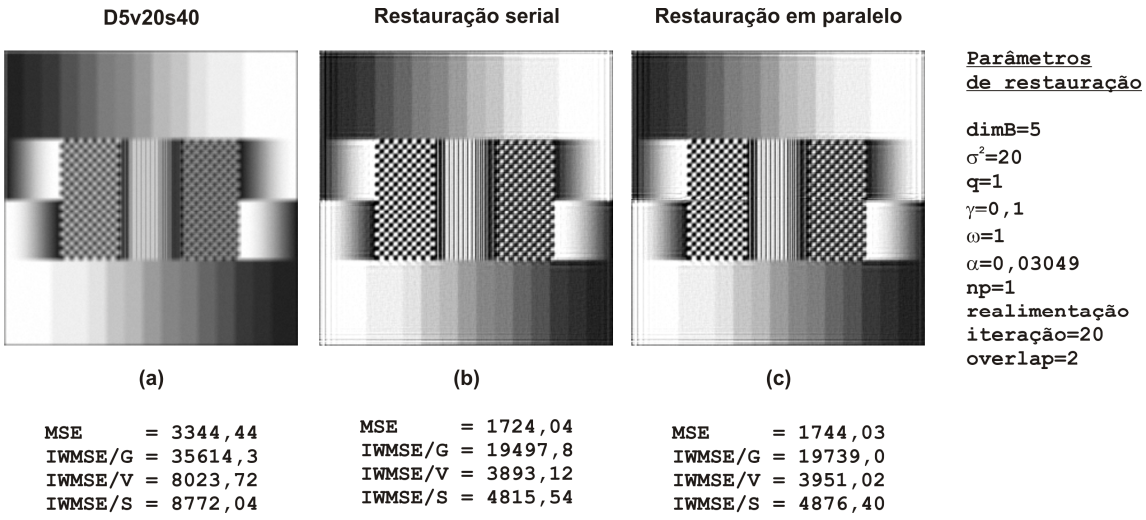
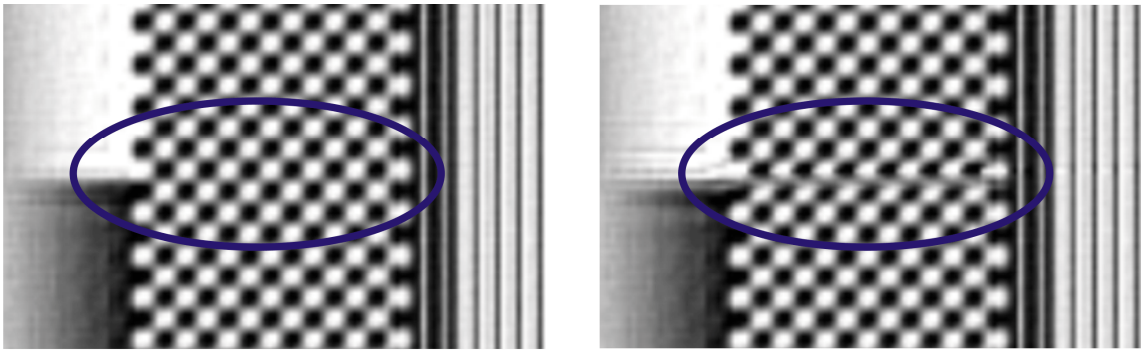


Figura 6.7 - Imagem-padrão modificada – D5v20s40 (a) – e resultados de restaurações feitas pelos programas: serial (b) e primeira versão paralela (c).

Essa pequena diferença entre os dois métodos, serial (MSE=1724,04) e paralelo (MSE=1744,03), ocorre por causa do efeito-borda que foi minimizado, mas não eliminado do processo de restauração em paralelo. Na Fig. 6.8, é mostrada em zoom uma comparação entre as duas restaurações da Fig. 6.7. No detalhe, observa-se uma diferença visual entre os resultados obtidos com os dois métodos, onde se percebe uma pequena perturbação na imagem restaurada em paralelo, resultado do efeito-borda.



(a) **(b)**

Figura 6.8 - Comparação visual entre as duas restaurações serial (a) e primeira versão paralela (b). No detalhe, a diferença entre os resultados obtidos com os dois métodos.

Dado que o tamanho do operador B é conhecido *a priori* ($\dim B = 2N+1$) e que a área de influência se estende N linhas por sobre a partição vizinha, o tamanho da área de sobreposição deve ser de no mínimo N linhas (*overlap area* $\geq N$). Quanto maior for a área de sobreposição (*overlap area*) utilizada, menores são os efeitos (artefatos) observados na imagem final restaurada em paralelo. Por outro lado, porém, as sobrecargas de trabalho (*overheads*) serão maiores em cada processador.

- ***Janelas de verificação***

Uma forma de se minimizar o impacto das comunicações no tempo de execução em paralelo seria estabelecer momentos de iteração apropriados para se fazer as atualizações e checagens de convergência entre os processadores (janelas de verificação), de modo a reduzir o custo operacional. Fora das janelas, as unidades de processamento não se comunicam.

Consequentemente, o custo operacional reduz e o desempenho aumenta. Esse processo, entretanto, não é sem perdas. A partir do momento em que as unidades de processamento não atualizam as suas áreas de sobreposição, os efeitos-borda vão aumentando e seguem em direção ao centro da partição, contaminando o resultado final (restauração). Assim, as janelas de verificação não devem ser muito grandes (>3) para não prejudicar o resultado e nem tão pequenas (=1) de modo que prejudiquem o desempenho do programa paralelo (Stutz, 2004).

- ***Estratégia de distribuição de cargas para minimização de desbalanceamentos***

Em processamento paralelo, de modo geral, procura-se empregar um conjunto de unidades de processamento mais ou menos homogêneas em termos de capacidade e processador. Nesses casos, quando a decomposição de domínio é aplicada, procura-se distribuir um mesmo volume de dados para cada uma das unidades de processamento.

Em algumas situações, entretanto, diferentes volumes de dados podem e/ou devem ser empregados para compensar e minimizar os efeitos do desbalanceamento de cargas entre os processadores. As situações mais comuns são:

a) O uso de unidades de processamento não homogêneas (heterogêneas); ou
b) Quando o volume de dados distribuído às unidades homogêneas não for divisível pelo número de processadores em paralelo (np) etc.

Essa última situação é bastante comum, principalmente, quando a eficiência de um programa em paralelo está sendo estudada ao se variar o número de processadores. Para esse caso, uma solução que pode ser aplicada para reduzir os efeitos causados pelo desbalanceamento é fazendo-se uma melhor distribuição de cargas entre as unidades de processamento de tal forma que se um volume v_d de dados, para ser distribuído a np de unidades de processamento, resulte em uma divisão inteira m com resto $r > 0$,

$$v_d = m * np + r, \quad 0 \leq r < np, \tag{79}$$

então, uma boa distribuição de cargas seria fazer com que as primeiras *(np-r)* unidades de processamento recebam um volume m de dados e as r unidades restantes $(m+1)$ (Fig. 6.9).

Dessa forma, os tempos de espera (*idle-time*) gerados com o desbalanceamento de carga seriam minimizados, visto que eles poderiam ser muitos piores se, por exemplo, apenas uma unidade de processamento recebesse todo o volume $(m + r)$ de dados.

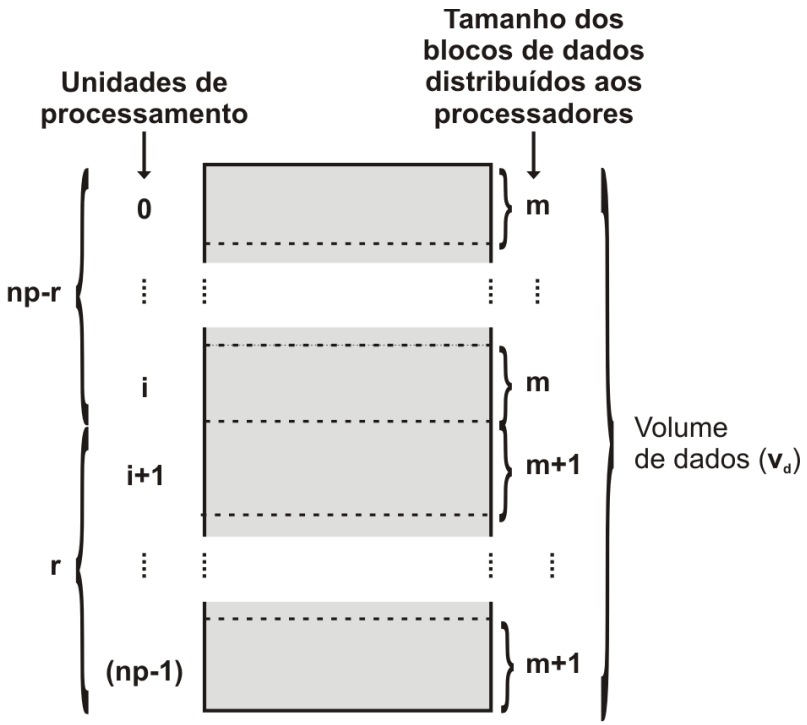


Figura 6.9 - Uma estratégia de distribuição de cargas para a minimização de desbalanceamento quando o volume de dados a ser processador não for divisível pelo número de processadores rodando em paralelo.

- ***Fluxograma da primeira versão do programa paralelo de restauração.***

Na Fig. 6.10 é mostrado, de forma simplificada, o fluxograma da primeira versão paralela do algoritmo de restauração.

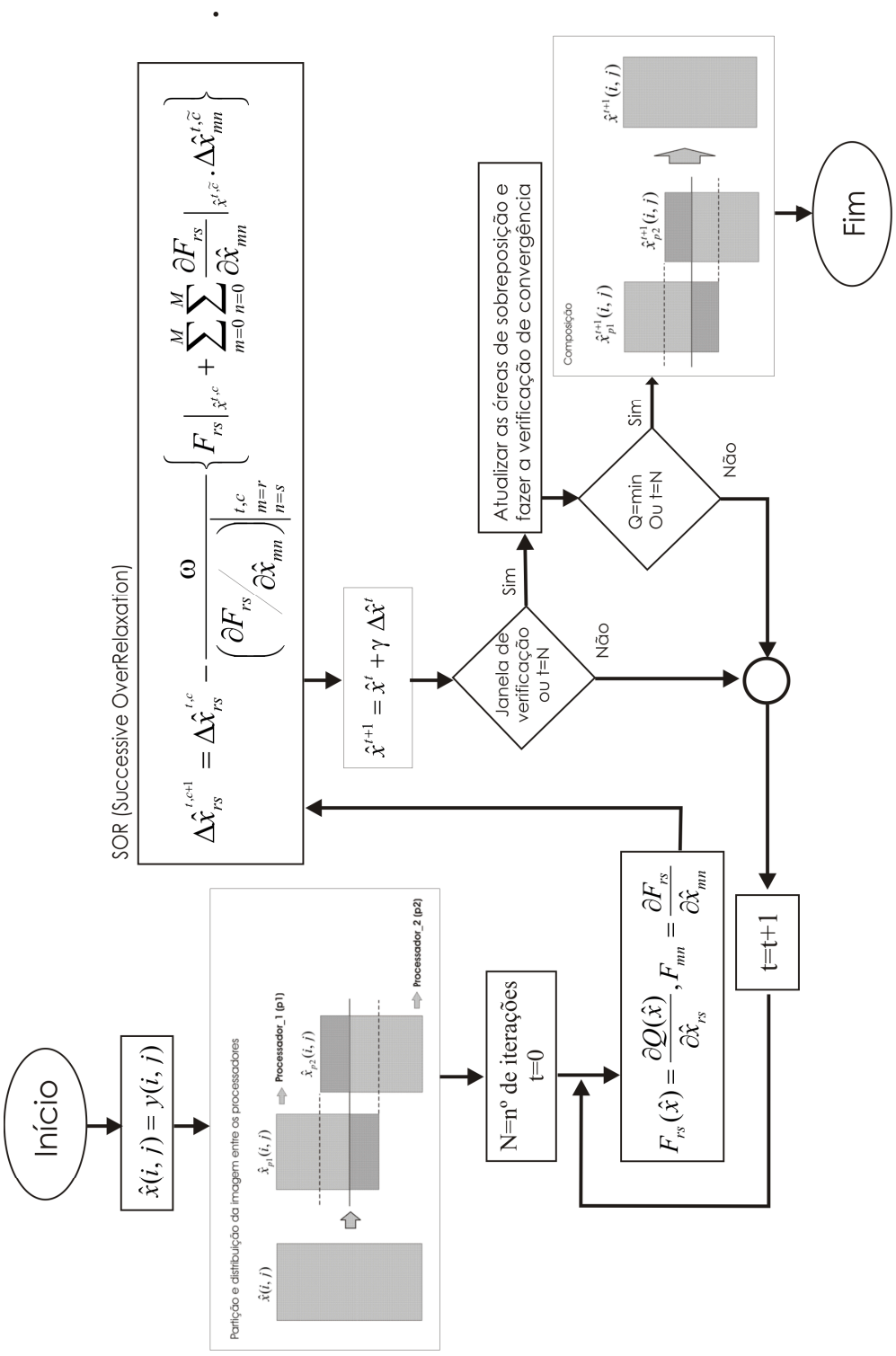


Figura 6.10 - Fluxograma simplificado da primeira versão do algoritmo paralelo de restauração de imagens (Stutz, 2004).

6.3 Problemas de sobrecargas da primeira versão paralela

- ***Arquitetura empregada***

Para a execução do programa serial e da primeira versão paralela, foi utilizado um cluster de computadores do Instituto Politécnico, Campus Regional da UERJ em Nova Friburgo/RJ, conhecido como Estação Cosmos, constituído por 16 nós interligados por um Switch Intel 10/100, formando um cluster de máquinas para processamento em paralelo, montado sobre o sistema operacional Slackware 9.0, rodando MPI/ LAM versão 7.0 com suporte às versões 1 e 2. Cada nó do cluster é composto por um microcomputador com processador AMD K6II de 450MHz, 128Mb de memória, disco IDE com capacidade de 15Gb e uma placa de rede 3Com 3C905-B PCI 10/100. Atualmente, a Estação Cosmos encontra-se desativada.

- ***Sobrecargas da primeira versão paralela***

Apesar da primeira versão paralela do programa de restauração ter alcançado um tempo computacional menor do que aqueles obtidos com o programa serial (Fig. 6.11a) e mesmo havendo um aumento na velocidade de processamento (Fig. 6.11b), detectou-se uma queda significativa na eficiência computacional do programa paralelo. Essa eficiência diminuía à medida que o número de processadores aumentava (Fig. 6.11c). Apesar da aceleração (*speedup*) aumentar a cada processador adicionado ao ambiente de execução em paralelo, o ganho relativo decaía de forma considerável (Fig. 6.11c) e os custos aumentavam quase que linearmente em uma curva bem ascendente (Fig. 6.11d).

Normalmente, após a inclusão de novas unidades de processamento, dependendo do problema, esse tipo de comportamento é esperado acontecer. O simples fato de termos quantidades maiores de processadores e quantidades menores de dados, por si só justificaria em parte essa queda no desempenho. Entretanto, a pequena quantidade de processadores envolvidos, a alta taxa de queda na eficiência e o custo muito ascendente chamaram a atenção para um novo problema nessa abordagem.

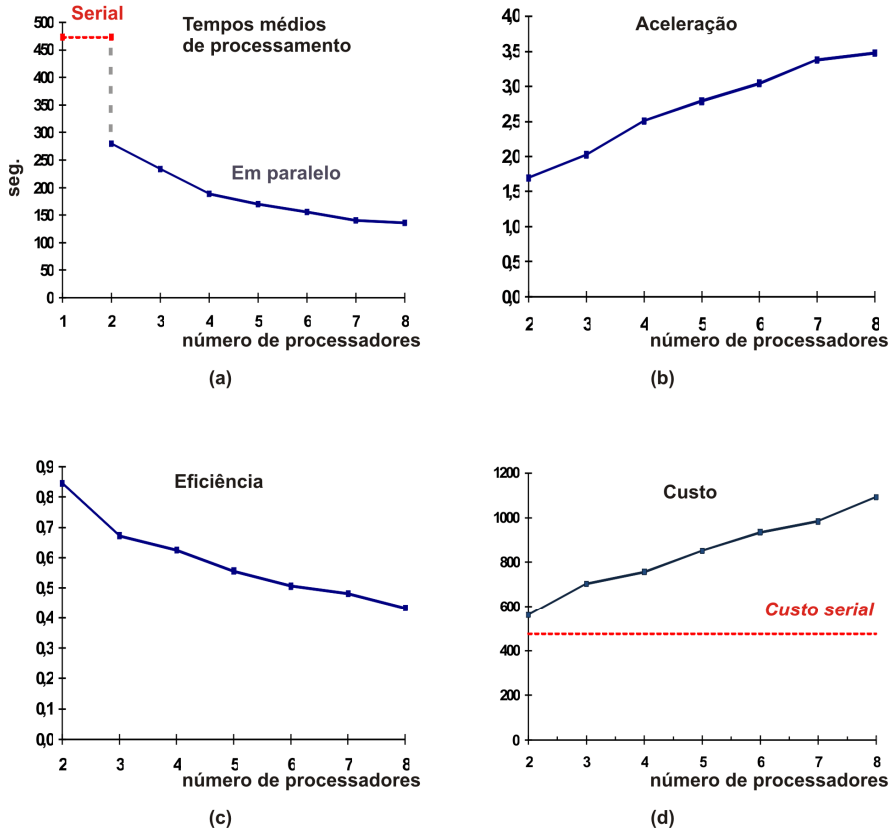


Figura 6.11 - Gráfico de desempenho do programa paralelo de restauração no processamento de uma imagem de tamanho 256x256, usando os seguintes parâmetros: $\dim B = 21$, área de sobreposição = 21, número máximo de iterações do método de Gauss-Seidel = 2 e do processo de restauração = 20, realimentação, $\sigma^2 = 20$, $\alpha = 0{,}05$, $q = 1$, $\gamma = 0{,}1$ e $\omega = 1$.

Investigando-se melhor as causas desse problema, verificou-se que, apesar dos blocos de dados distribuídos serem cada vez menores, à medida que a quantidade de processadores aumenta, o tamanho das partições aumenta significativamente e de forma desproporcional. Esse aumento se dá em razão do tamanho das áreas de sobreposição, que permanece fixo apesar do número de processadores aumentar.

Conforme ilustrado na Fig. 6.12, para a restauração de uma imagem de 256x256 pixels, por exemplo, usando uma área de sobreposição igual a 21 e oito processadores, tem-se um aumento na carga total de trabalho de mais de 114% em relação à mesma restauração empregando apenas um processador. Com exceção do primeiro e último processadores que têm uma sobrecarga (*overhead*) de 65,6%, as demais unidades de processamento (intermediárias) têm um aumento de 131,3% mais no volume de em suas partições.

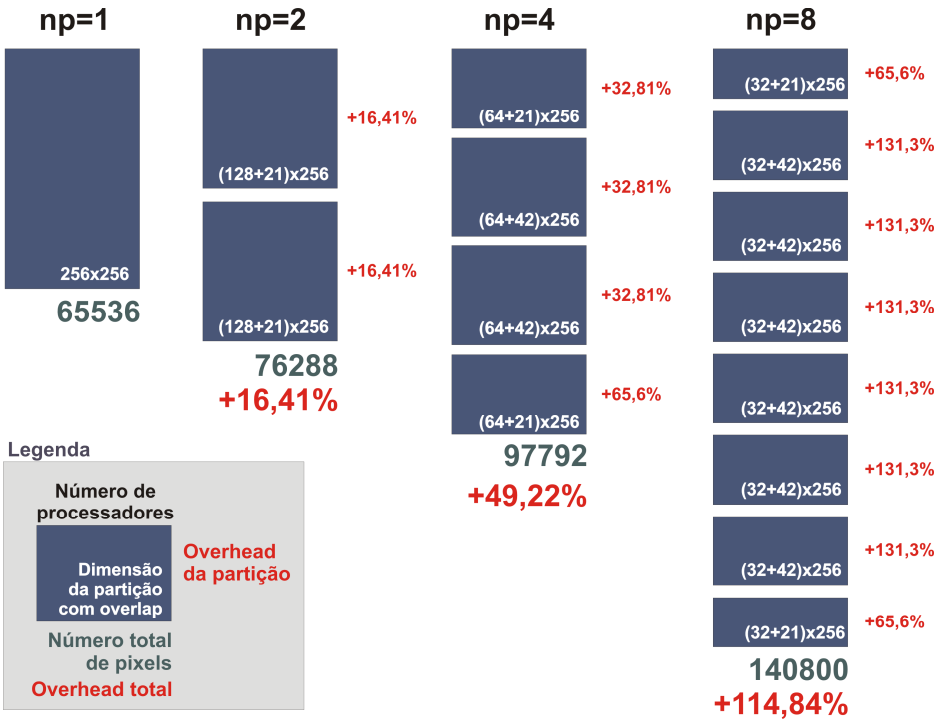


Figura 6.12 - Aumento no volume dos dados processados (*overhead*) que ocorre na primeira versão paralela à medida que se aumenta o número de processadores na restauração de uma imagem de dimensão 256x256, com uma área de sobreposição igual a 21.

Esse aumento na quantidade de dados a ser processada é significativo e resulta numa sobrecarga de trabalho adicional em cada processador, já que cada um deles processa uma mesma área de imagem (duplo processamento), sobrecarregando o processo de restauração em paralelo como um todo e reduzindo de forma indesejável a eficiência computacional dessa estratégia. Quanto maior o número de processadores envolvidos e quanto maior a área de sobreposição, maiores são as sobrecargas observadas nessa estratégia (Stutz, 2004).

7 NOVA ESTRATÉGIA DE IMPLEMENTAÇÃO PARALELA DO ALGORITMO DE RESTAURAÇÃO

7.1 Terminologias e convenções

Para uma melhor compreensão da nova estratégia de implementação do programa paralelo de restauração, é importante que sejam definidos alguns conceitos e termos que serão empregados neste capítulo.

Partição – resultado da divisão de uma matriz de dados (p.ex. uma imagem) em dois ou mais blocos-linha sobrepostos (*overlapping partition*) de tamanhos aproximadamente iguais, distribuídos às unidades de processamento diferentes para serem restauradas em paralelo (Fig. 7.1).

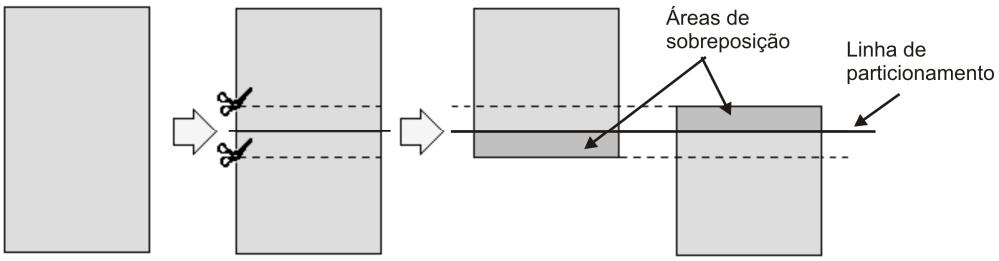


Figura 7.1 - Exemplo da divisão de um bloco de dados em duas partições blocos-linha sobrepostas.

Linhas de particionamento – são linhas que dividem uma matriz de dados em um certo número de blocos-linha não sobrepostos de tamanhos aproximadamente iguais (Figs. 6.5, 7.1 e 7.2).

Áreas de uma partição – conforme ilustrado na Fig. 7.2, em uma partição podem ser identificadas algumas áreas específicas. Denominaremos “Área-S”, a área da partição que se sobrepõe às partições vizinhas (área de sobreposição), e “Área-A”, o restante dela. A área-S subdivide-se em duas partes: S0 e S1. A área-S0 sobrepõe a partição vizinha acima e S1 abaixo. A área-A, por sua vez, subdivide-se em três partes: A0, A1 e A2. A área-A1 é a parte da área-A que não é sobreposta por nenhuma partição, enquanto que A0 e A2 são, respectivamente, as áreas sobrepostas pelas partições

vizinhas acima e abaixo. Visto que a primeira e a última partições não possuem, respectivamente, vizinhos acima e abaixo, cabe aqui ressaltar que a primeira partição não tem as áreas S0 e A0 e a última partição não possui as áreas A2 e S1.

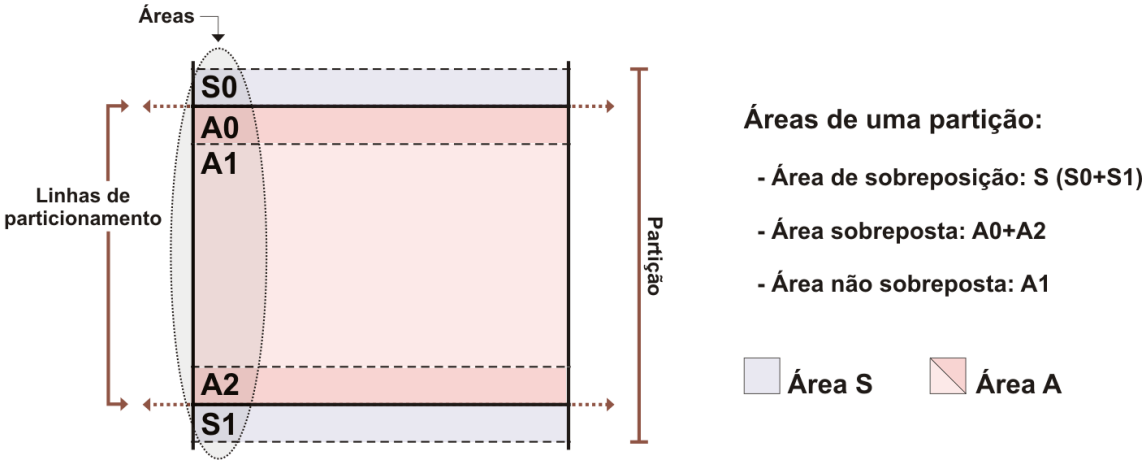


Figura 7.2 - Áreas de uma partição.

Classes de partições – No item 4.6, verificou-se que, para a solução do problema de minimização, além da imagem degradada outras matrizes de dados são necessárias na aplicação da técnica de restauração. Ao particionarmos a imagem, estaremos particionando de algum modo essas outras matrizes de dados. Para que se possa identificar essas matrizes, por um abuso da linguagem, serão definidos aqui alguns termos para as partições de dados com base em seu conteúdo. Portanto, denomina-se:

Y – a partição da imagem experimental (y) a ser restaurada.

X – a partição da imagem estimada ($\hat{x}$), obtida pelo processo iterativo de restauração.

BX – a partição da convolução $B * \hat{x}$.

$Y - BX$ – a partição da diferença entre as partições Y e BX.

Frs – a partição da derivada primeira Frs.

Fmn – a partição da derivada segunda Fmn.

ΔX – a partição com as correções $\Delta\hat{x}$.

Segmento – consiste em uma etapa do processo em paralelo que é responsável pela restauração de uma partição da imagem. Em alguns casos, o segmento pode ser usado também para designar o par: unidade de processamento+partição. Dentro de cada um dos segmentos encontramos os outros tipos de partições citados anteriormente.

7.2 Sequência de cálculos intrínseca ao processo de restauração

A técnica de restauração baseada na minimização do Funcional de Regularização de Tikhonov, apresentada no item 4.6 e ilustrada no fluxograma da Fig. 4.4, estabelece uma sequência de execução de cálculo de alguns termos que são usados no processo de restauração.

Conforme ilustrado na Fig. 7.3, observa-se que, para calcular as novas estimativas $\vec{\hat{x}}^{t+1}$, Eq. (42), é preciso que se calculem antes as correções $\Delta\vec{\hat{x}}$, Eqs. (45) ou (46). Contudo, para se obter essas correções, é preciso que se calculem antes as derivadas primeira F_{rs} e segunda F_{mn}, Eqs. (47) e (48) e essas, por sua vez, para serem obtidas demandam, respectivamente, o cálculo da diferença $Y - BX$ e da convolução $b_{kl}b_{m-r-k,n-s-l}$.

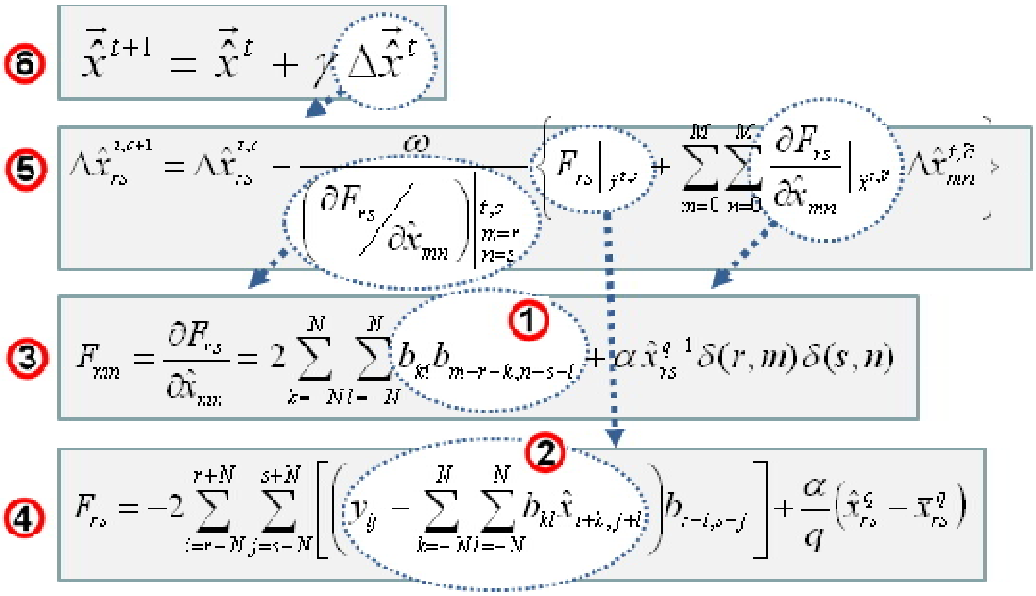


Figura 7.3 - Sequência de execução de cálculo dos termos usados no processo de restauração de imagens baseado na minimização do Funcional de Regularização de Tikhonov.

7.3 Estratégia de execução da sequência de cálculos em paralelo da nova estratégia paralela do algoritmo de restauração

7.3.1 Cálculo em paralelo da convolução $b_{kl}b_{m-r-k,n-s-l}$ ①

Analisando-se a convolução $b_{kl}b_{m-r-k,n-s-l}$, Eq. (48), percebe-se que esse termo não varia ao longo da execução do programa de restauração. Como o termo é dependente apenas da matriz de borramento B, cuja forma e tamanho são definidos *a priori*, o seu cálculo pode ser feito uma única vez de modo independente em cada uma das unidades de processamento e armazenado em uma matriz de dados temporária para ser usada toda vez que for preciso calcular a derivada de segunda ordem F_{mn}.

7.3.2 Cálculo em paralelo das diferenças $y - B\hat{x}$ ②

Dado que o operador de borramento B cria uma área de influência em torno de um ponto qualquer da imagem (Fig. 7.4), nota-se que a partir de um certo limite conforme esse ponto vai se aproximando da linha de particionamento, os dados necessários ao cálculo da convolução BX vão extrapolando a partição vizinha.

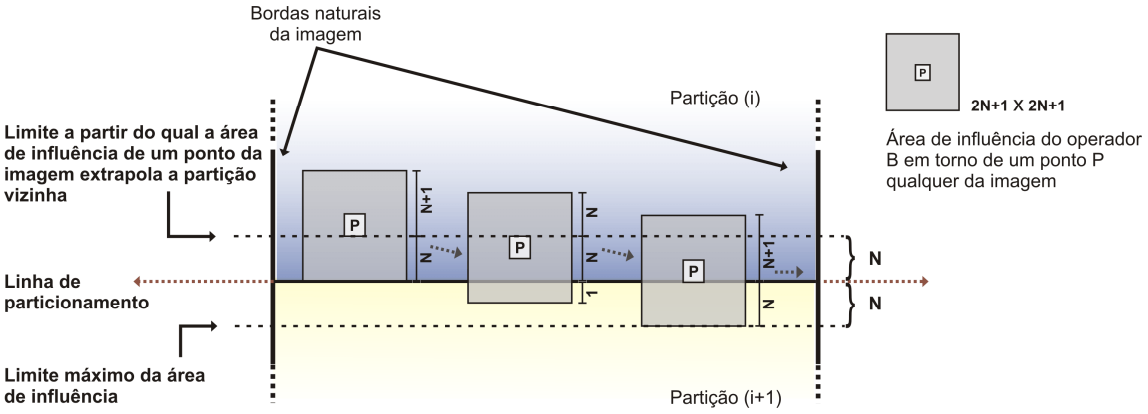


Figura 7.4 - Visualização da área de influência em torno do ponto P conforme esse ponto se aproxima da linha de particionamento.

Esse limite é dependente da dimensão do operador B e como essa dimensão é definida por um valor arbitrário N ($\dim B = 2N+1$), é fácil concluir que, para se calcular a convolução BX nos pontos próximos da linha de particionamento, é preciso que a partição X seja estendida (sobreposição) N linhas por sobre as partições vizinhas acima e abaixo (Fig. 7.5). Esse alargamento da partição X permite que cada um dos processadores calcule, de forma paralela e independente, a convolução BX e a diferença $Y - BX$ até o limite da linha de particionamento, ou seja, dentro da área-A (Fig. 7.6).

Conforme ilustrado nas Figs. 7.5 e 7.6, o processo de cálculo das diferenças $Y - BX$ pode ser descrito da seguinte forma:

(1) Antes de iniciar o processo de cálculo/restauração, propriamente dito,
   (a) primeiramente, faz-se o particionamento da imagem a ser restaurada em np (número total de processadores) partes com uma sobreposição de N linhas.
   (b) Logo a seguir, distribuem-se essas partições resultantes (Y) a cada uma das unidades de processamento (Pj, $j = 0,1,...,(np-1)$). Esse processo de carga é executado apenas uma vez, no início da execução do programa.
   (c) Como estimativa inicial, o valor da partição X^0 (valor da partição X no instante $t = 0$ de restauração) pode ser uma cópia da partição Y (Fig. 7.5). Quanto aos demais valores de X^t ($t = 1,2,...$), eles serão obtidos posteriormente, resultantes do processo iterativo de restauração;

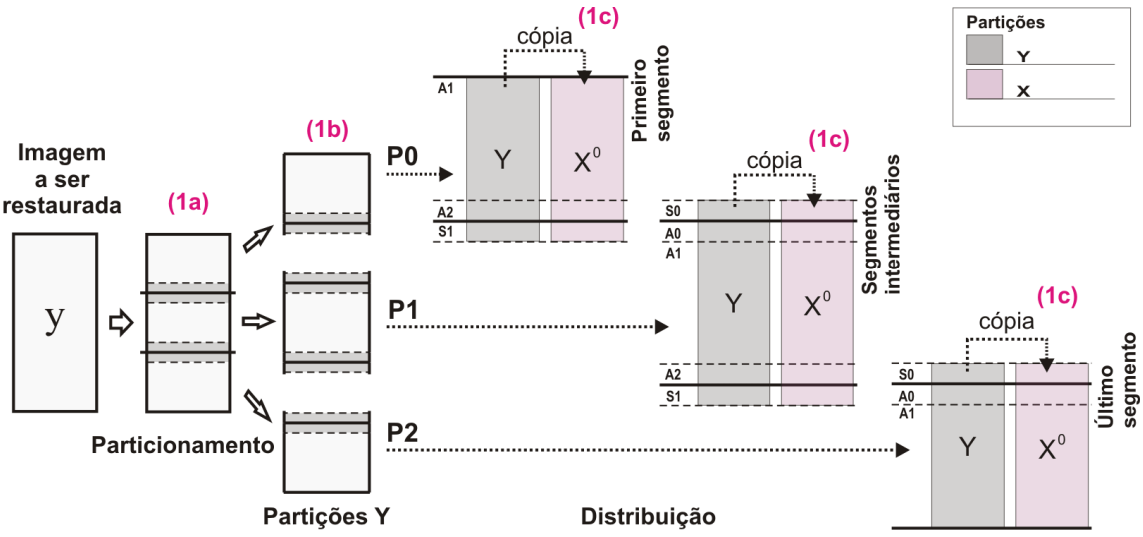


Figura 7.5 - Inicialização das matrizes de dados nas unidades de processamento feito antes do inicio do processo de restauração em paralelo.

(2) Uma vez tendo sido inicializado as partições X e Y, a unidade de processamento começa o processo paralelo de restauração calculando de forma independente a área-A dos termos

(a) BX e, em seguida,

(b) $Y - BX$, faltando apenas o cálculo desse último termo nas áreas de sobreposição (S0 e S1) (Fig. 7.6);

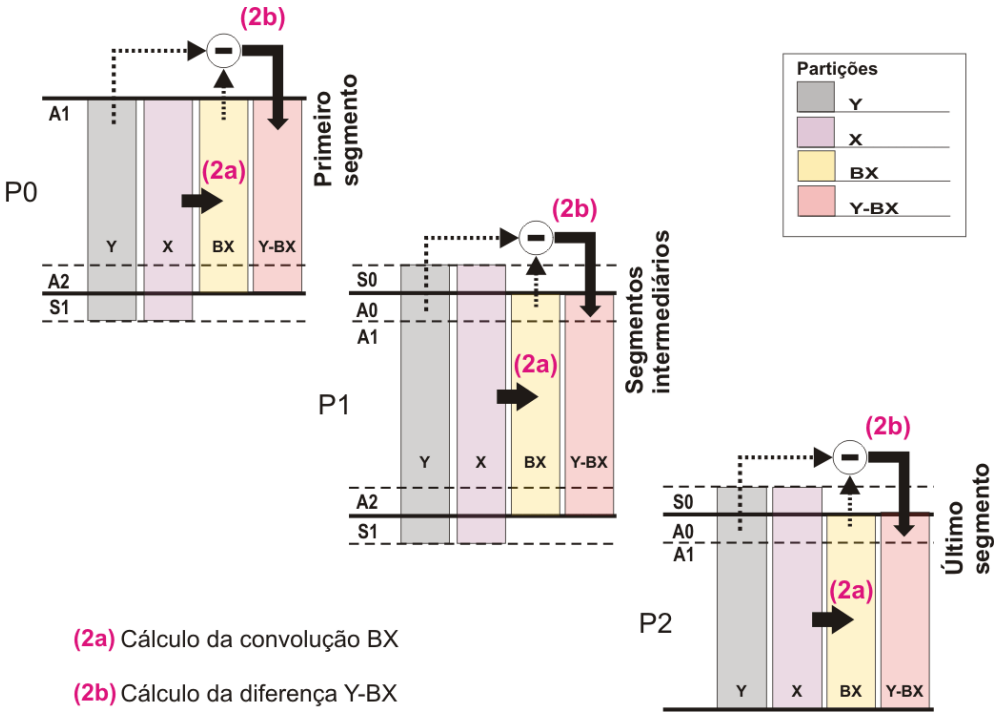


Figura 7.6 - Cálculo em paralelo dos termos BX e $Y - BX$.

A fim de evitar recálculos e novas sobreposições, as diferenças $Y - BX$ da área de sobreposição (área-S) de cada partição, que ficaram faltando no passo (2), podem ser obtidas através da troca de dados entre as unidades de processamento vizinhas. Uma vez que as partes que faltam em uma partição podem ser obtidas das partições acima e abaixo. Detalhadamente, o processo de troca desses dados entre os processadores pode ser descrito da seguinte maneira:

(3) As áreas A0 e A2 da partição $Y - BX$ de cada unidade de processamento são enviadas para os processadores vizinhos acima e abaixo, de forma a completarem as áreas S1 e S0, respectivamente. O processo de envio de dados pode ser feito de

modo assíncrono (*asynchronous send*), aproveitando a área de armazenamento de dados temporário (*buffer*) do canal de comunicação (Fig. 7.7). Isso faz com que a operação *send* retorne (devolva a execução e o controle aos processadores) antes que os dados tenham sido efetivamente entregues às unidades de processamento vizinhas, possibilitando que cada processador continue o seu processamento, aproveitando o tempo que ficaria aguardando por uma comunicação (*receiver*), para calcular, por exemplo, as derivadas de segunda ordem e, com isso, reduzindo o tempo de latência de comunicação;

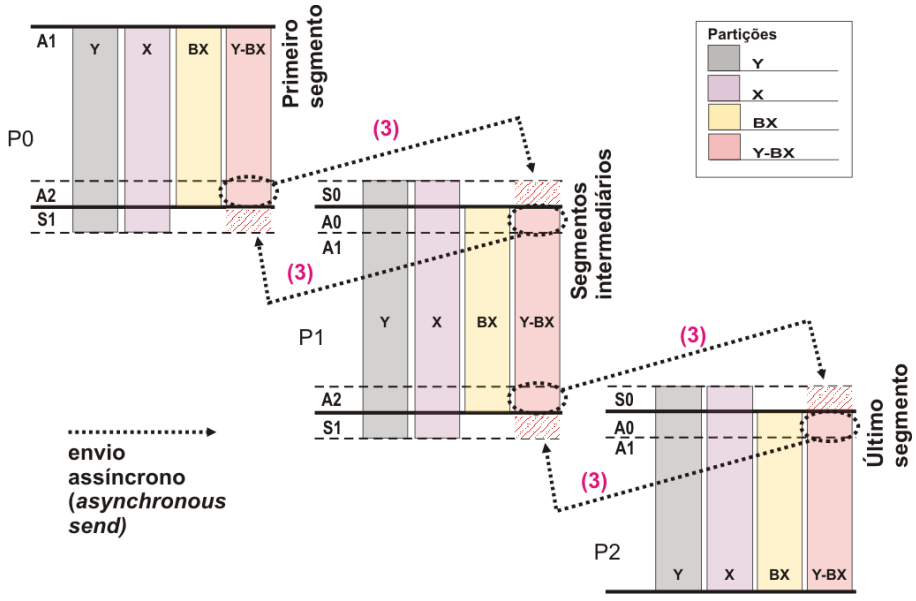


Figura 7.7 - Atualização das áreas S0 e S1 necessária ao cálculo da derivada primeira F_{rs} .

É a partir deste ponto que as duas estratégias paralelas (nova e anterior) diferem. Na estratégia paralela anterior, a imagem é processada da área-S0 até a área-S1, aplicando-se a técnica de corte nos pontos próximos aos limites da linha de particionamento, o que acaba resultando no efeito-borda observado nas restaurações feitas com a primeira versão paralela do programa de restauração (Fig. 6.3 e 6.8). Enquanto que na nova estratégia, a restauração é feita apenas na área-A e as áreas S0 e S1 restauradas são obtidas através da troca de dados entre as unidades de processamento vizinhas acima e abaixo, respectivamente. Essa modificação na nova estratégia faz com que esses pontos não sejam mais tratados de forma diferente, tal como na versão paralela anterior, mas sim tratados de forma igual aos outros pontos internos da imagem, conforme deveriam ser. Como resultado dessa mudança, o efeito-borda próximo das linhas de partição é completamente eliminado na nova estratégia.

7.3.3 *Cálculo em paralelo das derivadas de segunda ordem* F_{mn} ③

Fixando um ponto (r,s) qualquer dentro da área-A, o cálculo das derivadas de segunda ordem nesse ponto ($\left(\partial F_{rs} / \partial \hat{x}_{mn}\right)\Big|_{\substack{m=r \\ n=s}}$) não depende de nenhum dado de fora do segmento. Ao analisar a Eq. (48), verifica-se que o único termo que a equação precisa encontra-se no próprio segmento, precisamente na área-A da partição X.

Sendo um termo independente das demais partições (exceto de X), o cálculo paralelo das derivadas de segunda ordem pode ser executado tão logo tenham sido enviadas as diferenças $Y - BX$ das áreas A0 e A2, descrito em (3). Assim, ao invés de ficar esperando pelos dados das unidades de processamento vizinhas estarem disponíveis para recebimento, o processador pode, enquanto isso, aproveitar o tempo em que ficaria ocioso aguardando a comunicação (*idle-time*) para calcular as derivadas de segunda ordem (F_{mn}).

7.3.4 *Cálculo em paralelo das derivadas de primeira ordem* *Frs* ④

Detalhadamente, o processo de cálculo das derivadas de primeira ordem *Frs*, Eq. (47), pode ser descrito da seguinte maneira:

(4) Após o processo de cálculo das derivadas de segunda ordem,

(a) cada processador aguarda, de forma síncrona, pelo recebimento (*synchronous receive*) dos dados que foram enviados a ela pelas unidades de processamento vizinhas (passo 3).

(b) O cálculo da partição Frs, limitado apenas à área-A, é iniciado assim que a unidade de processamento receber esses dados (Fig. 7.8).

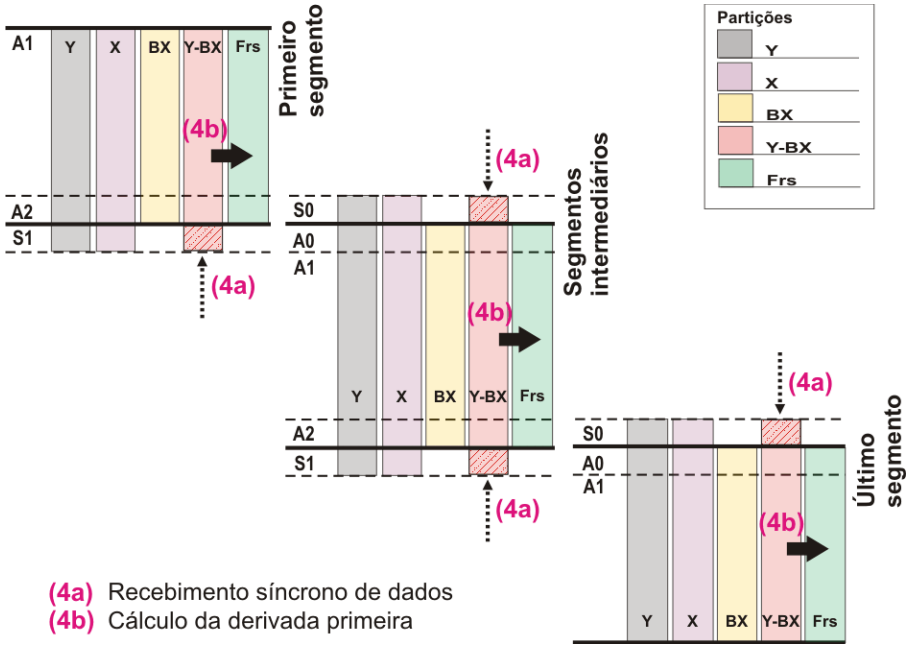


Figura 7.8 - Cálculo em paralelo do termo F_{rs}.

7.3.5 *Cálculo em paralelo das correções* $\Delta\vec{x}$ ⑤

O cálculo das correções $\Delta\vec{x}$ exige a solução de um sistema linear pelo método iterativo de Gauss-Seidel ou SOR, Eq. (45) e (46). Esses métodos estabelecem uma ordem em que os elementos da matriz de incógnitas ΔX são calculados. Essa ordem pode ser: (a) de cima para baixo (mais comum); (b) de baixo para cima, ou (c) em ambas as direções, em momentos alternados. Assumindo-se que a direção de cálculo seja de cima para baixo, estabelecemos também a sequência de inicialização dos processadores, bem como a ordem de restauração dos pontos da imagem nesse mesmo sentido. Ou seja, de cima para baixo.

Como efeito dessa decisão, as unidades de processamento com número menor de *rank* começam o processo de restauração primeiro, enquanto que as de número maior ficam aguardando os resultados da partição vizinha acima para iniciarem o seu processamento. De modo similar ao que acontece em um *pipeline*, essa espera acaba produzindo um efeito cascata no processo de cálculo das correções $\Delta\vec{x}$ que, por consequência, acaba introduzindo algum retardo no processamento em paralelo. Na literatura, essa espera por dados é conhecida como “conflito de dados”.

Um outro aspecto a ser observado é que, da mesma forma que o operador B afeta o cálculo da convolução BX ao criar uma área de influência em torno de um ponto da imagem e, consequentemente, a necessidade de se estender algumas partições, o cálculo dos elementos de ΔX também é influenciado por esse operador. Analisando-se as Eqs. (45), (46), (48) e (B.19), verifica-se que as áreas de sobreposição da partição ΔX são maiores (2N) que as demais partições (N) e que, a partir desse ponto, as derivadas de segunda ordem são nulas, Eq. (B.19). Portanto, para o cálculo das correções, precisamente aquelas que deverão ser obtidas na área de sobreposição (área-S), a partição ΔX deverá ter uma área de sobreposição duas vezes maior (2N) que as demais partições estendidas (N).

De modo a minimizar o conflito de dados, pode-se simplificar o processo de solução do sistema linear em paralelo, assumindo uma estimativa inicial igual a zero ($\Delta\vec{\hat{x}}^0 = 0$) e um número máximo de iterações igual a um para o método iterativo de Gauss-Seidel (ou SOR). Assim, ao admitir uma estimativa inicial igual a zero, elimina-se a necessidade de sobreposição da partição ΔX com a partição vizinha abaixo. E, ao estabelecer um número máximo de iterações igual a um, elimina-se a necessidade da partição de baixo transmitir os dados da área-A0 de ΔX para a partição acima. Desse modo, ao reduzirmos a necessidade de transmissão de dados entre as unidades de processamento, consequentemente, estamos reduzindo o conflito de dados e minimizando o seu impacto no cálculo das correções ΔX.

Conforme ilustrado na Fig. 7.9, o processo de cálculo das correções ΔX pode ser descrito da seguinte forma:

(5) inicia-se o processo com a primeira unidade de processamento calculando as correções ΔX na área-A, enquanto as outras unidades aguardam;
(6) Ao terminar o cálculo, a unidade transmite de modo assíncrono um bloco de $2N$ linhas da área-A2 de ΔX com as correções atualizadas para a unidade de processamento seguinte;
(7) A próxima unidade recebe os dados do processador anterior e,
(8) Logo a seguir, inicia o cálculo das correções ΔX dentro da área-A em sua partição;
(9) Os passos (6), (7) e (8) são repetidos pelas demais unidades de processamento, uma a uma, até que a última unidade de processamento termine o cálculo das correções em sua partição ΔX.

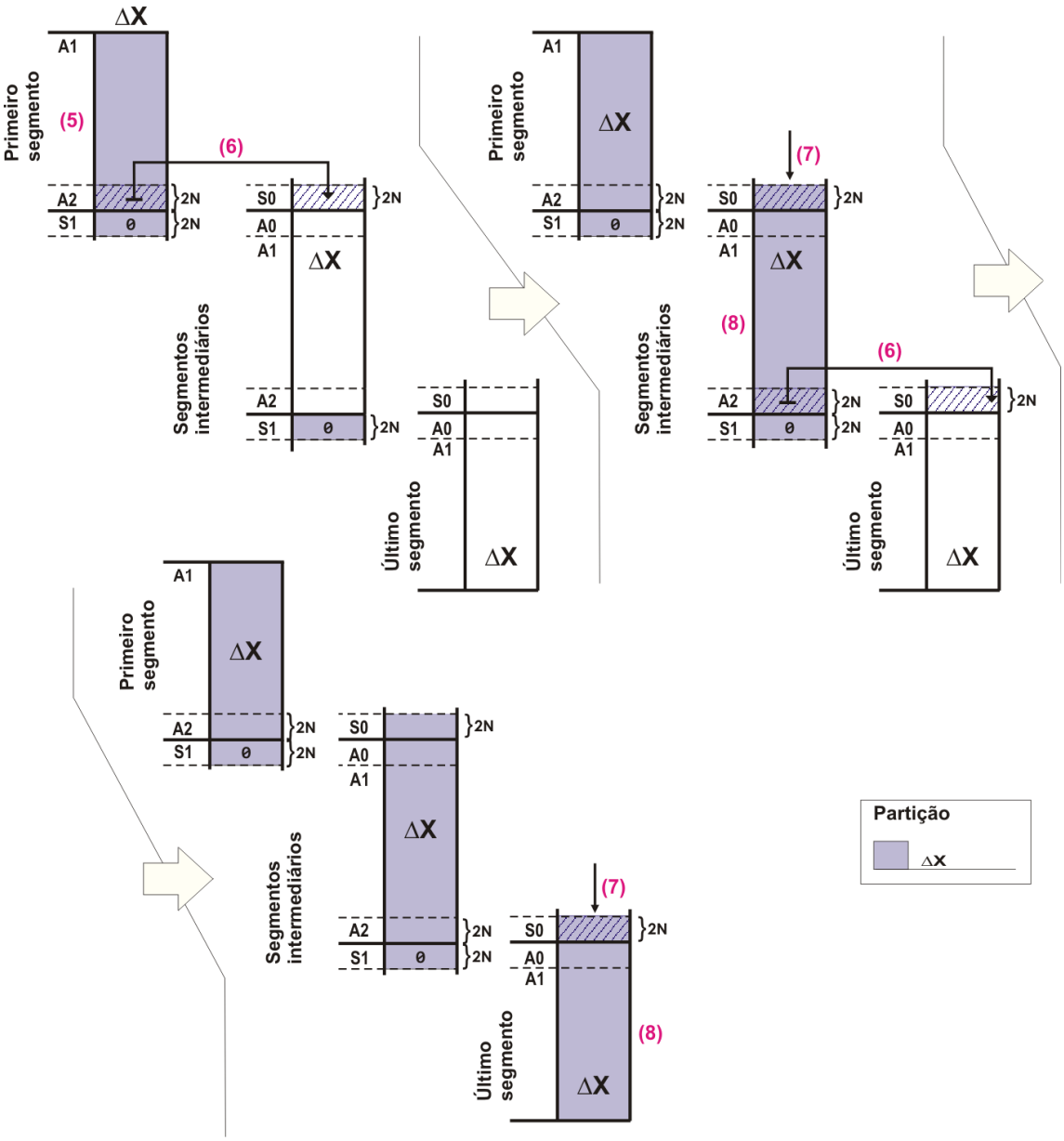


Figura 7.9 - Processo de cálculo das correções $\Delta\hat{\vec{x}}$ em paralelo.

7.3.6 Cálculo em paralelo das novas estimativas $\vec{\hat{x}}^{t+1}$ ⑥

Conforme ilustrado na Figs. 7.10 e 7.11, o processo de cálculo das novas estimativas pode ser descrito da seguinte forma:

(10) Uma vez obtidas as correções ΔX (passos 5 a 9) e tendo transmitido de forma assíncrona os dados para o processador vizinho abaixo (passo 6), a unidade de processamento pode, então, continuar a execução em paralelo, calculando as novas estimativas X^{t+1} ($t = 0,1,...,ni$) das áreas S0 e A (A0+A1+A2), enquanto que os demais processadores inferiores ainda continuam o processamento calculando as correções ΔX.

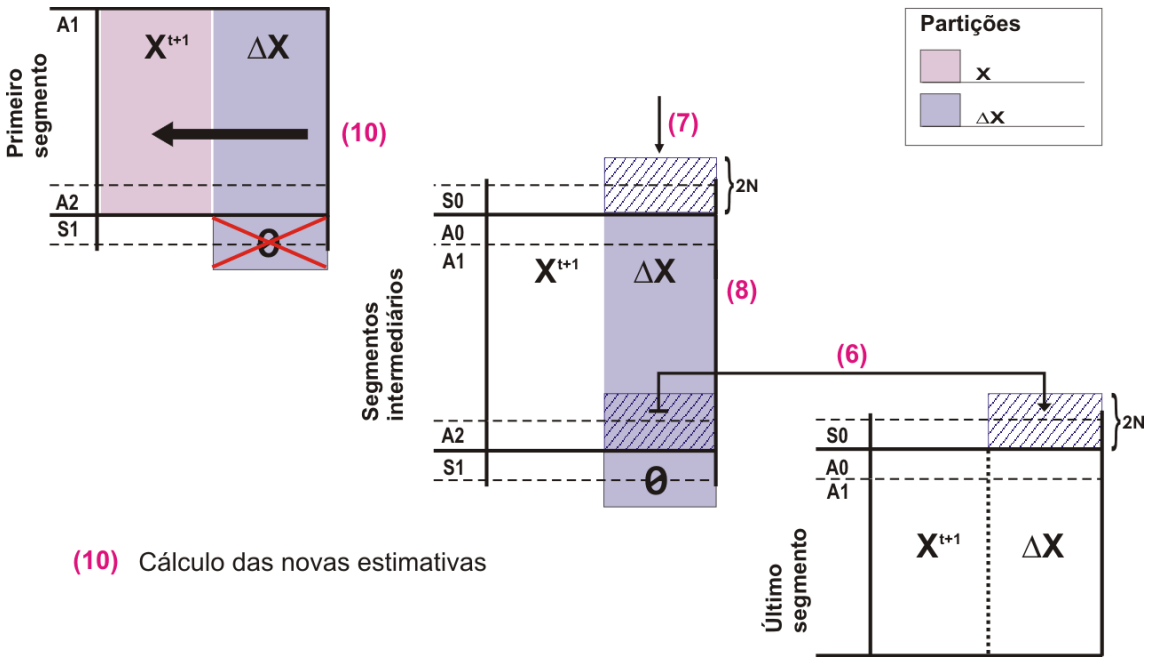


Figura 7.10 - Início do processo de cálculo das novas estimativas em paralelo.

(11) Assim que tiver calculado as novas estimativas X^{t+1},

(a) a unidade de processamento, então transmite a área-A0 ao processador acima,

(b) para que essa atualize os dados a área-S1.

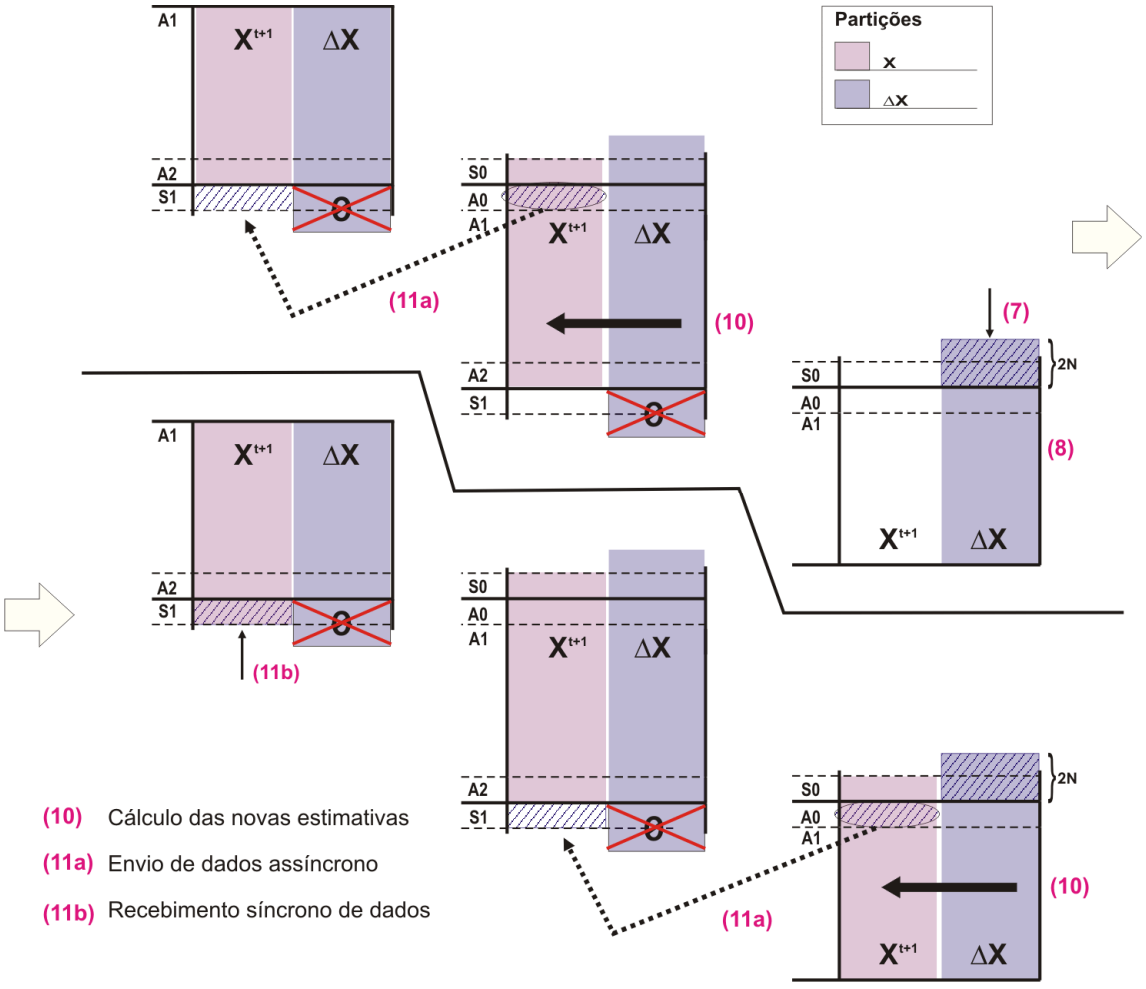


Figura 7.11 - Continuação do processo de cálculo em paralelo das novas estimativas.

Ao final do processo, todas as unidades de processamento terão em sua partição X o valor atualizado com as novas estimativas $\vec{\hat{x}}^{t+1}$. A partir desse momento, os processadores podem fazer as checagens de convergência e determinar se o processo de restauração termina ou se uma outra iteração começa (passos 2 a 11).

7.4 Processo de otimização e refinamentos da nova estratégia de implementação em paralelo

7.4.1 Resolvendo o problema de conflito de dados no cálculo das correções $\Delta\vec{\hat{x}}$

- ***Conflito de dados e uso de Gauss-Seidel***

Apesar dos cuidados tomados para se minimizar o conflito de dados no processo de cálculo das correções $\Delta\vec{\hat{x}}$ em paralelo (Item 7.3.5), nos testes que foram realizados, detectou-se que o efeito cascata do método iterativo de Gauss-Seidel ainda era muito forte, impactando negativamente nos tempos de processamento da nova estratégia. Conforme pode se observar nos gráficos da Fig. 7.12, os tempos obtidos na restauração de uma imagem de dimensão 256×256 com a nova estratégia paralela usando Gauss-Seidel, até um máximo de oito unidades de processamento, foram piores que os tempos obtidos com a versão paralela anterior.

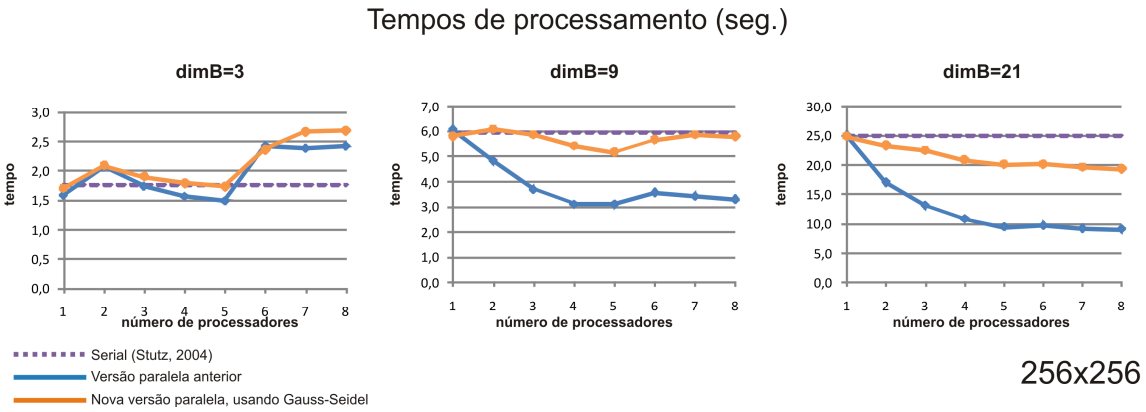


Figura 7.12 - Tempos de processamento gastos na restauração de uma imagem de dimensão 256x256 pixels, usando os programas: serial (Stutz, 2004) e versões paralelas anterior e a nova versão usando Gauss-Seidel, para um total de 100 iterações e $\dim B = \{3; 9; 21\}$.

Ao se empregar uma imagem de resolução maior (512×512 pixels), uma melhora nos tempos computacionais de ambos os programas em paralelo foi observada em relação aos tempos obtidos com o programa serial (Stutz, 2004). Apesar disso, entretanto, o desempenho da nova versão paralela empregando Gauss-Seidel, quando comparado com a versão paralela anterior, ainda deixa muito a desejar (Fig. 7.13).

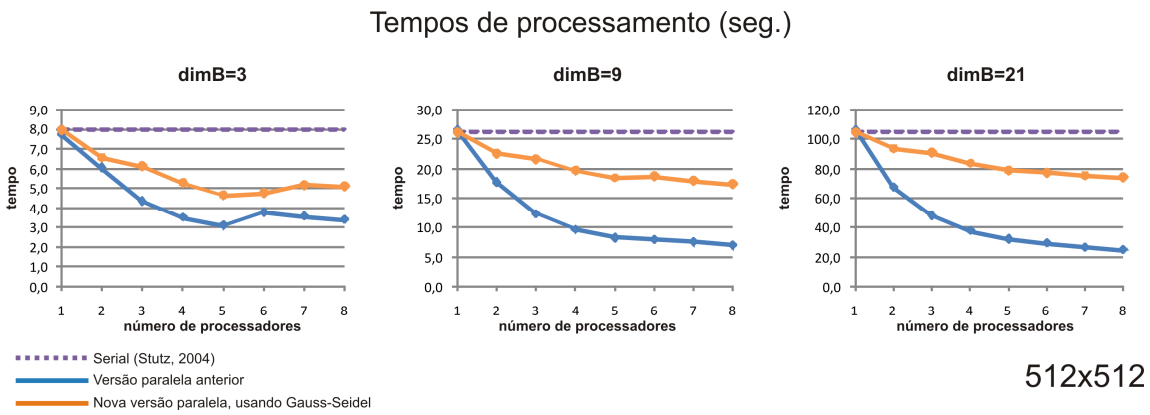


Figura 7.13 - Tempos de processamento gastos na restauração de uma imagem de dimensão 512x512 pixels, usando os programas: serial (Stutz, 2004) e versões paralelas anterior e a nova versão usando Gauss-Seidel, para um total de 100 iterações e $\dim B = \{3; 9; 21\}$.

Após os testes, verificou-se que apesar da sobrecarga de computação extra que existe na abordagem paralela anterior ter sido eliminada, o efeito cascata que ocorre no cálculo das correções $\Delta\hat{\vec{x}}$ em paralelo introduz um conflito de dados entre as unidades de processamento que torna a nova estratégia (até aqui) sem efeito e com baixo desempenho. Uma boa parcela da culpa desse problema pode ser atribuída ao método iterativo de Gauss-Seidel, Eq. (45) (ou SOR, Eq. (46)), empregado na solução do sistema linear, que cria uma relação de dependência entre os pontos e, consequentemente, entre processadores (conflito de dados). Essa dependência aumenta o tempo em que uma unidade de processamento fica aguardando por comunicação (*idle-time*), impactando diretamente nos tempos de execução do novo programa em paralelo e, por sua vez, inviabilizando o uso da nova estratégia tal como ela se apresenta até este momento.

- ***Uso do método iterativo de Jacobi na solução do problema de conflito de dados***

Para viabilizar o uso da nova estratégia é preciso melhorar o seu desempenho, solucionando o problema do conflito de dados causado pelo método de Gauss-Seidel (ou SOR). Uma solução que comumente é adotada consiste em empregar o método iterativo de Jacobi ao invés de Gauss-Seidel na solução do sistema linear que calcula as correções $\Delta\hat{\vec{x}}$. Assim, no lugar da Eq. (45) para a solução do sistema de equações lineares, pode-se empregar:

$$\Delta\hat{x}_{rs}^{t,c+1} = -\frac{1}{\left(\partial F_{rs} \big/ \partial \hat{x}_{mn}\right)\Big|_{\substack{t,c \\ m=r \\ n=s}}} \left\{ F_{rs}\Big|_{\hat{x}^{t,c}} + \sum_{\substack{m=0 \\ m\neq r}}^{M} \sum_{\substack{n=0 \\ n\neq s}}^{M} \frac{\partial F_{rs}}{\partial \hat{x}_{mn}}\Big|_{\hat{x}^{t,c}} . \Delta\hat{x}_{mn}^{t,c} \right\}, \tag{80}$$

onde $\Delta\hat{x}^{t,0} = 0$ e c é o contador de iterações do método iterativo de Jacobi.

De modo análogo ao que foi feito na Eq. (46), o elemento da diagonal principal também pode ser acrescentado à Eq. (80) de forma que esta se mantenha e, aplicando-se um fator de relaxação ω à equação resultante, obtém-se o método iterativo de Jacobi relaxado (JOR – *Jacobi OverRelaxation*) que pode ser usado no lugar do método iterativo de Jacobi, dado na Eq. (80)

$$\Delta\hat{x}_{rs}^{t,c+1} = \Delta\hat{x}_{rs}^{t,c} - \frac{\omega}{\left(\partial F_{rs} \big/ \partial \hat{x}_{mn}\right)\Big|_{\substack{t,c \\ m=r \\ n=s}}} \left\{ F_{rs}\Big|_{\hat{x}^{t,c}} + \sum_{m=0}^{M} \sum_{n=0}^{M} \frac{\partial F_{rs}}{\partial \hat{x}_{mn}}\Big|_{\hat{x}^{t,c}} . \Delta\hat{x}_{mn}^{t,c} \right\}, \tag{81}$$

onde ω é o fator de relaxação, $\mathrm{r,s} = 0,1,2,...,\mathrm{M}$, $\Delta\vec{\hat{x}}^{t,0} = 0$ e c é um contador de iterações. Para $\omega = 1$, a Eq. (80) se mantém. Caso contrário ($\omega \neq 1$), o método aplicado será o Jacobi relaxado.

- ***Avaliação de resultados do processo de restauração empregando Jacobi.***

Para permitir que o processo de restauração empregando Jacobi fosse avaliado e possibilitar uma comparação com o método de Gauss-Seidel, um grupo de imagens-testes borradas foi criado a partir da imagem-padrão (Fig. 5.4) para que fossem restauradas pelos dois métodos. Além disso, uma nova versão do programa serial empregando o método iterativo de Jacobi também foi criada, de modo que esta pudesse ser testada, comparada e produzisse os resultados aqui apresentados (Fig. 7.14 e Tabela 7.2).

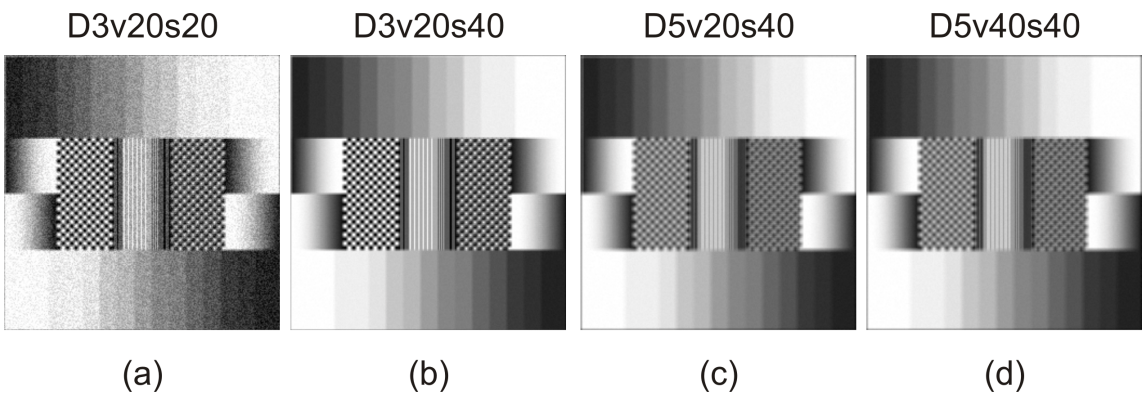


Figura 7.14 - (a-d) modificações feitas na imagem-padrão de modo a simular efeitos degenerativos.

No processo de recuperação das imagens, foram aplicados os mesmos parâmetros de restauração (Tabela 7.1), usando versões seriais do programa que implementam os dois métodos: Gauss-Seidel (Stutz, 2004) e Jacobi. Como valor de referência $\bar{x}$ inicial, empregou-se a própria imagem borrada. Além disso, tomou-se o cuidado em empregar valores que levassem em conta as características dos borramentos aplicados em cada uma das imagens-teste. Todas as imagens foram restauradas usando um número máximo de 50 iterações com aplicação da técnica de realimentação. Visto que a imagem original e as características do borramento são conhecidas, o valor adotado para o parâmetro α foi escolhido conforme foi sugerido por Branham e Katsaggelos (1997) e descrito na Eq. (56).

Tabela 7.1 – Parâmetros de restauração

Imagem	B	σ^2	α	q	γ	ω	**Realimentação**
D3v20s20	3	20	0,07424	1	0,1	1	Sim
D3v20s40	3	20	0,03002	1	0,1	1	Sim
D5v20s20	5	20	0,07744	1	0,1	1	Sim
D5v40s40	5	40	0,03049	1	0,1	1	Sim

Tabela 7.2 – Avaliação de resultados de restauração das imagens modificadas da imagem-padrão (a-d).

Métricas	Padrão modificada D3v20s20	GS	J
MAE	32,46	32,29	28,84
MSE	2834,11	2121,2	1742,6
NME	0,213977	0,212891	0,190141
NRMSE	7767,68	5813,75	4776,07
%MSE	*N/A*	74,85%	61,47%
SNR	10,4911	11,7494	12,6033
PSNR	13,6066	14,865	15,7188
IWMSE/G	31421,6	23594,2	19643,6
IWMSE/S	6327,26	4390,26	3591,8
IWMSE/V	7934,02	3973,88	3323,03
%Dif.	91,76%	87,87%	87,25%

(a)

Métricas	Padrão modificada D3v20s40	GS	J
MAE	24,64	11,79	10,58
MSE	2579,69	573,524	513,566
NME	21,0248	0,077749	0,069747
NRMSE	7070,36	1571,9	1407,57
%MSE	*N/A*	22,23%	19,91%
SNR	10,8996	17,4297	17,9093
PSNR	14,0151	20,5453	21,0248
IWMSE/G	24914,2	6799,75	6147,86
IWMSE/S	5865,29	1289,69	1096,8
IWMSE/V	7495,89	1298,89	1180,46
%Dif.	82,40%	83,15%	81,28%

(b)

Métricas	Padrão modificada D5v40s40	GS	J
MAE	31,09	18,18	16,25
MSE	3381,58	1314,51	1119,49
NME	0,20494	0,119876	0,107122
NRMSE	9268,15	3602,78	3068,22
%MSE	*N/A*	38,87%	33,11%
SNR	9,72407	13,1451	14,5251
PSNR	12,8396	16,9432	17,6407
IWMSE/G	35194,8	14301,3	12070,8
IWMSE/S	82018,6	2833,06	2344,96
IWMSE/V	8864,8	3269,92	2827,68
%Dif.	85,64%	84,77%	84,02%

(c)

Métricas	Padrão modificada D5v20s40	GS	J
MAE	30,83	18,23	17,12
MSE	3344,44	1357,98	1243,29
NME	0,203265	0,120191	0,112867
NRMSE	9166,36	3721,93	3407,58
%MSE	*N/A*	40,6%	31,17%
SNR	9,77203	13,6863	14,0695
PSNR	12,8876	16,8019	17,1851
IWMSE/G	35614,3	14752	13464,1
IWMSE/S	8023,72	2919,6	2615,46
IWMSE/V	8772,04	3524,39	3244,62
%Dif.	85,02%	85,58%	84,95%

(d)

N/A = não se aplica. %Dif. = Percentual de pontos diferentes em relação a imagem original. GS=Gauss-Seidel. J=Jacobi

Como pode ser observado na Tabela 7.2, o processo de restauração empregando o método iterativo de Jacobi apresentou, em todos os casos (D3v20s20, D3s20v40, D5v40s40 e D5v20s40), índices qualitativos melhores do que àqueles que foram obtidos com o mesmo processo de restauração empregando Gauss-Seidel. Por exemplo, o MSE e o IWMSE mais baixos informam que as diferenças entre a imagem original e as restaurações feitas empregando-se Jacobi são menores que as outras restaurações obtidas quando o método iterativo de Gauss-Seidel é empregado.

Em razão dos valores apresentados, pode-se afirmar que o emprego do método iterativo de Jacobi no processo de restauração, apresentado por Cidade (2000), é mais eficaz que o Gauss-Seidel, usado originalmente, podendo substituí-lo sem causar prejuízos à qualidade das imagens restauradas.

Além disso, uma outra vantagem dessa modificação na técnica de restauração original é que o método de Jacobi soluciona o problema de conflito de dados que ocorre quando Gauss-Seidel é empregado na solução do sistema linear para o cálculo das novas correções $\Delta\vec{\hat{x}}$ em paralelo.

- ***Método iterativo de Jacobi simplificado***

Empregando-se as mesmas restrições que foram adotadas no cálculo em paralelo das correções ΔX para minimizar o conflito de dados causado pelo método iterativo Gauss-Seidel: a) estimativa inicial $\Delta\vec{\hat{x}}^0 = 0$ e b) um número máximo de iterações igual a um, o método iterativo de Jacobi dado na Eq. (80) pode ser simplificado para

$$\Delta\hat{x}_{rs}^{t,c+1} = -\frac{F_{rs}\Big|_{\hat{x}^{t,c}}}{\left(\partial F_{rs} \Big/ \partial\hat{x}_{mn}\right)\Big|_{\substack{t,c \\ m=r \\ n=s}}}. \tag{82}$$

Essa simplificação no método iterativo de Jacobi torna a implementação do programa de restauração mais simples e, consequentemente, o processamento mais rápido (Tabela 7.3) ao eliminar do cálculo das correções ΔX um laço de repetição (*loop*) e necessidade das partições Frs e Fmn, que agora poderão ser calculadas pontualmente e armazenadas em uma variável comum.

Tabela 7.3 –.Tempos de execução obtidos com duas versões seriais do programa de restauração: uma empregando o método de Gauss-Seidel, desenvolvido por Stutz (2004), e outra o método de Jacobi simplificado, despendidos na restauração de uma imagem de 256x256 pixels, empregando um total de 100 iterações e variando a dimensão do operador B: $\dim B = \{1; 3; 5; 9; 13; 21\}$.

dimB	Tempos seriais (seg.)		Ganho
	Usando Gauss-Seidel	Usando Jacobi Simplificado	
1	1,085	0,971	-10,5%
3	1,758	1,292	-26,5%
5	2,665	1,676	-37,1%
9	5,968	2,939	-50,7%
13	11,134	4,711	-57,7%
21	24,836	10,038	-59,6%

7.4.2 Cálculo em paralelo das correções $\Delta\vec{\hat{x}}$ usando o método de Jacobi simplificado

O emprego do método de Jacobi simplificado no cálculo das correções $\Delta\vec{\hat{x}}$ em paralelo modifica os passos (5) a (11), descritos nos itens 7.3.5 e 7.3.6. Conforme ilustrado nas Figs. 7.15, 7.16 e 7.17, com o emprego o método de Jacobi simplificado, esses passos são redefinidos e passam a ser descritos da seguinte maneira:

(5-9) Com a eliminação do laço de repetição no Jacobi simplificado, os passos (5) a (9) se juntam e passam a ser apenas um: (5-9). Uma vez que as interdependências entre os pontos não existem no método de Jacobi simplificado, todas as unidades de processamento podem efetuar os cálculos das correções dentro da área-A em sua partição ΔX, de modo independente dos demais processadores (Fig. 7.15). Além disso, como o número máximo de iterações do método de Jacobi simplificado é igual a um, as unidades de processamento não precisam se comunicar para trocar dados da partição ΔX com os processadores vizinhos.

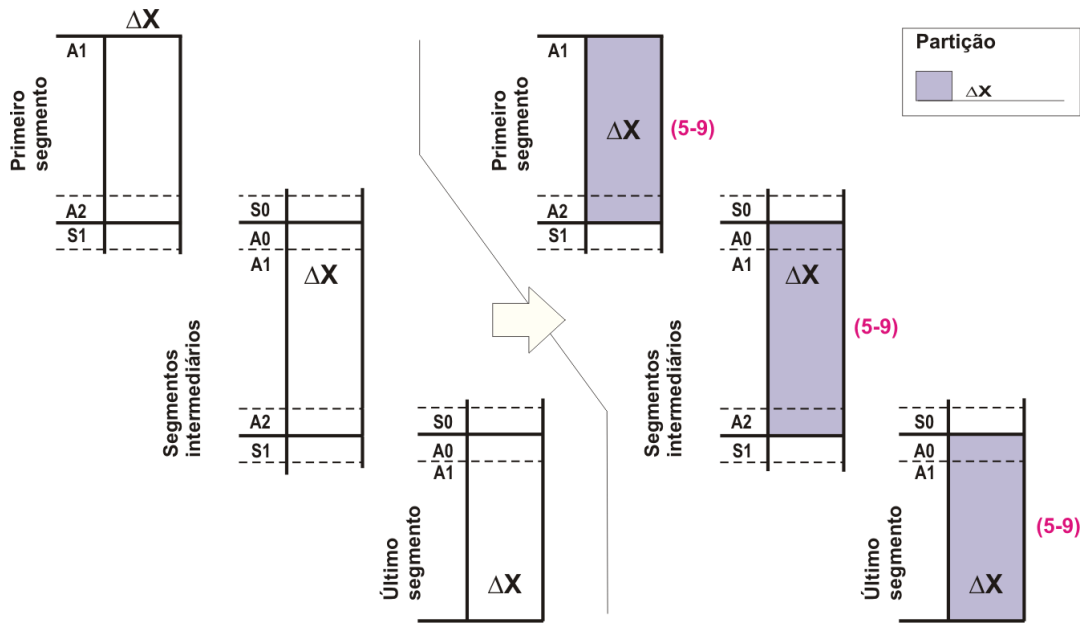


Figura 7.15 - Processo de cálculo em paralelo das correções $\Delta\hat{\vec{x}}$ empregando o método de Jacobi simplificado.

(10) Tendo obtidas as correções ΔX no passo (5-9), os processadores podem prosseguir adiante com o seu processamento, de modo independente, calculando as novas estimativas X^{t+1} $(t = 0,1,...,ni)$ dentro da área-A, restando apenas o cálculo deste termo nas áreas de sobreposição: S0 e S1 (Fig. 7.16).

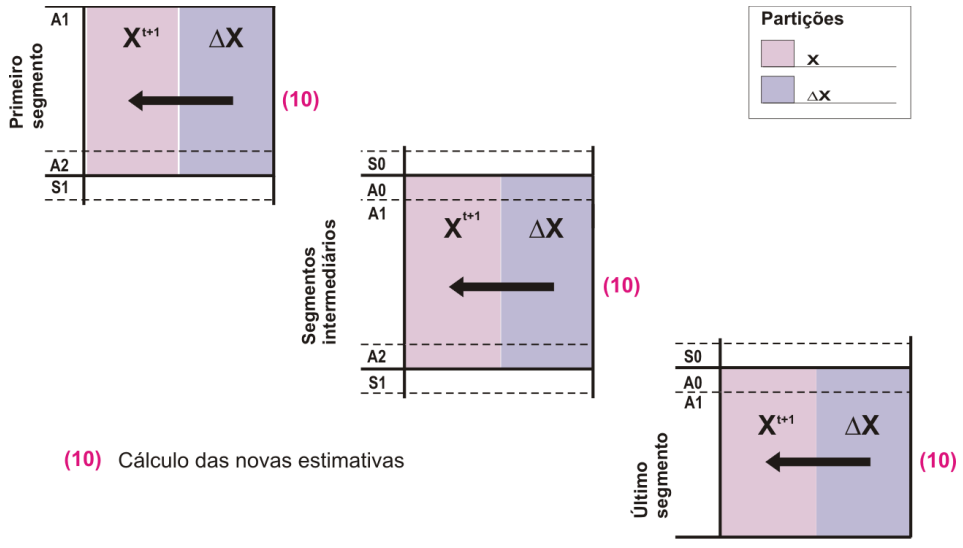


Figura 7.16 - Modificação no processo de cálculo em paralelo das novas estimativas na nova versão paralela quando o método de Jacobi simplificado é empregado.

(11) As áreas S0 e S1, que faltam no passo (10), são obtidas através da troca de dados entre as unidades de processamento acima e abaixo (Fig. 7.17),

(a) o envio dos dados pode ser executado de forma assíncrona, mas

(b) os recebimentos devem ser feitos de modo síncrono.

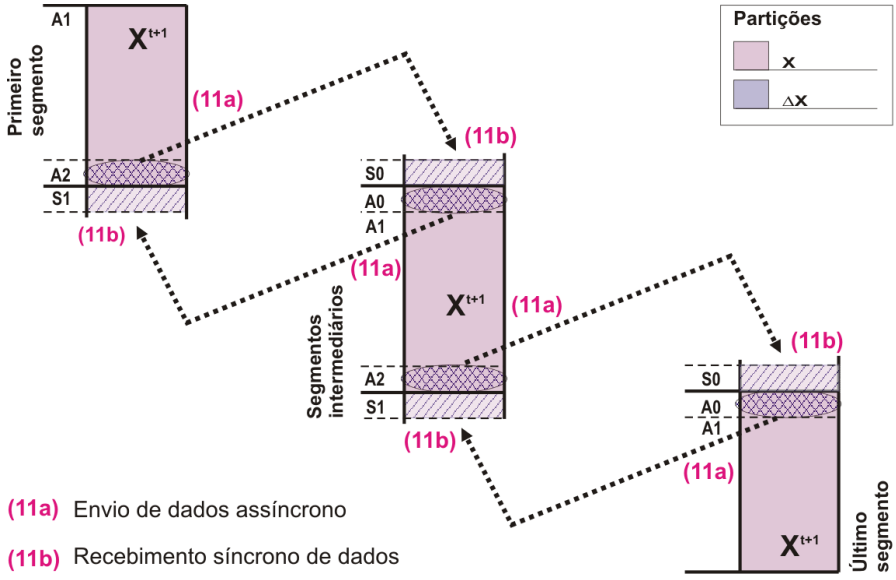


Figura 7.17 - Obtenção das áreas S0 e S1 da partição X^{t+1} que ficam faltando do processo de cálculo em paralelo das novas estimativas na nova versão paralela quando o método de Jacobi simplificado é empregado.

Nas Figs. 7.18 e 7.19 são apresentados os gráficos dos tempos de processamento dos mesmos testes feitos anteriormente (Figs. 7.12 e 7.13), porém, substituindo-se aqui os tempos do programa serial empregando Gauss-Seidel (Stutz, 2004) por outro serial empregando Jacobi simplificado e incluindo-se os tempos da nova versão do programa paralelo de restauração que emprega, também, o método iterativo de Jacobi simplificado. Pode-se observar que a nova versão paralela do programa de restauração, usando o método iterativo de Jacobi simplificado, apresentou tempos computacionais menores do que das demais implementações. Apesar do conflito de dados ter sido eliminado do cálculo das novas correções $\Delta\hat{\vec{x}}$ em paralelo, os gráficos da nova estratégia usando Jacobi simplificado apresentam algumas descontinuidades, por exemplo, quando np=3, que pode ser explicada em razão de problemas de comunicação interprocessadores (*idle-time*).

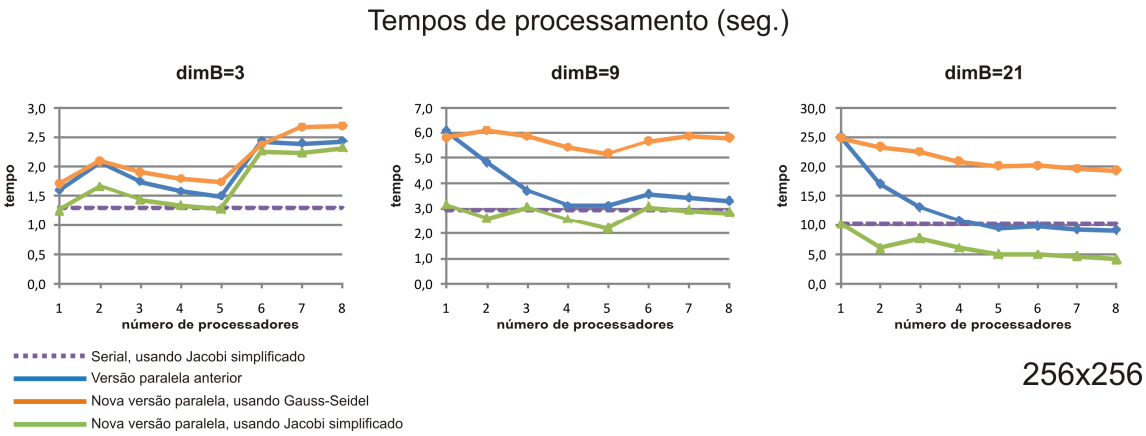


Figura 7.18 - Tempos de processamento gastos na restauração de uma imagem de dimensão 256x256 pixels, usando os programas: serial com Jacobi simplificado, primeira versão paralela e nova versão paralela usando Gauss-Seidel e Jacobi simplificado, em um total de 100 iterações e empregando $\dim B = \{3; 9; 21\}$.

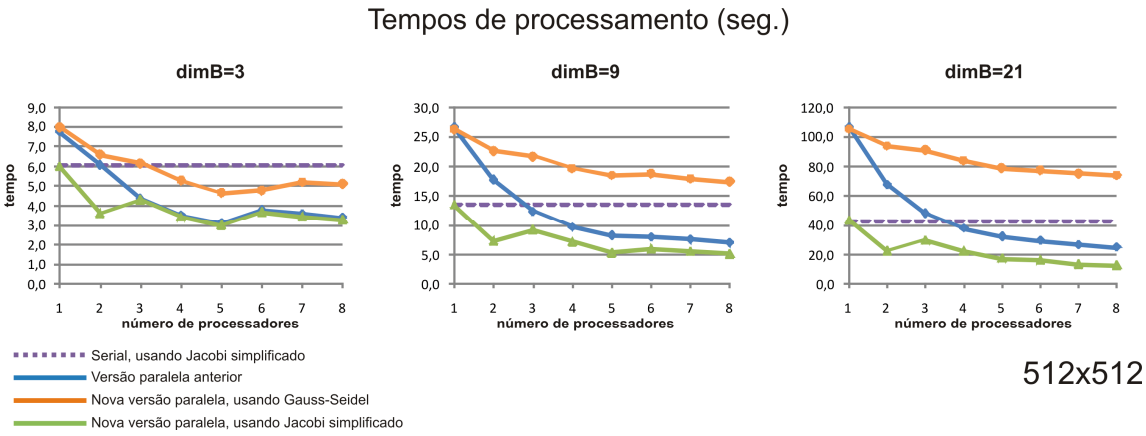


Figura 7.19 - Tempos de processamento gastos na restauração de uma imagem de dimensão 512x512 pixels, usando os programas: serial com Jacobi simplificado, primeira versão paralela e nova versão paralela usando Gauss-Seidel e Jacobi simplificado, em um total de 100 iterações e empregando $\dim B = \{3; 9; 21\}$.

7.5 Redução dos tempos de comunicação

Conforme descrito no item 3.5, as sobrecargas (*overheads*) em um programa paralelo podem se originar de diversas formas. Nem sempre, entretanto, a sua origem é clara e, quando ela é identificada, a sua solução nem sempre é simples e/ou de fácil resolução. Basta observar, por exemplo, todo o esforço que já foi feito até aqui para se otimizar e refinar a nova estratégia em paralelo (estratégias para o cálculo dos termos em paralelo, adoção do método

iterativo de Jacobi no lugar de Gauss-Seidel etc.), bem como outras soluções que já foram implementadas na versão paralela anterior (técnicas de particionamento, janelas de checagem, processo de distribuição de cargas etc.).

- ***Visualização do processo de comunicação através do MPE e do programa Jumpshot-4***

Para entender melhor os processos de comunicação do programa paralelo da nova versão usando MPI (*Message Passing Interface*) é preciso fazer uso de ferramentas que permitam a visualização do que realmente ocorre dentro de um ambiente de execução em paralelo. Se por um lado o MPI fornece uma base para a construção de programas paralelos, por outro lado, a biblioteca MPE (*Multi-Processing Environment*) (Chan et al., 1998) fornece uma série de recursos que auxiliam o programador MPI, que incluem: a) um conjunto de sub-rotinas para a criação de arquivos de logs (arquivos contendo registros das operações de processamento em um computador), b) rotinas para depuração de programas, c) rotinas de sequencialização etc. O MPE oferece diversas maneiras de se gerar arquivos de log que descrevem o andamento do processo de execução dos programas paralelos. Esses arquivos podem ser visualizados e examinados por um programa gráfico, o Jumpshot-4 (Chan et al., 2007), que é distribuído junto com o MPE. Os arquivos de logs são gerados em um formato conhecido como CLOG2 e a facilidade com que eles são criados, permite que os programas MPI não precisem ser modificados, bastando apenas (re)compilá-lo com a biblioteca MPE. Essa biblioteca contém funções com perfis semelhantes às da biblioteca MPI que interceptam as chamadas MPI da aplicação, gerando os arquivos de log de uma forma bem simples e transparente.

Na Fig. 7.20 é mostrada a visualização de um log de execução do programa paralelo de restauração através do programa Jumpshot-4. No lado esquerdo da figura, tem-se a janela de legendas com uma relação de nomes, representação gráfica etc. dos objetos (mensagem, chamadas/esperas de funções MPI, início/término do processamento paralelo etc.) que são visualizados na janela de linha do tempo (lado direito da figura). No centro dessa outra janela, tem-se a linha de tempo da execução em paralelo da nova versão paralela do programa de restauração usando Jacobi, empregando-se três unidades de processamento para a restauração de uma imagem de 256x256 pixels– identificados pelos números de 0, 1 e 2, conforme pode se observar no lado esquerdo das linhas de tempo. Abaixo fica o eixo de coordenada-tempo em segundos.

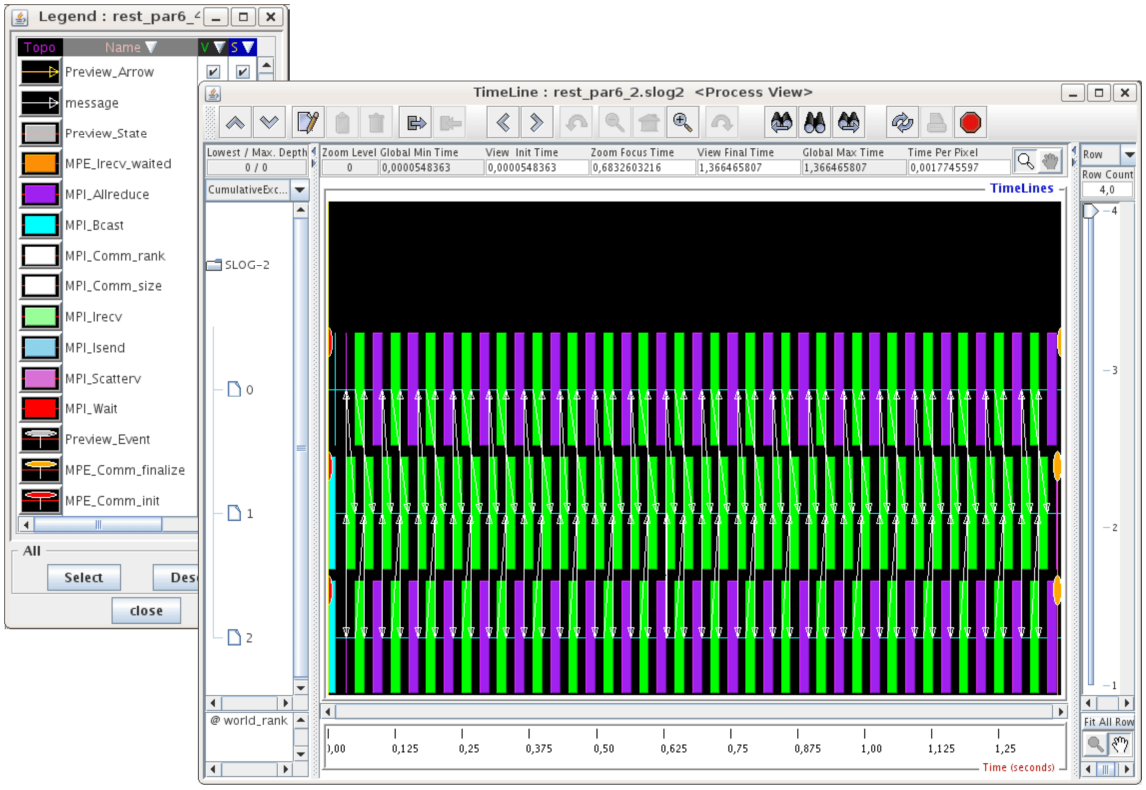


Figura 7.20 - Janelas do programa Jumpshot-4: legendas (esquerda) e a de linha de tempo (direita) da execução da nova versão paralela do programa de restauração de imagens empregando Jacobi, usando três unidades de processamento. Os blocos verdes e roxos em destaque representam, respectivamente, os tempos de comunicação gastos com as funções MPI_Recv() e MPI_Allreduce().

Na Fig. 7.21 é mostrado em detalhes (zoom) o processo de comunicação que ocorre, por exemplo, na primeira iteração do laço principal do programa de restauração em paralelo ilustrado na Fig. 7.20. Cada um dos blocos coloridos representa a quantidade de tempo consumida por uma chamada de comunicação interprocessadores e os espaços nas linhas de tempos entre esses blocos representam os tempos de execução de instruções do programa de restauração em paralelo que funcionam de forma independente dos outros processadores.

Quanto maiores forem esses blocos, maiores serão os tempos de espera (*idle-time*) por comunicação e, consequentemente, maiores serão os tempos de execução do programa. Por outro lado, quanto menores forem os blocos, mais rápido será a execução do programa em paralelo.

Para se ter uma idéia de proporção, as duas janelas menores (“*Drawable Info Box*”) mostram em detalhes os tempos gastos (*duration*) pelo processador-0 na execução das funções *MPI_Recv()* (16,85405*ms*, bloco verde) e *MPI_Allreduce()* (16,96181*ms*, bloco roxo). A primeira função é responsável pelo recebimento (*receive*) de um bloco de dados e a

segunda pelo cálculo do mínimo do funcional Q, Eq. (37), distribuído a todas as unidades de processamento e que é usado como critério para aceite/rejeite da imagem recém-restaurada como solução do problema e, também, para continuação/encerramento do programa de restauração.

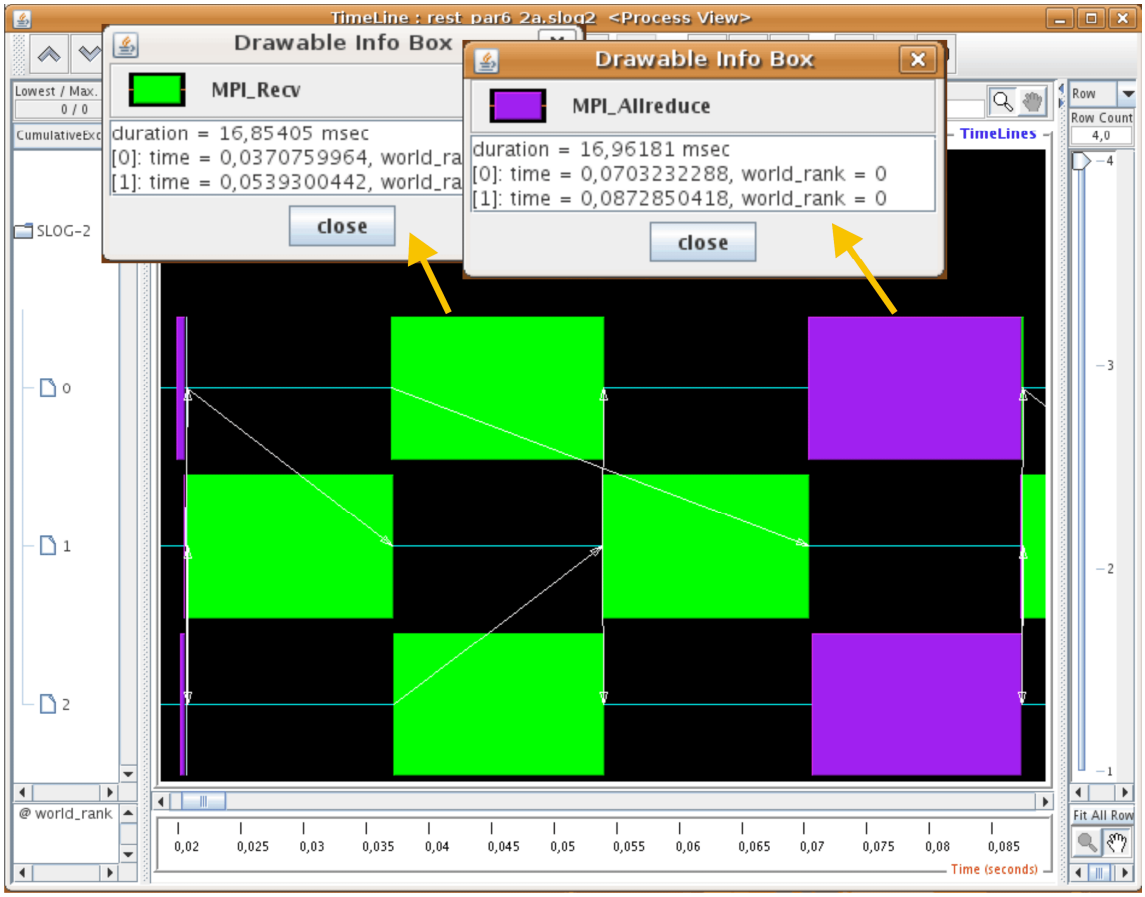


Figura 7.21 - Zoom da linha de tempo, mostrando a execução em paralelo da primeira iteração do laço principal da nova versão paralela do programa de restauração de imagens empregando Jacobi. Os blocos verdes e roxos em destaque representam, respectivamente, os tempos de comunicação gastos com as funções MPI_Recv() e MPI_Allreduce().

- ***Reduzindo as latências de comunicação através de requisições antecipadas de pedido de recebimento de mensagem (Message prefetching)***

Analisando-se detalhadamente as Figs. 7.20 e 7.21, observa-se, pelos tamanhos dos blocos de comunicação, que um tempo muito grande (latência) ocorre do início ao fim de cada bloco. Isso eleva os tempos de espera (*idle-time*) que consomem uma boa parcela do tempo de execução do programa em paralelo, diminuindo muito o seu desempenho. Ke et al.

(2005) investigaram esse problema e, como solução, propuseram uma técnica em que a latência de comunicação poderia ser reduzida através do envio antecipado de pedidos de recebimento de mensagens (*message prefetching*) ao remetente (*sender*) antes mesmo deles terem sido efetivamente enviados (*send*).

A biblioteca MPI fornece um bom conjunto de operações para comunicação ponto a ponto, comunicações coletivas e operações sincronizadas. A operação básica de recebimento de dados (*receive*) do MPI é feita através da função *MPI_Recv()* e de envio através da função *MPI_Send()*. A chamada da função *MPI_Recv()* é um processo bloqueante. Isso significa que a função só retorna o controle da execução do programa ao local de chamada quando toda a mensagem tiver sido recebida. Para contornar essa situação, o MPI incluiu em sua biblioteca uma outra função de recebimento não bloqueante, o *MPI_Irecv()*, que retorna imediatamente logo após a sua chamada, tendo ou não completado o recebimento da mensagem. Quando a mensagem for realmente necessária (ou seja, os dados), faz-se então uma chamada a função *MPI_Wait()* (bloqueante) que irá aguardar até que a mensagem tenha sido totalmente recebida. Isso permite que se inclua alguma programação entre as chamadas *MPI_Irecv()* e *MPI_Wait()* de modo que algumas instruções possam ser executadas em paralelo, enquanto a mensagem não chega, substituindo, desse modo, tempos de espera por comunicação (*idle-time*) por tempos de processamento, escondendo parte da latência.

Na Fig. 7.22, é ilustrada uma comparação entre os métodos convencional de envio/recebimento e a técnica de envio antecipado de pedido de recebimento de mensagens (*message prefetching*). O padrão vertical representa a fase de computação normal, a horizontal a fase de comunicação mais o tempo de espera (*idle-time* – parte escura) e a diagonal a sobreposição das fases de comunicação e computação. Enquanto uma mensagem bloqueante é feita (*MPI_Recv()* ou *MPI_Wait()*) no método tradicional, a implementação usando *message prefetching* antecipa o processo de recebimento fazendo uma chamada não-bloqueante a função *MPI_Irecv()*, que reserva a área de memória necessária para receber os dados que ainda serão enviados pela função *MPI_Send()* e libera o processador para que execute outras instruções em paralelo. Quando a função *MPI_Send()* é finalmente feita pelo remetente (*sender*), uma comunicação mais rápida é feita, uma vez que parte do processo de comunicação já havia sido feito antecipadamente no destinatário (*receiver*), quando este fez uma chamada à função *MPI_Irecv()*. Assim, conforme ilustrado na Fig. 7.22, a técnica tem o potencial para esconder parte da latência da comunicação e, consequentemente, melhorar o desempenho do programa paralelo (Ke et al., 2005).

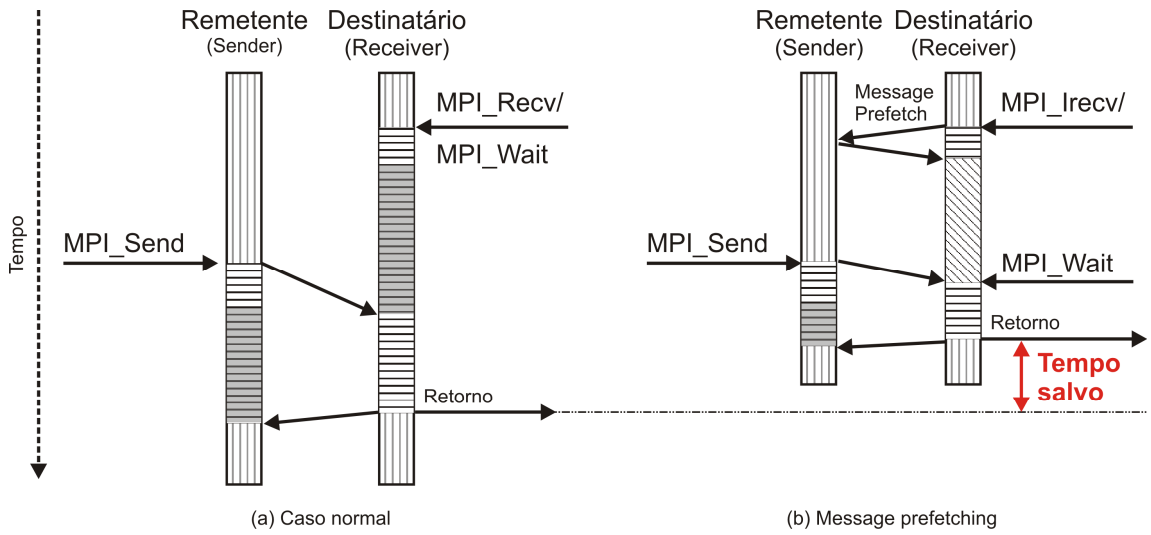


Figura 7.22 - Comparação entre (a) uma comunicação paralela normal e outra (b) empregando *message prefetching* (Ken et al., 2005).

O emprego da técnica *message prefetching* na nova estratégia modifica o passo (2b), apresentado no item 7.3.2, e o passo (10), apresentado inicialmente no item 7.3.6 e depois modificado no item 7.4.2 em razão do emprego do Jacobi simplificado. Com o *message prefetching*, esses passos são redefinidos e passam a ser descritos da seguinte maneira:

(2b) Executam-se requisições antecipadas de recebimento (*message prefetching*) das áreas S0 e S1 da partição $Y - BX$ às unidades de processamento vizinhas acima e abaixo, respectivamente. A seguir, calcula-se a área-A da partição $Y - BX$, faltando apenas o cálculo deste termo nas áreas de sobreposição, S0 e S1. Estas áreas serão obtidas através de chamadas à função *MPI_WAIT()* feitas posteriormente no passo (4a), conforme descrito no item 7.3.4.

(10) Executam-se requisições antecipadas de recebimento (*message prefetching*) da área S0 e S1 da partição X^{t+1} às unidades de processamento vizinhas acima e abaixo, respectivamente. A seguir, calcula-se as novas estimativas X^{t+1} ($t = 0,1,...,ni$) da área-A, restando o cálculo dos termos nas áreas de sobreposição S0 e S1, que serão obtidas através da chamada a função *MPI_WAIT()* feitas no passo seguinte.

Na Fig. 7.23, é ilustrado o log de execução da nova versão paralela do programa de restauração empregando Jacobi simplificado mais a técnica *message prefetching*. Nota-se claramente nesta figura que houve uma redução significativa no tamanho dos blocos de comunicação. Comparando-se com a Fig. 7.20 (Fig. 7.24), percebe-se uma redução substancial nos tempos de comunicação e no tempo de execução do programa paralelo.

Sem a aplicação da técnica, a nova versão do programa paralelo levou em torno de 1,35*s* para fazer uma restauração de uma imagem de 256x256 pixels, empregando 20 iterações e dimB=21. Com a aplicação do *message prefetching*, o mesmo programa gastou aproximadamente 0,70*s* para fazer a mesma restauração. Isso representa um ganho de velocidade de processamento da ordem de 48,2% com o uso do *message prefetching*, só nesse caso.

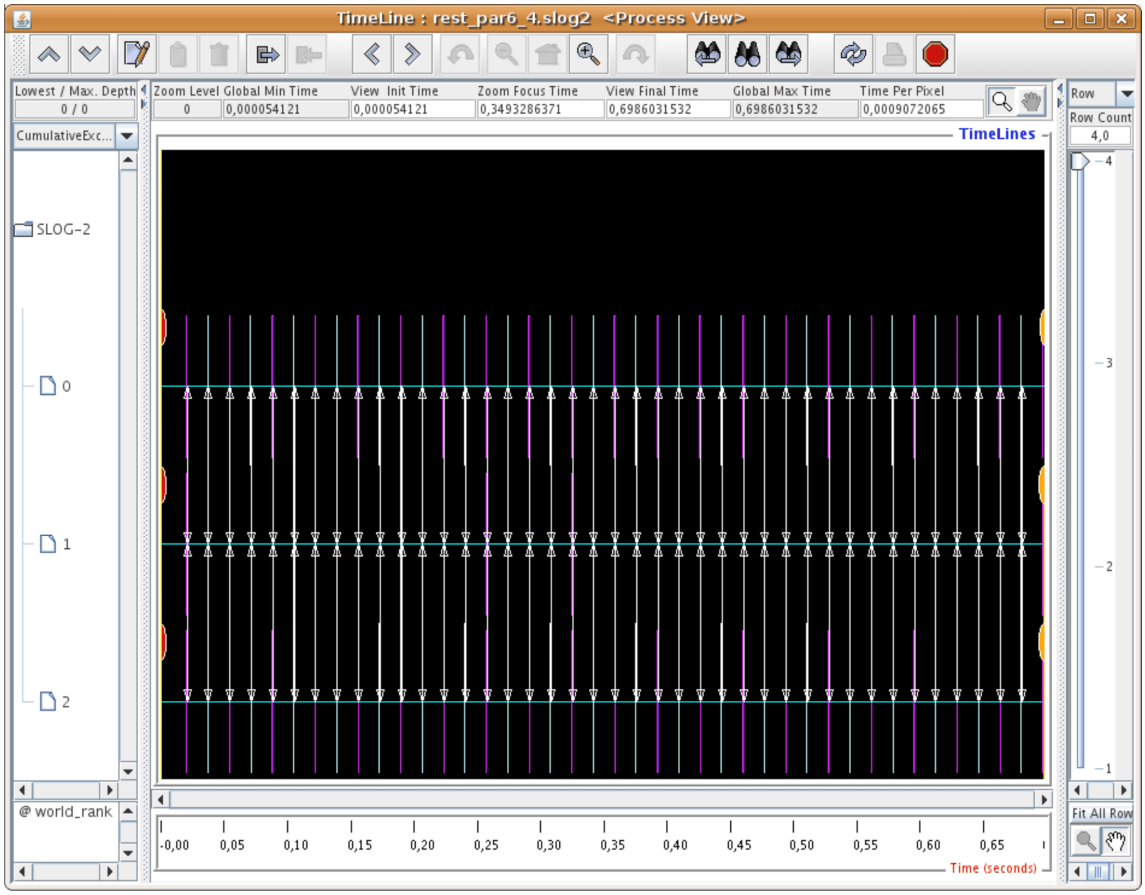


Figura 7.23 - Linha de tempo da execução da nova versão paralela do programa paralelo de restauração de imagens, usando Jacobi e *message prefetching*.

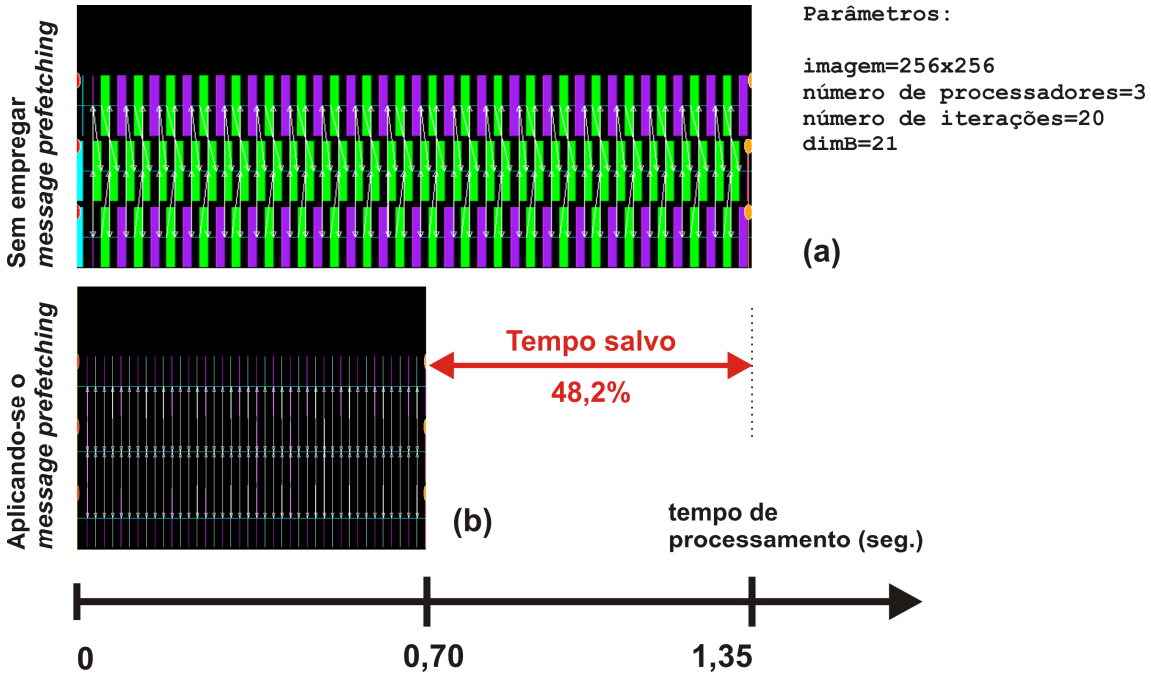


Figura 7.24 - Comparação entre os tempos de execução da nova versão do programa paralelo de restauração sem aplicar a técnica *Message Prefetching* (a) e com a aplicação dela (b).

Na Fig. 7.25, é mostrado o zoom do processo de comunicação que ocorre na primeira iteração do laço principal da nova versão paralela do programa de restauração em paralelo ilustrado na Fig. 7.23. Comparando-se com a Fig. 7.21, nota-se que, com a aplicação da técnica *message prefetching*, os blocos de comunicação são menores e, portanto, mais rápidos. Uma chamada à função *MPI_WAIT()*, por exemplo, levou 0,52595*ms*, enquanto que à *MPI_Allreduce()* levou 0,58603*ms*. Outra chamada à função *MPI_Irecv()* foi tão rápida (5,9046μs) que quase não se nota a sua presença no gráfico.

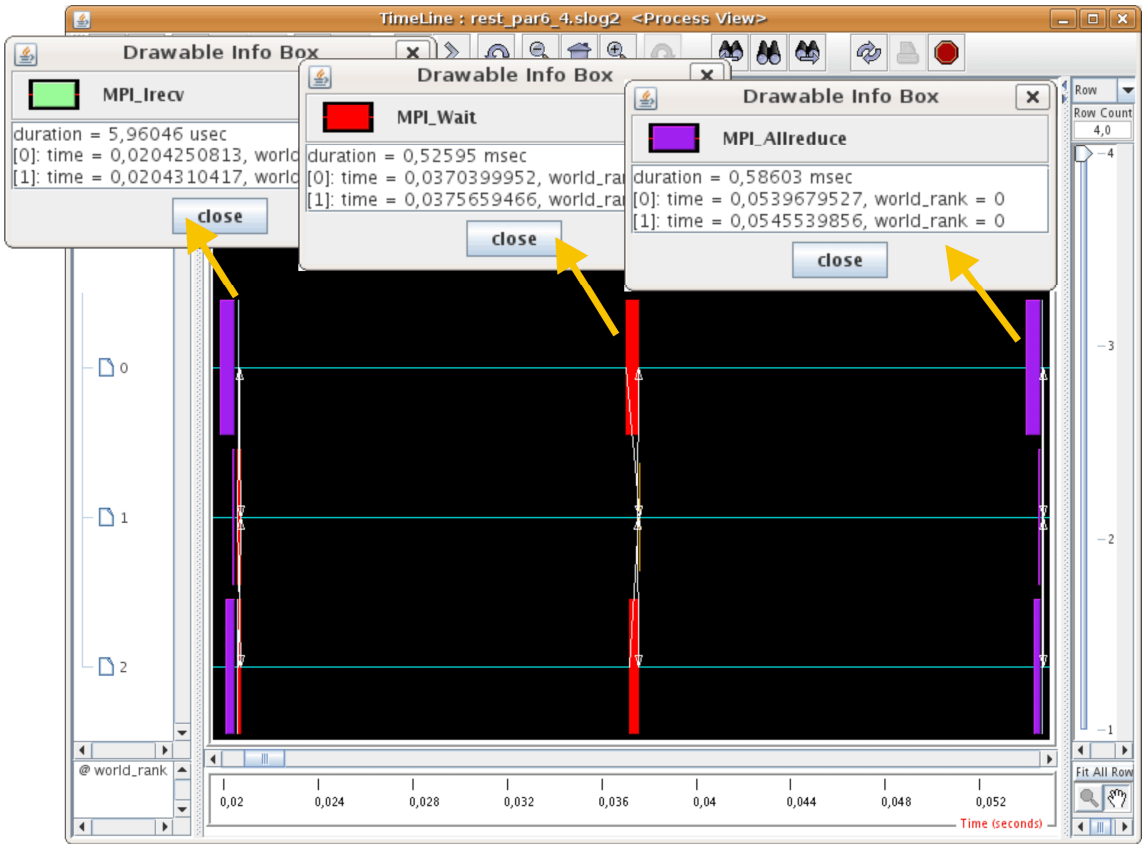


Figura 7.25 - Zoom da linha de tempo, mostrando a execução em paralelo da primeira iteração do laço principal da nova versão paralela do programa de restauração de imagens usando Jacobi mais a técnica *message prefetching*.

Na Figs. 7.26 e 7.27, são apresentados alguns gráficos dos tempos de processamento dos testes feitos anteriormente (Figs. 7.18 e 7.19), incluindo-se aqui os tempos da nova versão paralela do programa de restauração empregando Jacobi mais a técnica *message prefetching*. O pseudocódigo completo da nova versão programa de restauração usando a nova estratégia é apresentado no Apêndice-C.

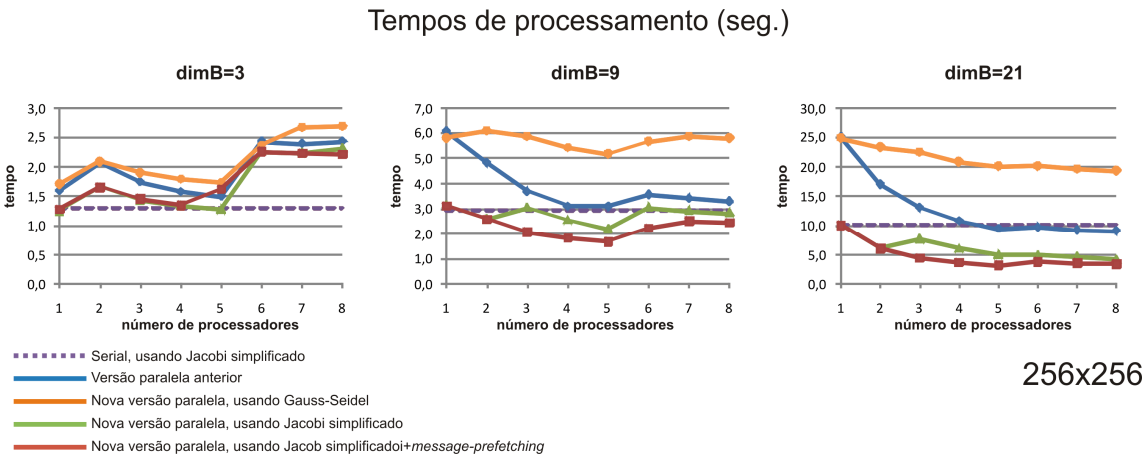


Figura 7.26 - Tempos de processamento gastos na restauração de uma imagem de dimensão 256x256 pixels, usando os programas: serial e versões paralelas anterior e nova usando Gauss-Seidel, Jacobi e Jacobi+*message prefetching*, para um total de 100 iterações e $\dim B = \{3; 9; 21\}$.

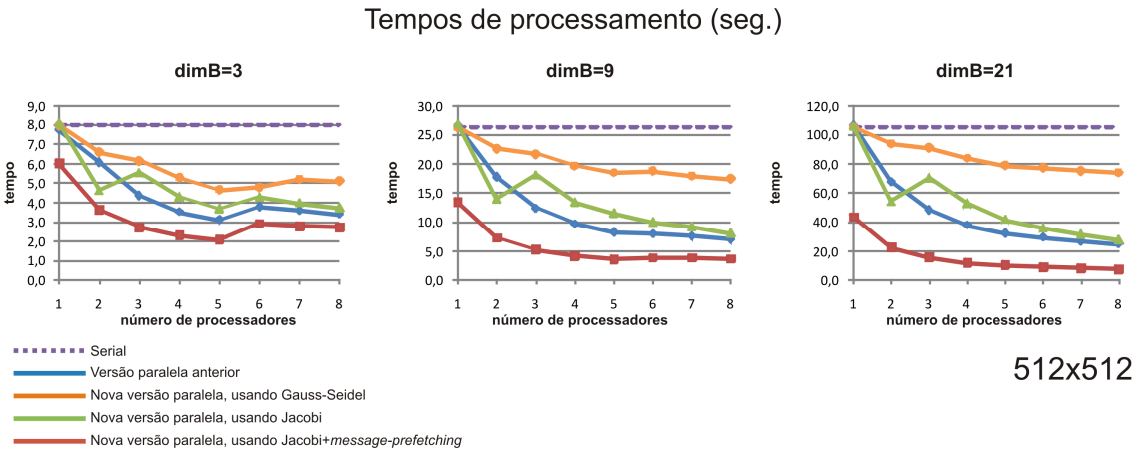


Figura 7.27 - Tempos de processamento gastos na restauração de uma imagem de dimensão 512x512 pixels, usando os programas: serial e versões paralelas anterior e nova usando Gauss-Seidel, Jacobi e Jacobi+*message prefetching*, para um total de 100 iterações e $\dim B = \{3; 9; 21\}$.

8 RESULTADOS E DISCUSSÕES

8.1 Metodologia de avaliação

Para se avaliar a estratégia de implementação da nova versão paralela do programa de restauração e compará-la com as versões anteriores que implementam o mesmo algoritmo de restauração, serial e primeira versão paralela, fez-se necessário estabelecer uma metodologia de avaliação de maneira que os resultados obtidos refletissem de forma fiel e real as medidas de qualidade e de desempenho alcançados com as referidas implementações na arquitetura usada. Para que esse objetivo pudesse ser alcançado, alguns procedimentos e cuidados foram tomados:

- ***Avaliação de desempenho***

a) A avaliação do desempenho dos programas em paralelo teve com base de comparação uma versão serial otimizada empregando Jacobi simplificado que é uma versão modificada do programa de restauração desenvolvido durante o período de mestrado (Stutz, 2004), cuja codificação faz uso eficiente de recursos computacionais e emprega técnicas avançadas de programação;
b) Cada um dos tempos de execução apresentado nos gráficos foi obtido através da média aritmética dos tempos alcançados em três execuções completas dos programas de restauração: a) serial, b) primeira versão paralela e c) a nova versão paralela empregando Jacobi simplificado mais *message prefetching*, numa mesma arquitetura de hardware, num total de 100 iterações completas, aplicando-se os mesmos parâmetros de restauração;
c) De modo que as versões em paralelo pudessem ser comparadas de forma igualitária, as execuções da primeira versão paralela do programa de restauração empregaram uma janela de verificação igual a um, de forma que, a cada iteração, houvesse uma parada em ambos os programas para a verificação de convergências. Uma janela de verificação maior poderia interferir nos tempos de execução obtidos com a primeira versão do programa paralelo, reduzindo-os. Porém, isso também afetaria a qualidade das restaurações, piorando-as (para

maiores detalhes, veja “Janelas de verificação” no item 6.2). É importante salientar que além do desempenho dos programas de restauração, a qualidade das restaurações é também um dos quesitos avaliados nesse trabalho;

d) Na avaliação de desempenho foram empregadas cinco diferentes resoluções de imagem: 128x128 (imagem pequena), 256x256, 512x512, 1024x1024 e 2048x2048 (imagem muito grande para um AFM). A primeira imagem (resolução pequena) foi usada para se avaliar o comportamento dos programas na restauração de pequenos volumes de dados (p.ex. restauração de partes específicas de uma imagem). A última (imagem de alta resolução) foi usada para se analisar o comportamento dos programas na restauração de grandes volumes de dados, visto que ao longo do tempo os equipamentos (AFM) têm evoluído e gerado imagens com resoluções cada vez maiores. As demais resoluções (intermediárias) foram usadas para se avaliar o comportamento dos programas à medida que as dimensões das imagens fossem aumentadas;

e) O esforço computacional exigido para se restaurar essas imagens e o volume de dados comunicado entre os processadores em paralelo foi variado, empregando-se diferentes tamanhos do operador de borramento B: $\dim B = \{1, 3, 5, 9, 13, 21\}$. Quanto maior esse valor, maior é o esforço computacional e o volume de dados comunicados entre os processadores. Entretanto, dada a quantidade de gráficos produzidos nesse item e para não sobrecarregar demais esse capítulo com um volume muito grande de informações, foram escolhidas três dimensões de B: $\dim B = \{3,\ 9,\ 21\}$. O conjunto completo de gráficos é apresentado no Apêndice-D;

f) Por questões de comparabilidade entre os resultados obtidos com as duas versões do programa paralelo de restauração, na execução da primeira versão paralela do programa de restauração empregou-se uma área de sobreposição igual a N, cujo valor é obtido de $\dim B$, estabelecido *a priori*, onde $N = \frac{\dim B - 1}{2}$. Esse valor representa o tamanho mínimo que deve ser usado para se minimizar o efeito-borda da primeira versão paralela (para maiores detalhes, veja “Solução para o problema do efeito-borda” no item 6.2) e que é compatível com o tamanho da área de sobreposição (S0 e S1) empregada na nova versão paralela do programa de restauração.

- ***Avaliação da qualidade dos resultados***

a) Para a avaliação dos resultados e da qualidade das imagens recuperadas pelos programas de restauração, foram utilizadas duas imagens conhecidas, a imagem-padrão (Fig. 5.4) e uma imagem extraída de um trecho de um livro, como imagens originais e, de modo a simular os efeitos degenerativos produzidos por um AFM durante o processo de aquisição de imagens (borramento e ruído aditivo), com essas imagens produziu-se um conjunto de imagens-testes borradas para que fossem recuperadas pelos programas de restauração e, ao final do processo, os resultados encontrados (restaurações) fossem avaliados e comparados qualitativamente com a imagem original conhecida;

b) O conjunto de imagens-testes borradas foi gerado a partir das imagens originais, conforme o método descrito no item 5.3.2, variando-se os parâmetros $\dim B$, SNR e σ^2;

c) Todas as restaurações empregaram um total de 50 iterações completas, com realimentação, sendo que os programas em paralelo utilizaram oito unidades de processamento e a primeira versão paralela empregou uma janela de verificação igual a um e uma área de sobreposição igual a $N = \frac{\dim B - 1}{2}$;

d) Na restauração das imagens borradas, empregaram-se os mesmos parâmetros usados para se borrar as imagens-testes e o valor do parâmetro α adotado foi escolhido conforme sugerido por Branham e Katsaggelos (1997), Eq. (56);

e) Para exemplificar um caso real de restauração de imagens e para demonstrar a eliminação do efeito-borda das restaurações feitas com a nova versão paralela do programa de restauração, algumas imagens de AFM foram restauradas, empregando-se o programa serial implementado com o método iterativo de Jacobi e a nova versão paralela do programa de restauração.

- ***Arquitetura empregada***

Os resultados apresentados nesse trabalho foram obtidos através de uma estação de trabalho adquirida com verbas de projeto CNPq no final de 2008 e que se encontra atualmente instalada no Laboratório de Experimentação e Simulação Numérica em Transferência de

Calor e Massa (LEMA – http://www.lema.iprj.uerj.br/) do Campus Regional Nova Friburgo IPRJ/UERJ. O equipamento é um Dell Precision Workstation T7400, com dois processadores Intel® Xeon® Quad-Core E5405 2 GHz, 2X6 MB L2 cache, 1333 MHz FSB - totalizando oito unidades de processamento -, Memória de 4 GB DDR2 SDRAM 667MHz, placa de vídeo PCIe x16 nVidia Quadro FX570 256MB, 4 discos rígidos de 300 GB SAS (15.000 rpm) em RAID 5, placa de rede 10/100/1000 Gigabit Ethernet Broadcom NetXtreme PCI Express, monitor 22 polegadas UltraSharp™ 2208FPW Widescreen, com dual boot: Windows Vista® Business SP1 (instalação original) e o sistema operacional Ubuntu 4.24 (Linux 2.6.24-24 generic), instalado posteriormente. As aplicações serial e em paralelo foram todas elas executadas no ambiente Linux usando os pacotes mais recentes dos programas OpenMPI 1.2.5 e MPE 2-1.0.6p1.

- ***Considerações***

Cabe aqui ressaltar que os tempos computacionais que são apresentados nesta tese podem variar de acordo com a arquitetura empregada. Diferentes arquiteturas paralelas produzirão diferentes resultados e, consequentemente, resultarão em diferentes níveis de desempenho. As conclusões aqui apresentadas levam em conta apenas os resultados obtidos com a arquitetura aqui descrita, mas isso de maneira alguma invalida a aplicação das técnicas computacionais empregadas na estratégia usada para se implementar a nova versão paralela do programa de restauração de imagens.

8.2 Análise de desempenho

8.2.1 Estudo de caso I – Restauração de uma imagem de dimensão 128x128 pixels

Na Fig. 8.1 são mostrados os gráficos das medidas de desempenho (tempo de execução, aceleração, eficiência e custo) obtidas com as restaurações feitas com os programas: serial, primeira versão paralela e nova versão paralela, empregando-se três diferentes dimensões de B: $\dim B = \{3, 9, 21\}$.

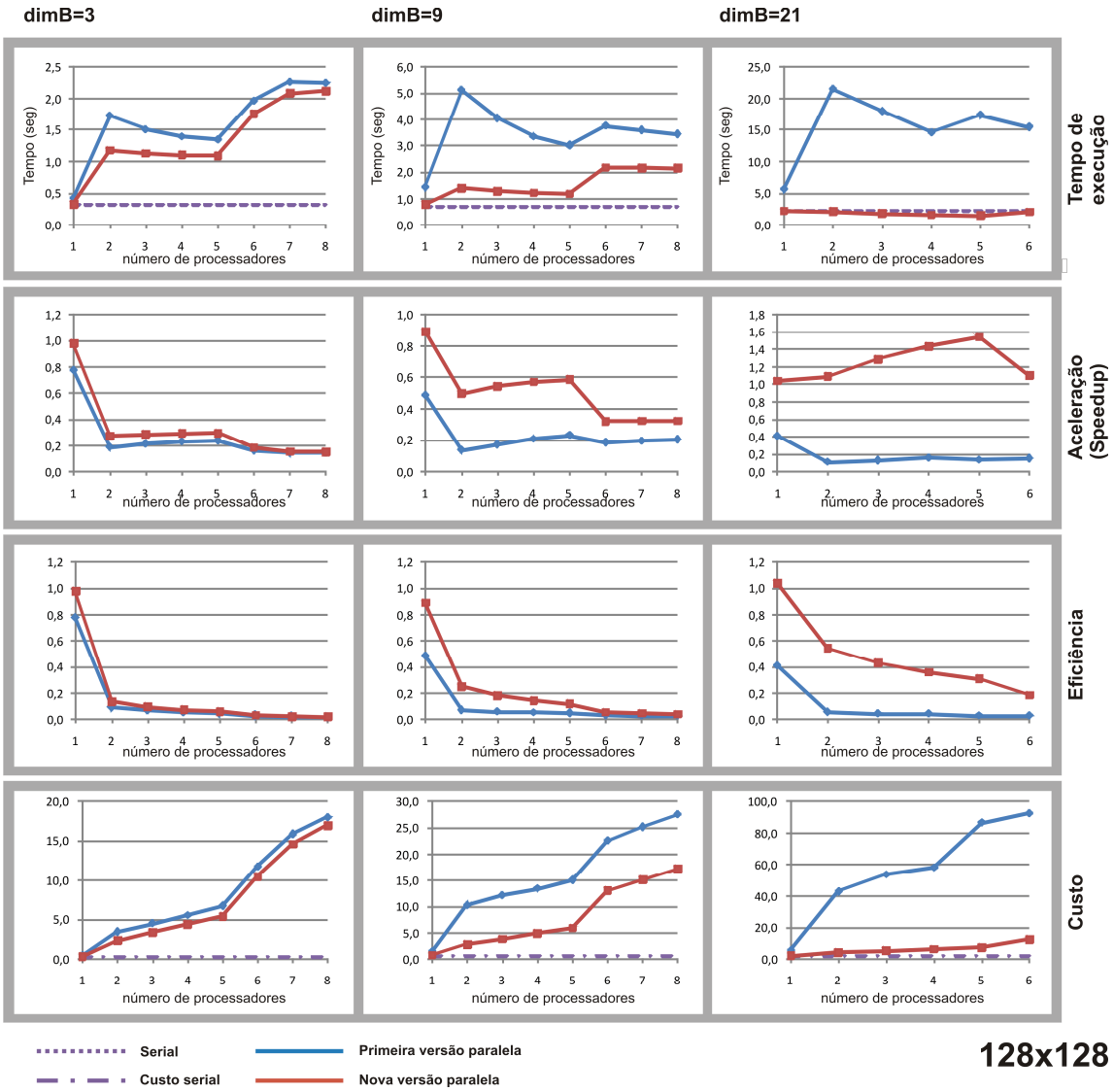


Figura 8.1 - Gráficos de medidas de desempenho obtidas com restaurações de uma imagem de 128x128 pixels (Caso I).

Analisando-se os gráficos de desempenho dos programas paralelos, observa-se que ambas as estratégias paralelas não são eficientes no tratamento de imagens de resoluções pequenas (128x128 ou menores), usando mais de um processador. Esse comportamento é esperado, visto que a aplicação da técnica de computação paralela só se justifica nos casos em que há um processamento intenso e/ou um volume de dados grande para ser processado que compensem os tempos gastos com as sobrecargas (*overheads*) da computação paralela e, ainda assim, produza algum ganho no tempo de execução e faça uma melhor aplicação dos recursos computacionais (eficiência e custo).

Um menor número de processadores observado nos gráficos da Fig. 8.1, quando $\dim B$=21, ocorre em razão da quantidade de processadores (np) usada na restauração de uma imagem em paralelo possuir um limite máximo que varia de acordo com o tamanho da imagem restaurada e a dimensão do operador de borramento B ($\dim B$), ambos conhecidos *a priori*. O tamanho de uma partição, descontando-se as áreas sobreposição (S0 e S1), não pode ser inferior a $\dim B$, para que não haja sobreposição das áreas S0 e S1 e que exista ao menos uma linha de imagem na área-A para ser restaurada. Ou seja,

$$\frac{M}{np} > \dim B \Rightarrow 1 \le np \le \text{int}\left(\frac{M}{\dim B}\right) \tag{83}$$

onde $M \times M$ é a dimensão da imagem, np o número de processadores, $\dim B$ a dimensão do operador de borramento B e $\text{int}\left(\frac{M}{\dim B}\right)$ é a divisão inteira de M por $\dim B$.

Dessa restrição, Eq. (83), resulta que, para imagens de dimensão 128x128 com $\dim B = 21$, por exemplo, a quantidade de processadores em paralelo não pode exceder o número máximo de seis unidades.

8.2.2 Estudo de caso II – Imagem de 256x256

Na Fig. 8.2, são mostrados os gráficos das medidas de desempenho obtidas com três implementações (serial, primeira versão paralela e nova versão paralela) do algoritmo de restauração no processamento de uma imagem com resolução de 256x256 pixels.

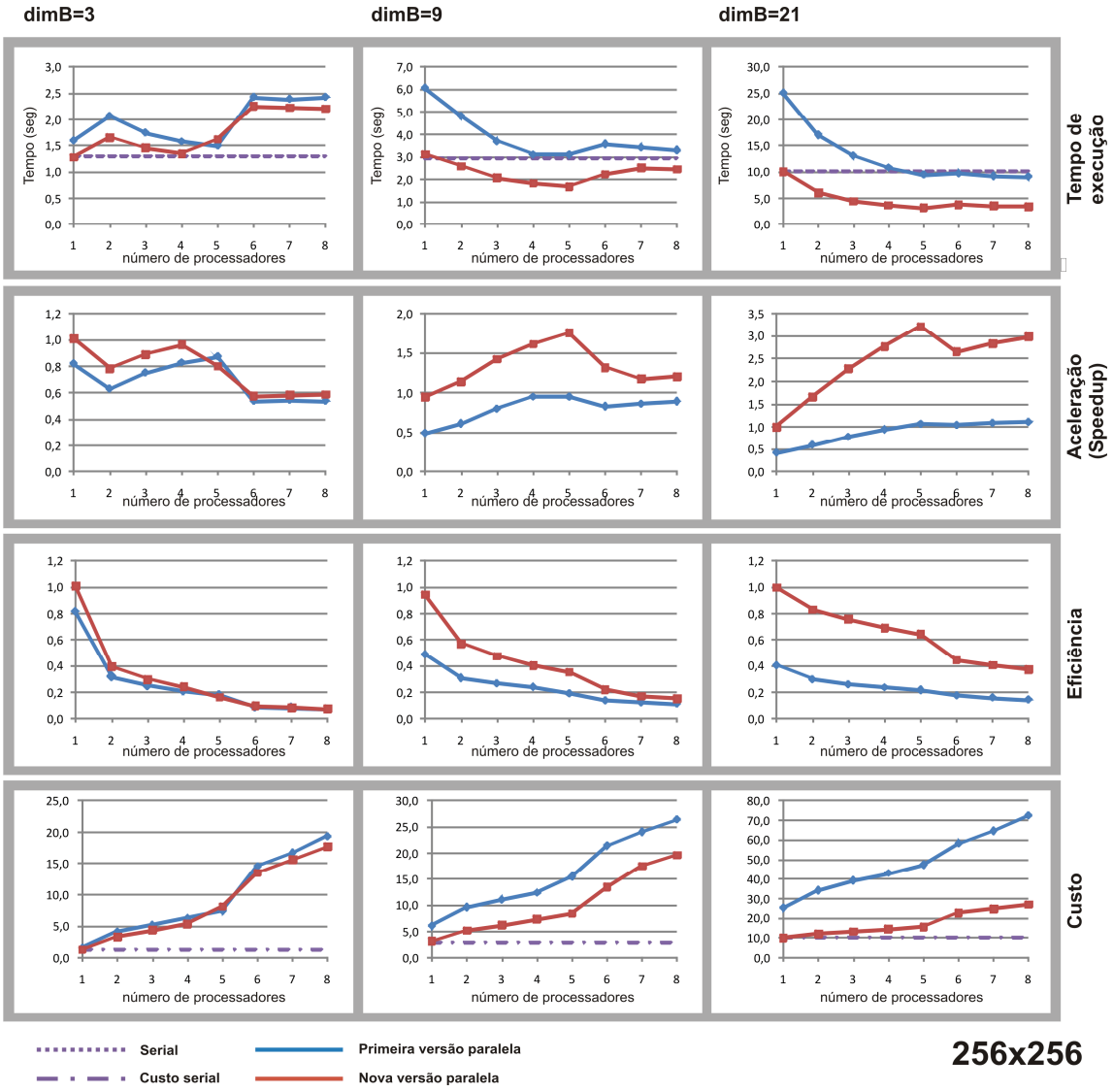


Figura 8.2 - Gráficos de medidas de desempenho obtidas com restaurações de uma imagem de 256x256 pixels (Caso II).

Nesse estudo de caso, observa-se que à medida que o esforço computacional aumenta, resultante do aumento de $\dim B$, a nova versão do programa paralelo apresenta desempenhos cada vez melhores em relação às demais implementações.

8.2.3 Estudo de caso III – Imagem de 512x512

Na Fig. 8.3 são mostrados os gráficos das medidas de desempenho dos programas de restauração no processamento de uma imagem de 512x512 pixels.

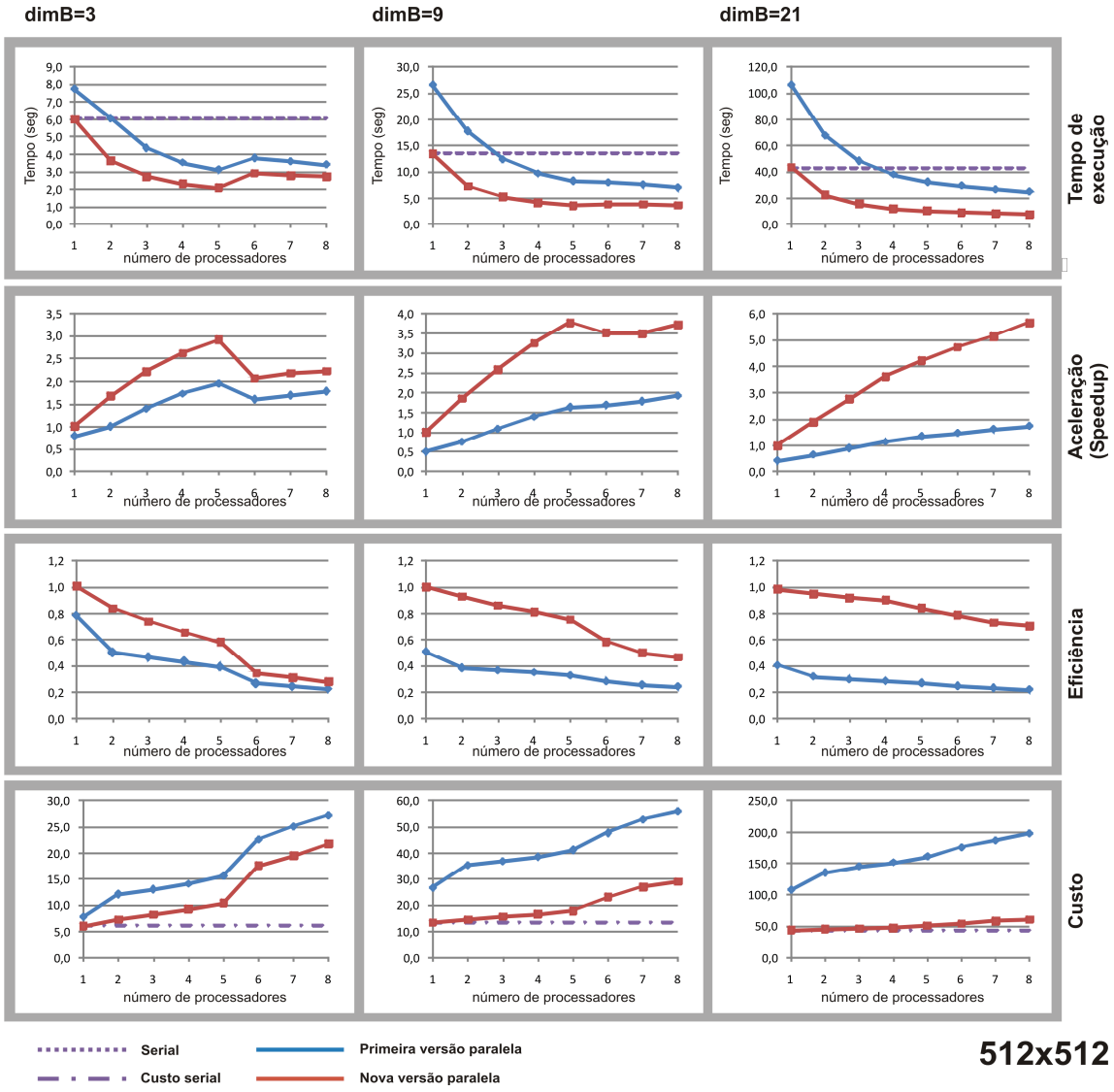


Figura 8.3 - Gráficos de medidas de desempenho obtidas com restaurações de uma imagem de 512x512 pixels (Caso III).

Nesse caso em particular, a nova versão paralela apresentou, em quase todas as situações, desempenhos melhores do que as das outras implementações do programa de restauração.

Uma queda no desempenho foi observada a partir do uso de cinco processadores (*np* >5) que diminuía à medida que o esforço computacional aumentava.

8.2.4 Estudo de caso IV – Imagem de 1024x1024

Na Fig. 8.4 são mostrados os gráficos das medidas de desempenho obtidos com os programas de restauração no processamento de uma imagem de 1024x1024 pixels.

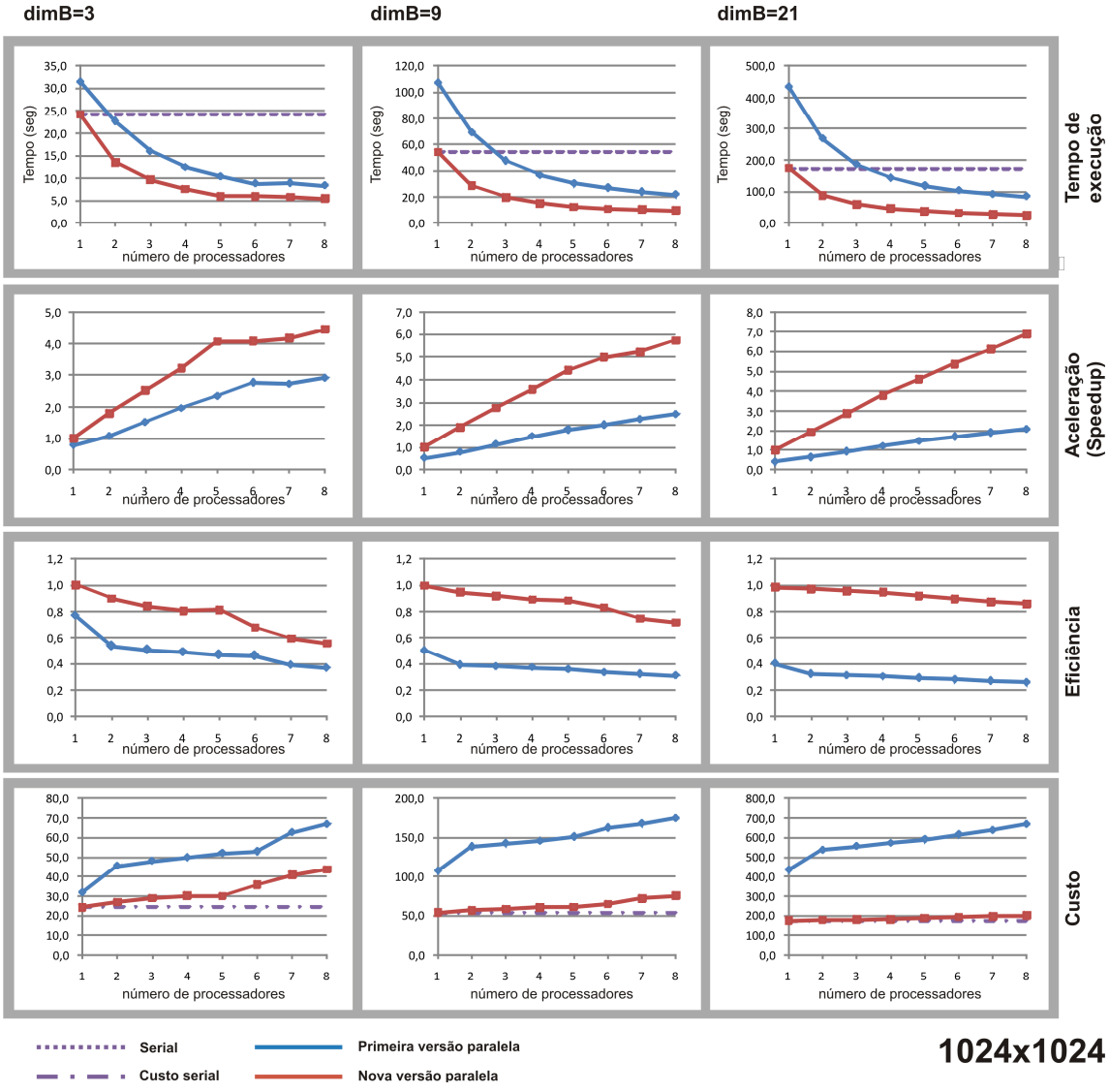


Figura 8.4 - Gráficos de medidas de desempenho obtidas com restaurações de uma imagem de 1024x1024 pixels (Caso IV).

A nova versão do programa paralelo apresentou um bom rendimento e desempenhos computacionais superiores as demais implementações nos testes em que uma imagem de dimensão 1024x1024 pixels foi restaurada.

8.2.5 Estudo de caso V – Imagem de 2048x2048

Na Fig. 8.5 são mostrados os gráficos das medidas de desempenho obtidos com os programas de restauração no processamento de uma imagem de resolução de 2048x2048 pixels (imagem grande para os tamanhos padrões de imagem de AFM).

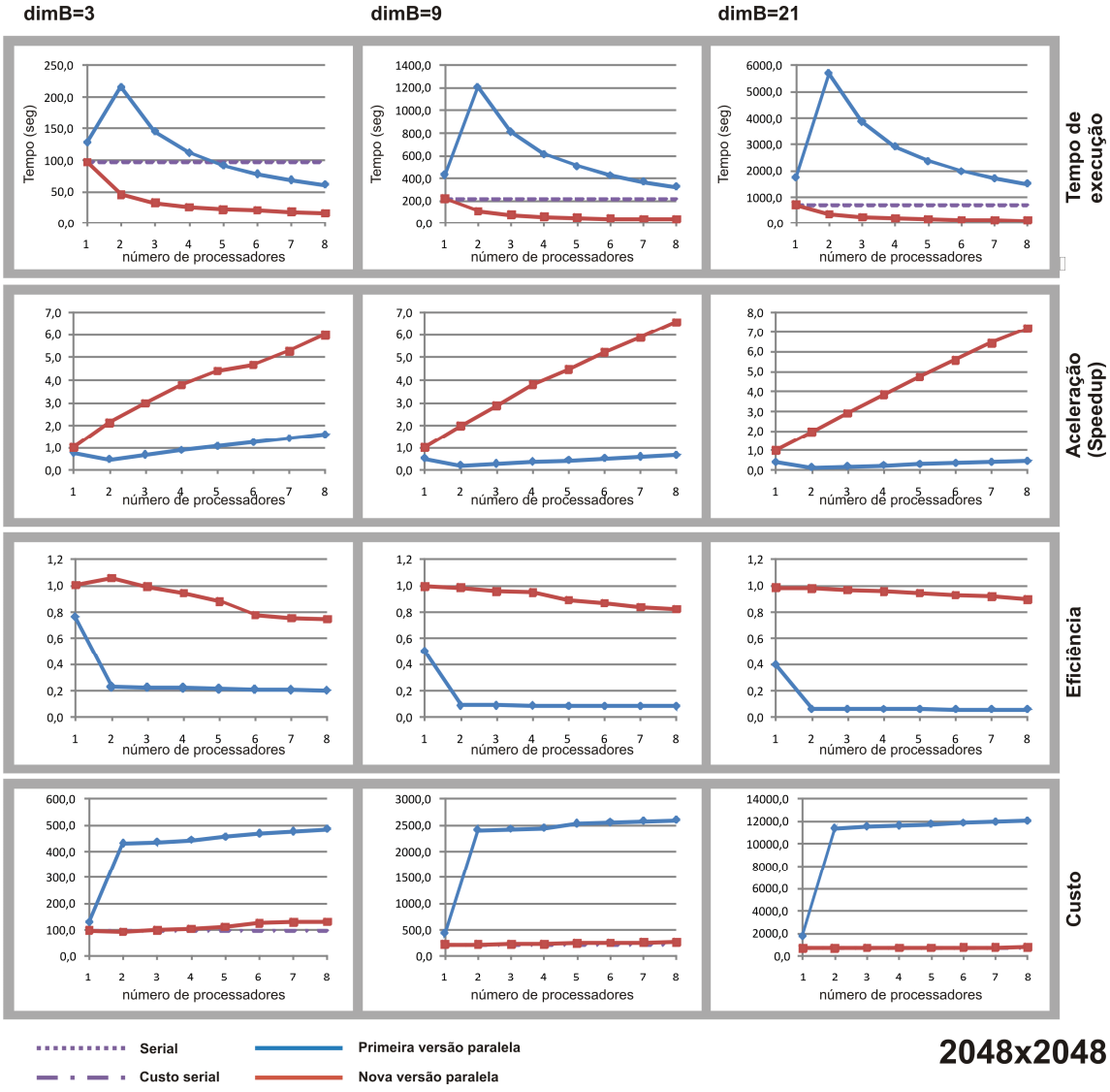


Figura 8.5 - Gráficos de medidas de desempenho obtidas com restaurações de uma imagem de 2048x2048 pixels (Caso V).

Comparando os gráficos das medidas de desempenho dos casos I a V (Figs. 8.1 a 8.5), observa-se que a nova versão paralela do programa de restauração apresenta um melhor rendimento no processamento de imagens com resolução maiores ou quando o processo de restauração demanda um esforço computacional maior como, por exemplo, quando se aumenta a dimensão do operador de borramento $\dim B$. Verifica-se que à medida que o esforço computacional cresce, a nova versão paralela do algoritmo de restauração torna-se mais eficiente quando comparado com as outras implementações.

8.3 Análise de qualidade

8.3.1 Estudo de caso VI – Avaliação de qualidade das restaurações de imagens-padrão modificadas, empregando-se uma matriz de borramento de dimensão $\dim B = 3$*, variando-se* σ^2*={20, 40} e SNR={20, 30, 40}.*

Na Fig. 8.6 é mostrado um conjunto de imagens-testes modificadas da imagem-padrão (Fig. 5.4) usando uma matriz de borramento de dimensão $\dim B = 3$. Junto à essas imagens estão as restaurações obtidas com os programas paralelos de restauração: primeira versão e a nova versão paralelas.

As imagens foram restauradas usando os parâmetros de restauração mostrados na Tabela 8.1 e as medidas de qualidade das restaurações são apresentadas na Tabela 8.2.

Tabela 8.1 – Parâmetros de restauração empregados no estudo de caso VI.

Imagem	B	σ^2	α	q	γ	ω	Realimentação
D3v20s20	3	20	0.07427	1	0,1	1	Sim
D3v20s30	3	20	0.04284	1	0,1	1	Sim
D3v20s40	3	20	0.03002	1	0,1	1	Sim
D3v40s20	3	40	0.07408	1	0,1	1	Sim
D3v40s30	3	40	0.04278	1	0,1	1	Sim
D3v40s40	3	40	0.03000	1	0,1	1	Sim

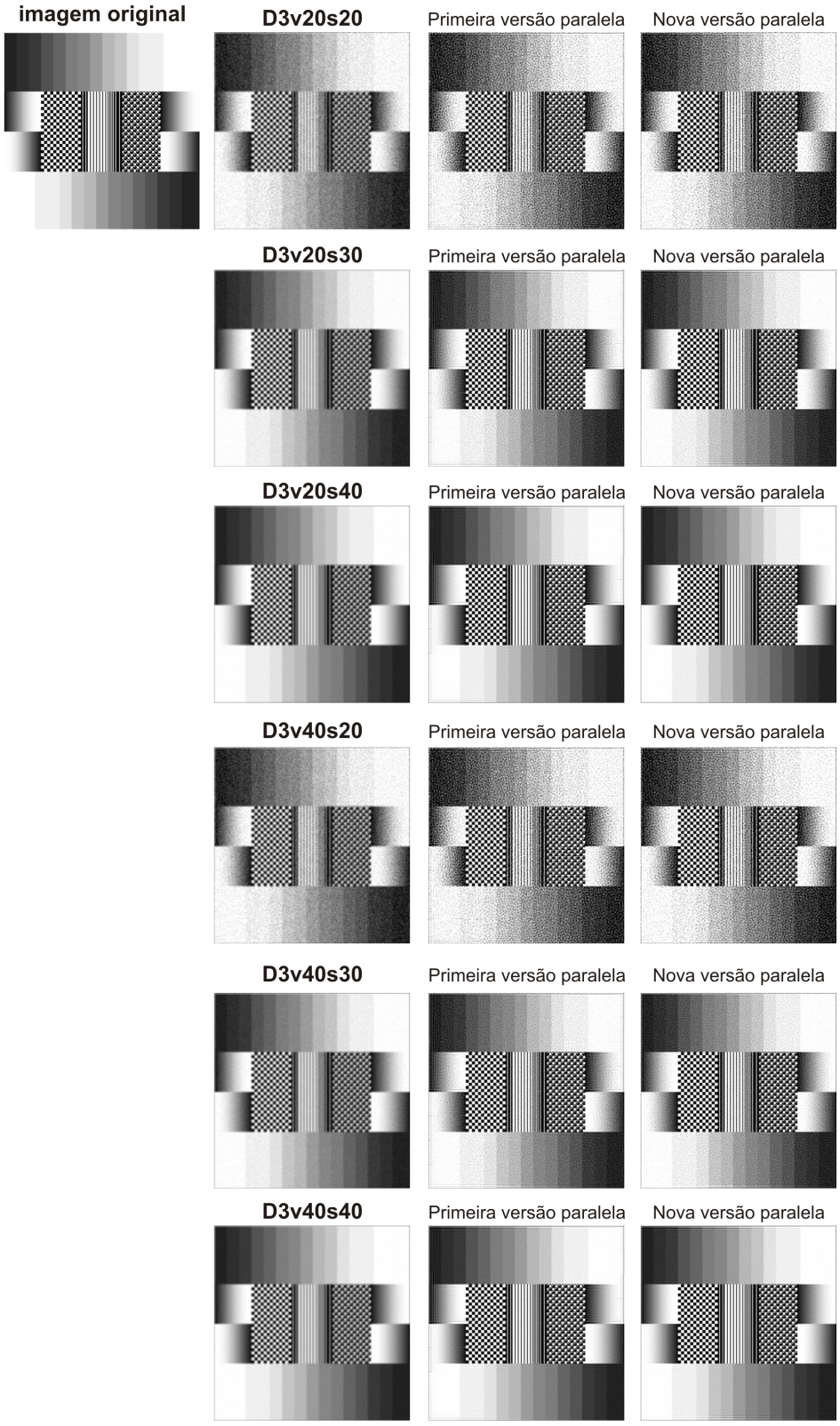


Figura 8.6 - Imagens-padrão modificadas e restaurações destas feitas com os programas paralelo anterior e nova versão paralela (Estudo de caso VI).

Tabela 8.2 –Medidas de qualidade de restaurações feitas com os programas: paralelo anterior e nova versão paralela, em um conjunto de imagens-padrão modificadas (a-f) - Estudo de caso VI.

Métricas	Padrão modificado D3v20s20	Paralela anterior	Nova versão paralela
MAE	32,4759	32,4149	28,9123
MSE	2835,15	2147,08	1753,96
NME	0,214092	0,21369	0,1906
NRMSE	7770,52	5884,66	4807,21
%MSE	N/A	75,7%	61,9%
SNR	10,4895	11,6968	12,5751
PSNR	13,605	14,8123	15,6906
IWMSE/G	31509,1	23840,1	19773,2
IWMSE/S	6331,41	4436,33	3604,84
IWMSE/V	7930,31	4043,84	3347,72
Dif%	91,74	87,87	87,23

(a)

Métricas	Padrão modificado D3v20s30	Paralela anterior	Nova versão paralela
MAE	26,33	16,4883	15,2305
MSE	2603,81	760,201	671,576
NME	0,173577	0,108697	0,100404
NRMSE	7136,46	2083,54	1840,64
%MSE	N/A	26,8%	23,7%
SNR	10,8592	16,206	16,7443
PSNR	13,9747	19,3215	19,8599
IWMSE/G	27728	8787,41	7815,78
IWMSE/S	5846,66	1641,48	1403,16
IWMSE/V	7538,34	1661,68	1460,57
Dif%	88,66	86,54	85,26

(b)

Métricas	Padrão modificado D3v20s40	Paralela anterior	Nova versão paralela
MAE	24,6324	10,715	10,5756
MSE	2579,5	561,034	513,556
NME	0,162385	0,0706372	0,0697177
NRMSE	7069,84	1537,67	1407,54
%MSE	N/A	19,8%	18,1%
SNR	10,8999	17,5254	17,9094
PSNR	14,0154	20,6409	21,0249
IWMSE/G	24917,7	6794,6	6144,34
IWMSE/S	5864,61	1264,42	1095,48
IWMSE/V	7495,55	1313,1	1180,52
Dif%	82,14	81,76	81,27

(c)

Métricas	Padrão modificado D3v40s20	Paralela anterior	Nova versão paralela
MAE	32,4677	32,3801	28,8573
MSE	2838,12	2144,74	1748,02
NME	0,214038	0,21346	0,190237
NRMSE	7778,66	5878,25	4790,93
%MSE	N/A	75,6%	61,6%
SNR	10,4849	11,7015	12,5898
PSNR	13,6005	14,8171	15,7053
IWMSE/G	31511,8	23809,5	19678,8
IWMSE/S	6338,55	4426,25	3589,89
IWMSE/V	7939,88	4030,72	3329,98
Dif%	91,54	87,85	87,24

(d)

Métricas	Padrão modificado D3v40s30	Paralela anterior	Nova versão paralela
MAE	26,3261	16,3718	15,1484
MSE	2607,54	752,249	662,872
NME	0,173551	0,107929	0,0998632
NRMSE	7146,68	2061,75	1816,78
%MSE	N/A	26,5%	23,4%
SNR	10,853	16,2516	16,801
PSNR	13,9685	19,3672	19,9165
IWMSE/G	27651,3	8679,39	7716,44
IWMSE/S	5858,15	1621,31	1383,45
IWMSE/V	7548,82	1638,07	1438,83
Dif%	88,75	86,54	85,32

(e)

Métricas	Padrão modificado D3v40s40	Paralela anterior	Nova versão paralela
MAE	24,5783	10,5952	10,4687
MSE	2583,26	554,95	505,032
NME	0,162029	0,0698475	0,0690131
NRMSE	7080,13	1520,99	1384,18
%MSE	N/A	19,6%	17,8%
SNR	10,8936	17,5727	17,9821
PSNR	14,0091	20,6883	21,0976
IWMSE/G	24819,2	6715,08	6038,04
IWMSE/S	5873,42	1247,99	1072,59
IWMSE/V	7506,52	1296,13	1158
Dif%	81,23	81,25	81,24

(f)

N/A = não se aplica. %Dif. = Percentual de pontos diferentes em relação a imagem original.

Neste estudo de caso, conforme pode ser observado na Tabela 8.2, as restaurações obtidas com a nova versão paralela do programa de restauração apresentam resultados qualitativos melhores do que da versão paralela anterior. Isto se deve em parte ao método iterativo de Jacobi aplicado na nova versão que produz resultados melhores que do método de que o Gauss-Seidel empregado na anterior, conforme o que já foi verificado no item 7.41 ("Avaliação de resultados do processo de restauração empregando Jacobi"), mais o fato da nova versão paralela não introduzir artefatos (efeito-borda) nas imagens restauradas, que ocorrem nas restaurações feitas com a primeira versão paralela.

8.3.2 Estudo de caso VII – Avaliação de qualidade de restaurações de modificações da imagem-padrão, empregando-se uma matriz de borramento de dimensão $\dim B$ *=5, variando-se* σ^2 *={20, 40} e SNR={20, 30, 40}.*

Na Fig. 8.7 é mostrado um conjunto de imagens-testes borradas por uma matriz de borramento de dimensão $\dim B$=5. Junto das imagens-testes aparecem as restaurações de cada uma delas, obtidas com os programas paralelos de restauração.

Os processos de restauração empregaram os mesmos parâmetros de borramento usados para se gerar as imagens-teste (borradas). Esses parâmetros são listados na Tabela 8.3 e as medidas de qualidade dessas restaurações são apresentadas na Tabela 8.4.

Tabela 8.3 – Parâmetros de restauração empregados no estudo de caso VII.

Imagem	B	σ^2	α	q	γ	ω	Realimentação
D5v20s20	5	20	0.07749	1	0,1	1	Sim
D5v20s30	5	20	0.04383	1	0,1	1	Sim
D5v20s40	5	20	0.03050	1	0,1	1	Sim
D5v40s20	5	40	0.07743	1	0,1	1	Sim
D5v40s30	5	40	0.04384	1	0,1	1	Sim
D5v40s40	5	40	0.03050	1	0,1	1	Sim

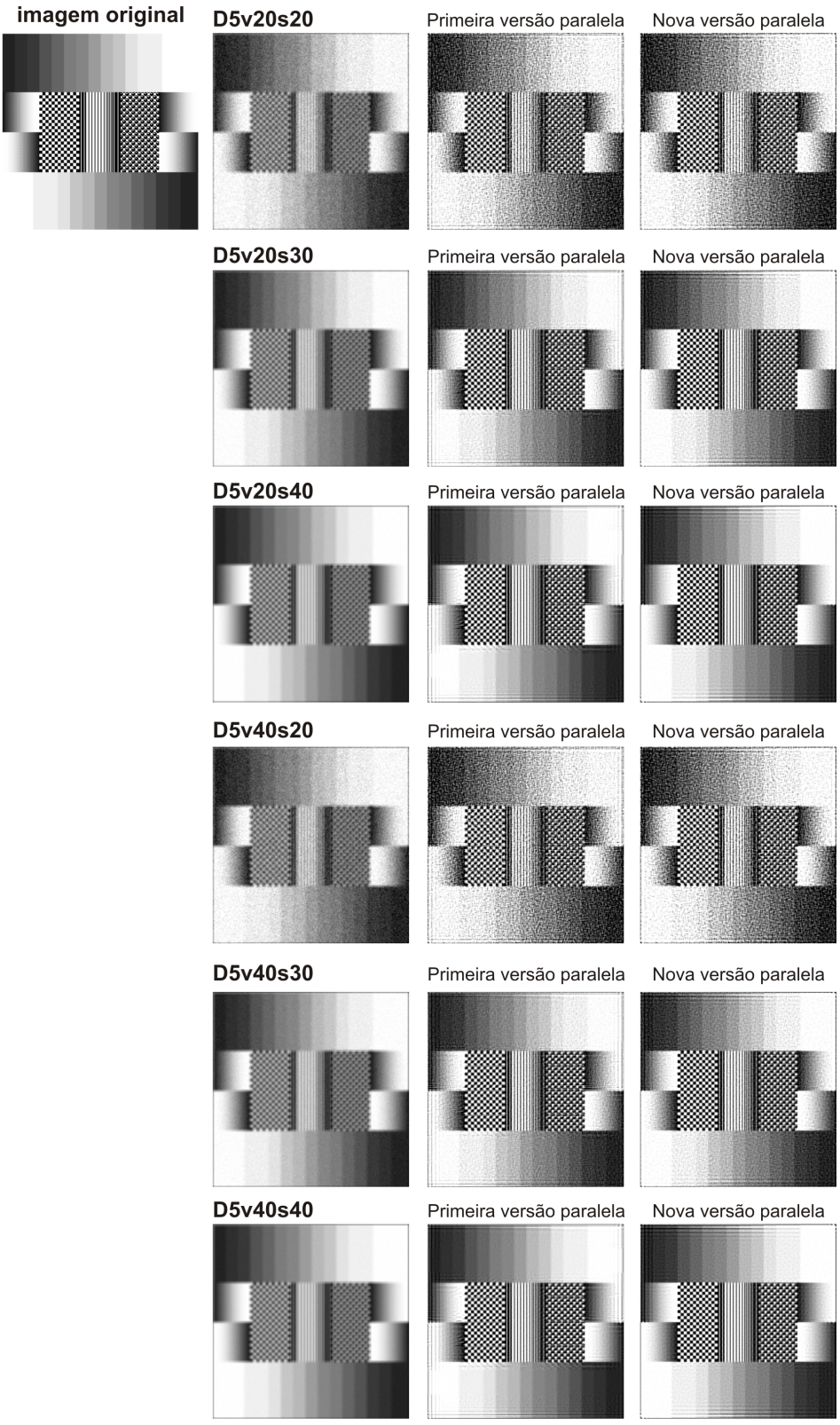


Figura 8.7 - Imagens-padrão modificadas e restaurações destas feitas com os programas paralelos anterior e nova versão (Estudo de caso VII).

Tabela 8.4 –Medidas de qualidade de restaurações feitas com os programas:primeira versão paralela e nova versão paralela, em um conjunto de imagens-padrão modificadas (a-f) – Estudo de caso VII.

Métricas	Padrão modificado D5v20s20	Paralela anterior	Nova versão paralela
MAE	37,9852	38,6889	35,9201
MSE	3600,8	3165,04	2805,08
NME	0,250412	0,255051	0,236798
NRMSE	9869	8674,66	7688,1
%MSE	N/A	87,9%	77,9%
SNR	9,45126	10,0115	10,5358
PSNR	12,5668	13,127	13,6513
IWMSE/G	39166,8	33860,4	30122,8
IWMSE/S	8143,77	6686,81	5880,12
IWMSE/V	9213,11	6635,95	5878,14
Dif%	93,21	88,24	87,24

(a)

Métricas	Padrão modificado D5v20s30	Paralela anterior	Nova versão paralela
MAE	32,3292	22,3691	21,4894
MSE	3368,36	1582,46	1386,6
NME	0,213125	0,147465	0,141666
NRMSE	9231,92	4337,17	3800,37
%MSE	N/A	43,9%	38,5%
SNR	9,74108	13,0219	13,5957
PSNR	12,8566	16,1375	16,7113
IWMSE/G	36499,9	17107,8	14914
IWMSE/S	8052,63	3451,36	2925,87
IWMSE/V	8803,62	3895,94	3341,07
Dif%	90,75	87,53	86,38

(b)

Métricas	Padrão modificado D5v20s40	Paralela anterior	Nova versão paralela
MAE	30,8329	17,5795	16,2606
MSE	3344,69	1396,49	1149,33
NME	0,203261	0,11589	0,107195
NRMSE	9167,05	3827,47	3150,06
%MSE	N/A	38,8%	31,9%
SNR	9,7717	13,5649	14,4108
PSNR	12,8872	16,6804	17,5264
IWMSE/G	35546,3	15406,7	12324,5
IWMSE/S	8030,7	3045,18	2400,6
IWMSE/V	8770,58	3587,59	2898,15
Dif%	84,65	83,46	83,84

(c)

Métricas	Padrão modificado D5v40s20	Paralela anterior	Nova versão paralela
MAE	38,1336	39,2863	36,3008
MSE	3638,78	3256	2863,05
NME	0,25139	0,258989	0,239307
NRMSE	9973,08	8923,98	7846,98
%MSE	N/A	89,5%	78,7%
SNR	9,4057	9,88841	10,447
PSNR	12,5212	13,004	13,5625
IWMSE/G	39575,7	34746,5	30871,2
IWMSE/S	8241,2	6900,76	6021,43
IWMSE/V	9276,74	6889,75	6063,15
Dif%	93,45	88,5	87,27

(d)

Métricas	Padrão modificado D5v40s30	Paralela anterior	Nova versão paralela
MAE	32,5367	22,5669	21,6493
MSE	3406,36	1580,3	1377,22
NME	0,214493	0,148769	0,142719
NRMSE	9336,06	4331,26	3774,66
%MSE	N/A	43,4%	37,8%
SNR	9,69236	13,0279	13,6252
PSNR	12,8079	16,1434	16,7408
IWMSE/G	36880,6	17139,9	14897,5
IWMSE/S	8182,2	3454,65	2913,99
IWMSE/V	8905,53	3886,65	3331,44
Dif%	90,9	87,34	86,35

(e)

Métricas	Padrão modificado D5v40s40	Paralela anterior	Nova versão paralela
MAE	31,0857	17,4986	16,2673
MSE	3382,05	1370,61	1123,37
NME	0,204927	0,115357	0,10724
NRMSE	9269,44	3756,54	3078,91
%MSE	N/A	37,7%	30,9%
SNR	9,72346	13,6461	14,51
PSNR	12,839	16,7617	17,6256
IWMSE/G	35209,8	15254,2	12129,9
IWMSE/S	8185,9	3011,4	2355,5
IWMSE/V	8864,51	3512,65	2846,51
Dif%	85,48	83,86	84

(f)

N/A = não se aplica. %Dif. = Percentual de pontos diferentes em relação a imagem original.

Nesse estudo de caso foram observados resultados semelhantes aos descritos no caso anterior (VI) e ambos apontam a nova versão paralela do programa de restauração como sendo a aplicação que apresenta melhores resultados qualitativos quando comparado com a implementação paralela anterior.

8.3.3 Estudo de caso VIII – Avaliação de qualidade de restaurações de uma imagem borrada de um texto, empregando uma matriz de borramento de dimensão dim B *=5, variando-se* σ^2 *={20, 40} e SNR={20, 30, 40}.*

Na Fig. 8.8 é mostrado um conjunto de imagens-testes borradas de uma imagem extraída de um trecho de um livro, usando uma matriz de borramento de dimensão dim B=5. Ao lado das imagens-testes, encontram-se as restaurações de cada uma delas, obtidas com as duas implementações paralelas do algoritmo de restauração: nova versão e anterior.

Nos processos de restauração foram empregaram os mesmos parâmetros usados para se fazer os borramento das imagens-teste. Esses parâmetros encontram-se listados na Tabela 8.5 e as medidas de qualidade alcançadas pelas restaurações são apresentadas na Tabela 8.6.

Tabela 8.5– Parâmetros de restauração empregados no estudo de caso VIII.

Imagem	B	σ^2	α	q	γ	ω	**Realimentação**
D5v20s20	5	20	0.15826	1	0,1	1	Sim
D5v20s30	5	20	0.06220	1	0,1	1	Sim
D5v20s40	5	20	0.03925	1	0,1	1	Sim
D5v40s20	5	40	0.16048	1	0,1	1	Sim
D5v40s30	5	40	0.06297	1	0,1	1	Sim
D5v40s40	5	40	0.03987	1	0,1	1	Sim

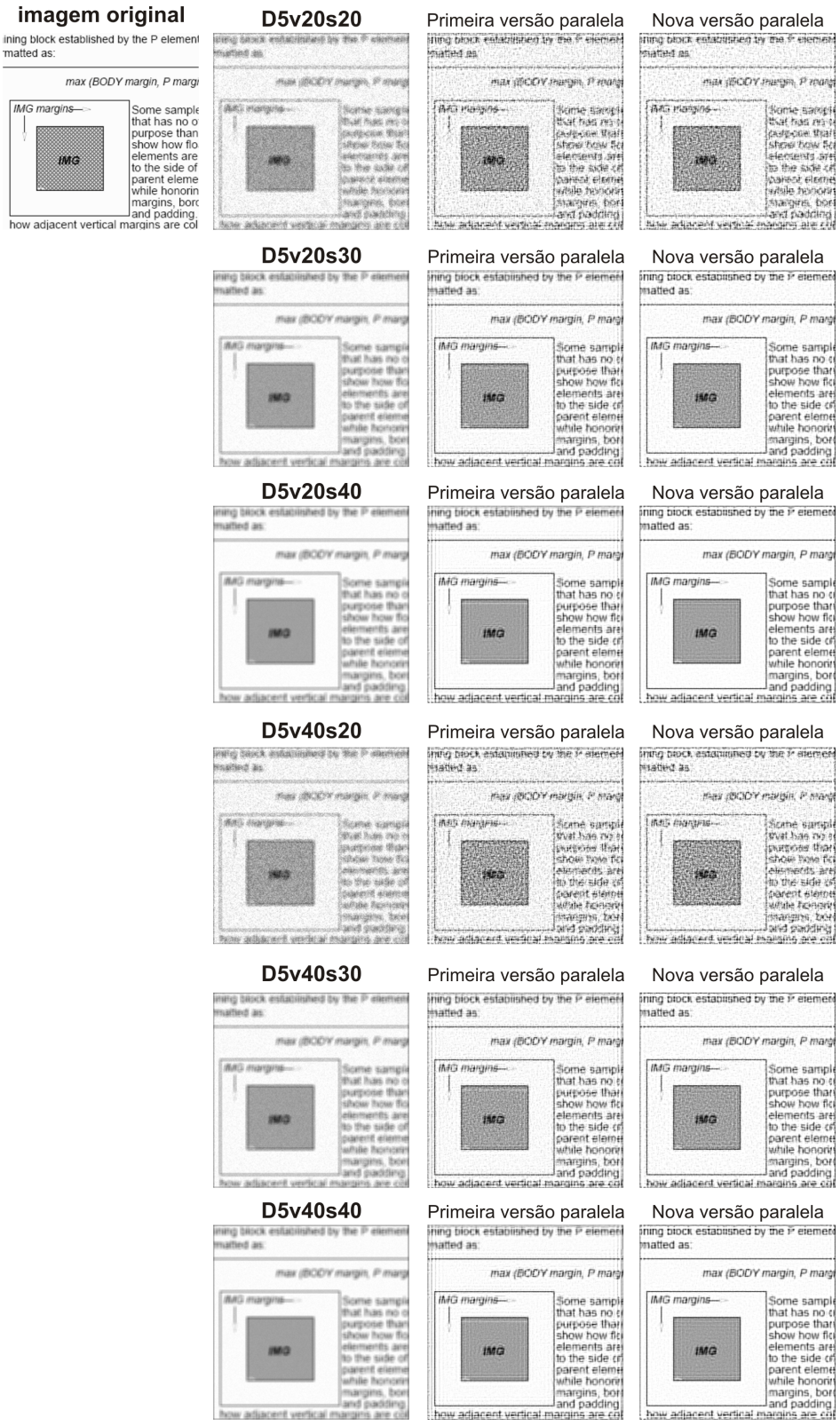


Figura 8.8 - Imagem de um texto e restaurações feitas com os programas paralelos anterior e nova versão (Estudo de caso VIII).

Tabela 8.6 –Medidas de qualidade de restaurações feitas com os programas: primeira versão paralela e nova versão paralela, em um conjunto de imagens borradas de um texto (a-f) – Estudo de caso VIII.

Métricas	Texto modificado D5v20s20	Paralela anterior	Nova versão paralela
MAE	40,0731	35,7066	33,364
MSE	3917,44	3448,38	3214,23
NME	0,182225	0,16237	0,151717
NRMSE	14472,1	12739,3	11874,2
%MSE	N/A	88,0%	82,0%
SNR	11,3259	11,8798	12,1852
PSNR	12,2008	12,7546	13,06
IWMSE/G	44688,6	43320,5	41003,1
IWMSE/S	7512,8	6812,72	6274,1
IWMSE/V	8977,55	7652,8	7188,69
Dif%	98,82	99,22	99,23

(a)

Métricas	Texto modificado D5v20s30	Paralela anterior	Nova versão paralela
MAE	36,4014	26,7723	23,881
MSE	3581,41	2185,72	1945
NME	0,165529	0,121742	0,108594
NRMSE	13230,7	8074,66	7185,36
%MSE	N/A	55,8%	49,6%
SNR	11,7154	13,86	14,3668
PSNR	12,5903	14,7349	15,2416
IWMSE/G	40126,6	28924,6	26233,6
IWMSE/S	7208,42	4180,28	3657,62
IWMSE/V	8131,59	4803,47	4239,19
Dif%	98,07	98,73	98,77

(b)

Métricas	Texto modificado D5v20s40	Paralela anterior	Nova versão paralela
MAE	35,4424	24,8409	21,1502
MSE	3546,64	1998,32	1701,3
NME	0,161168	0,11296	0,0961769
NRMSE	13102,3	7382,35	6285,07
%MSE	N/A	56,3%	48,0%
SNR	11,7578	14,2493	14,9482
PSNR	12,6326	15,1242	15,823
IWMSE/G	39263,9	26579,5	23195,6
IWMSE/S	7128,59	3768,95	3268,73
IWMSE/V	8046	4441,16	3700,24
Dif%	93,37	98,09	97,85

(c)

Métricas	Texto modificado D5v40s20	Paralela anterior	Nova versão paralela
MAE	40,5419	35,786	33,3122
MSE	3975,25	3438,87	3182,72
NME	0,184357	0,162731	0,151481
NRMSE	14685,7	12704,1	11757,9
%MSE	N/A	97,0%	89,7%
SNR	11,2623	11,8918	12,228
PSNR	12,1372	12,7666	13,1028
IWMSE/G	45255,3	43208	40601
IWMSE/S	7609,88	6818,46	6244,49
IWMSE/V	9111,99	7366,81	6791,7
Dif%	98,85	99,23	99,24

(d)

Métricas	Texto modificado D5v40s30	Paralela anterior	Nova versão paralela
MAE	36,7809	26,7874	23,9021
MSE	3630,72	2161,76	1912,47
NME	0,167255	0,121811	0,108691
NRMSE	13412,9	7986,15	7065,21
%MSE	N/A	59,5%	52,7%
SNR	11,656	13,9079	14,44
PSNR	12,5309	14,7827	15,3148
IWMSE/G	40709	28538,4	25732,5
IWMSE/S	7297,54	4141,77	3610,07
IWMSE/V	8259,81	4700,15	4071,75
Dif%	97,06	98,8	98,79

(e)

Métricas	Texto modificado D5v40s40	Paralela anterior	Nova versão paralela
MAE	35,6451	24,635	21,161
MSE	3593,74	1958,76	1664,89
NME	0,16209	0,112023	0,096226
NRMSE	13276,3	7236,19	6150,55
%MSE	N/A	53,9%	45,9%
SNR	11,7005	14,3362	15,0421
PSNR	12,5753	15,211	15,917
IWMSE/G	39542	26079,2	22650,6
IWMSE/S	7220,95	3695,5	3201,82
IWMSE/V	8169,38	4321,27	3541,16
Dif%	92,65	97,99	97,95

(f)

Nesse estudo de caso, mais uma vez foram observados os mesmos comportamentos apresentados nos dois estudos anteriores (VI e VII). Os resultados são semelhantes àqueles obtidos anteriormente e, portanto, as observações que foram feitas nesses dois casos, aqui também, se aplicam.

Um detalhe interessante que se observa nas Tabela 8.2, Tabela 8.4 e Tabela 8.6 é que as medidas baseadas no IWMSE (subjetivos), em momento algum divergiram dos resultados obtidos com as demais medidas tradicionais (objetivos) nos três estudos de caso (VI, VII e VIII).

8.3.4 Estudo de caso IX – Verificação da eliminação do efeito-borda em restaurações de imagens de AFM feitas com a nova versão paralela do programa de restauração.

Na Fig. 8.9 são mostradas as restaurações de três imagens de AFM, obtidas com a versão modificada do programa serial de restauração implementada com Jacobi simplificado e a nova versão paralela empregando, também, o Jacobi simplificado, com oito processadores em paralelo. As semelhanças nas restaurações, produzidas pelos dois programas, não são somente visuais. O MSE obtido com as duas imagens restauradas é igual a zero. Isso significa que as restaurações produzidas pela nova versão paralela são exatamente iguais àquelas que são obtidas com a versão serial do programa de restauração, numa clara demonstração de que o problema de efeito-borda, que ocorria na primeira versão paralela (anterior), não ocorre mais na nova versão do programa de restauração.

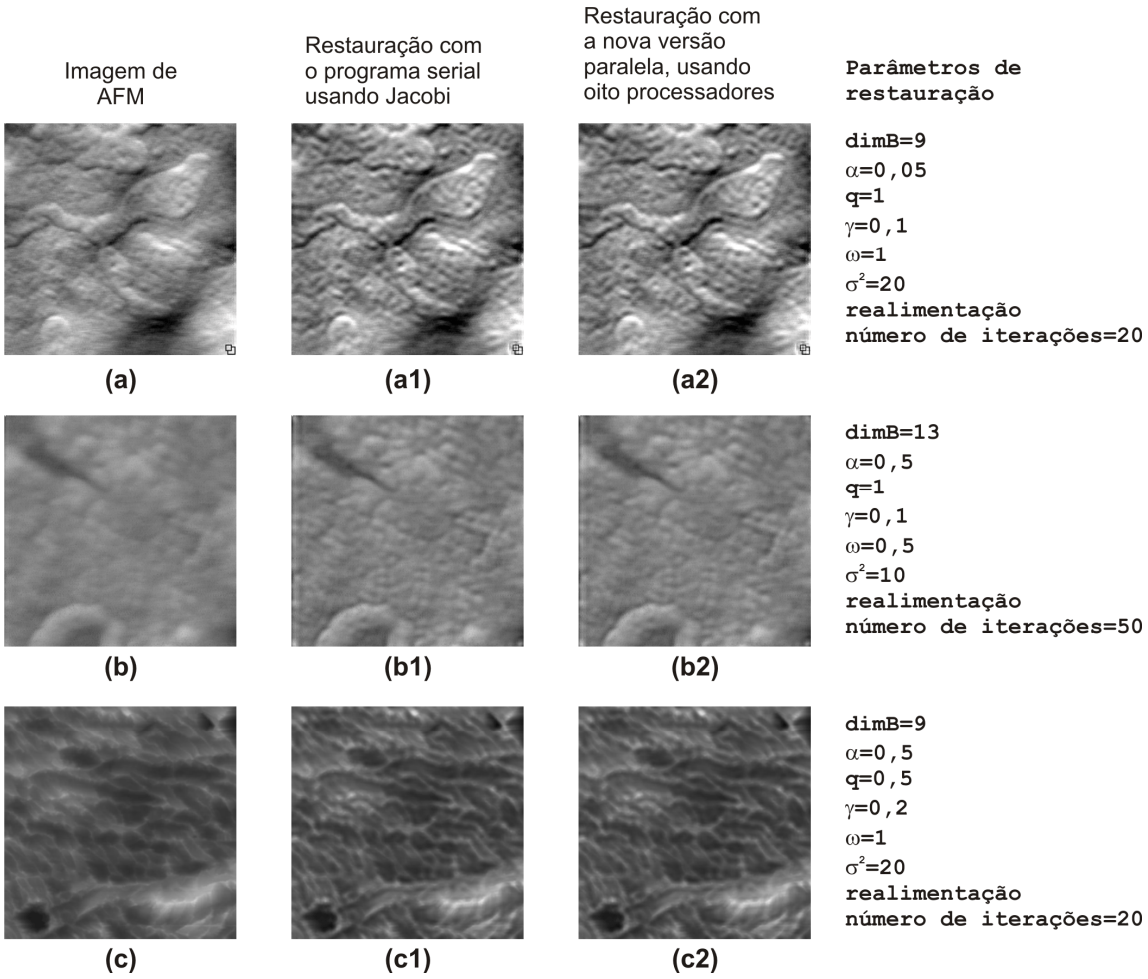


Figura 8.9 - Restaurações de imagens de AFM: (a) imagem de um eritroblasto em estado leucêmico medindo 1000nm x 1000nm e (b) 600nm x 600nm (Cidade, 2000) e (c) imagem da superfície de uma amostra de ferro submetidas a dissolução em H_2SO_4 de 30µm x 30µm (Santos, 2008), obtidas com os programas serial, empregando Jacobi simplificado (a1-c1), e nova versão paralela, usando oito processadores (a2-c2).

9 CONCLUSÕES E TRABALHOS FUTUROS

9.1 Conclusões

A nova versão paralela do algoritmo de restauração apresentou resultados muito bons. À medida que o processo de otimização e refinamentos iam sendo incorporados à nova estratégia de implementação, os tempos computacionais obtidos com a nova versão do programa paralelo diminuíam na mesma proporção até alcançarem níveis mais baixos e os desempenhos tornavam-se maiores do que das outras implementações anteriores do mesmo algoritmo.

Nos testes realizados, observou-se que a solução adotada para se implementar a nova versão paralela, além de atingir um dos objetivos desta tese que seria reduzir os tempos de execução em paralelo do programa de restauração (objetivo principal), resolveu também alguns dos problemas que haviam sido detectados na estratégia anterior. São eles:

a) Redução das sobrecargas da primeira versão paralela – a forma como a sequência de cálculos do algoritmo de restauração é executada em paralelo na nova estratégia eliminou as sobrecargas intrínsecas da abordagem paralela anterior. A sobrecarga introduzida pela atualização pelas médias ponto a ponto das áreas de sobreposição feitas em *broadcast* na versão paralela anterior do programa de restauração foi eliminada, visto que a nova versão não precisa mais fazer isso. Essa modificação não foi totalmente sem perdas. Em troca disso, a nova estratégia precisou acrescentar uma comunicação de dados interprocessadores para atualizações em algumas partições de dados que não existiam anteriormente na primeira versão paralela. De modo geral, os overheads não foram eliminados, eles apenas foram reduzidos com essa troca;
b) Eliminação do efeito-borda próximo das linhas de partição, observados nas restaurações feitas com a primeira versão do programa paralelo de restauração – modificações na estratégia de implementação em paralelo adotados na nova estratégia eliminou por completo o efeito-borda que antes aparecia nas restaurações feitas com a primeira versão paralela;

c) Restauração de imagens com resoluções maiores – a redução das sobrecargas na nova estratégia permitiu a restauração de imagens de maior dimensão, usando um número maior de processadores sem introduzir sobrecargas extras e desnecessárias ao processamento em paralelo, abrindo possibilidades de aplicação comercial da nova versão do programa paralelo de restauração. Um outro aspecto importante que pôde ser notado, é que a nova versão do programa paralelo apresenta níveis de desempenho maiores à medida que os tamanhos das imagens restauradas aumentam. Essa característica é importante, uma vez que à medida que os equipamentos eletrônicos vão evoluindo e aumentam as suas capacidades, as imagens obtidas por esses equipamentos crescem em resolução e tamanho, assim como o esforço computacional para analisá-las e restaurá-las.

Para viabilizar o uso da nova estratégia foi preciso melhorar o seu desempenho, solucionando o problema do conflito de dados causado pelo método iterativo de Gauss-Seidel usado no algoritmo de restauração. Como solução do problema modificou-se o algoritmo de restauração original, empregando-se o método iterativo de Jacobi no lugar do Gauss-Seidel. Apesar de ser um procedimento padrão em computação paralela, descobriu-se que o método iterativo de Jacobi, quando aplicado à técnica de restauração, produz resultados qualitativamente melhores que àqueles obtidos com método de Gauss-Seidel. Essa modificação na técnica, além de melhorar a velocidade de processamento na nova versão, reduziu os conflitos de dados que ocorriam nas abordagens anteriores que empregam o método de Gauss-Seidel, sem causar prejuízos na qualidade das restaurações.

Normalmente, os métodos convencionais (MSE, NME, NRMSE etc.) empregados na avaliação da qualidade de imagens têm como base critérios objetivos, obtidos essencialmente por meio de métricas apoiadas em erros estatísticos. Entretanto, os resultados obtidos com esses métodos (objetivos) diferem bastante de critérios subjetivos que tenham como base características do sistema visual humano, por exemplo. Atualmente, muitos trabalhos e pesquisas científicas em diversas áreas têm dado uma maior ênfase aos métodos que se baseiam no sistema visual humano. Entretanto, apesar da complexidade desse sistema, à medida que os modelos visuais vão sendo introduzidos nos métodos de avaliação objetivos, o relacionamento entre eles vai se aperfeiçoando e os resultados de avaliação tendem a melhorar, alcançando uma melhor correlação com a resposta dos observadores humanos.

Recentemente, vários trabalhos em processamento de imagens, visão computacional e compressão de imagens têm aceitado as novas abordagens de avaliação, baseadas na percepção/visão humana, como sendo superiores às abordagens convencionais. Baseado nessa idéia é que foi desenvolvido uma nova métrica, o IWMSE (*Information Weighted Mean Square Error*), que fornece uma medida de comparação de imagens baseados nas características do sistema visual humano, que corresponde a um critério subjetivo de comparação e expressa a qualidade visual percebida. Apesar de não ser o objeto principal abordado nesta tese, o IWMSE aparece como uma contribuição ao trabalho e foi empregado aqui como um método alternativo ao MSE na avaliação qualitativa dos resultados da restauração de imagens.

O que se pôde observar em todos os resultados obtidos com as restaurações que foram feitas empregando o IWMSE, é que em momento algum o valor obtido com essa métrica diferiu daqueles obtidos com as métricas tradicionais (objetivas), a exceção do exemplo-teste apresentado no item 5.1.4, onde o MSE falha. Pelo que foi observado neste trabalho, o uso do IWMSE mostrou-se eficaz e promissor. Entretanto, como o IWMSE é um método novo, é preciso maiores pesquisas a respeito e uma bateria maior de testes em diferentes situações e aplicações de modo a validá-lo ainda mais.

9.2 Trabalhos futuros

- ***Desenvolvimento de um método heurístico que faça um balanceamento de cargas e um despacho inteligente de lotes de imagens para restauração***

Visto que diversas arquiteturas de processamento em paralelo podem ser empregadas e que os níveis de desempenho variam de acordo com a configuração adotada, faz-se necessário a criação de um método heurístico que avalie a arquitetura empregada, o tamanho e a quantidade de imagens a ser restaurada etc. e determine a melhor forma como essas imagens serão despachadas (uma seguida da outra, de dois em dois etc.), o número e a relação de processadores que deverão ser alocados para o processamento paralelo de cada uma dessas imagens, o tamanho das partições etc., de modo a maximizar o uso dos equipamentos e explorar a arquitetura disponível, buscando alcançar um melhor desempenho e produzir resultados mais rápidos.

- ***Restauração em paralelo empregando particionamento 2D***

A partição 1D (bloco-linha ou bloco-coluna) proporciona um melhor desempenho quando a razão entre a quantidade de dados transmitida e o número de processadores é pequeno. Porém, conforme as imagens vão aumentando de tamanho e o número de processadores envolvidos no processamento em paralelo aumenta, esse particionamento 1D deixa de ser aplicado em razão dos *overheads* desse tipo de particionamento e em seu lugar deve-se empregar uma partição 2D (tabuleiro). Como trabalho futuro, pode-se modificar a técnica de implementação em paralelo do algoritmo de restauração descrita nesta tese, de modo a adaptá-la à partição 2D.

- ***Adaptação do método iterativo de busca do α ótimo no programa paralelo de restauração***

Visto não ter havido tempo hábil para a implementação do método iterativo de busca do α ótimo no programa paralelo de restauração, fica para trabalho futuro a adaptação desse método no processamento paralelo e uma modificação na técnica substituindo o método iterativo de Gauss-Seidel pelo método iterativo de Jacobi simplificado.

- ***Grid-computing***

A idéia de se desenvolver um programa de restauração em Grid, com acesso via Internet, foi ventilada ainda no início do doutorado, porém, não foi possível implementá-la. A sua aplicação se justifica ao se restaurar vídeos, onde existe uma quantidade muito grande de imagens para serem processadas. Só para se ter uma ideia, em um minuto de vídeo de 60 fps (frames por segundo), por exemplo, existem 3600 imagens para serem restauradas.

- ***Estação de trabalho cliente-servidor para restaurações de imagens***

Há algum tempo já se fala sobre a possibilidade de criação de um ambiente cliente-servidor para o processamento e restauração de imagens de AFM. Estações-clientes, através de uma interface gráfica de trabalho com várias ferramentas para processamento local de imagens, poderiam fazer uso de um canal de comunicação qualquer (rede local, Internet etc.)

e enviar requisições para uma servidora dedicada, rodando o programa de restauração em paralelo, e, através dela, fazer restaurações em um tempo de processamento mais rápido.

- ***Restauração de imagens astronômicas usando o funcional de Tikhonov***

No artigo do Arzner e Magun (1997) é descrita a aplicação de uma deconvolução, usando máxima entropia, para a restauração de imagens astronômicas e rádio-astronômicas. Observa-se que os resultados apresentados nesse artigo possuem uma semelhança muito grande com os resultados obtidos com o processo de restauração de imagens de AFM, descrito por Cidade et al. (2000).

Apesar das escalas empregadas serem extremamente diferentes (uma nanoscópica e a outra macroscópica), a aplicação do processo de restauração, descrito neste trabalho, não se limita apenas à restauração de imagens de AFM. O processo, também, pode ser aplicado à restauração de imagens com escala astronômica, apesar, do estudo de aplicação desse método ainda não ter sido feito. Como trabalho futuro, aqui se sugere que se faça um estudo mais detalhado dessa aplicação.

- ***IWMSE+***

Atualmente, muitos trabalhos em processamento de imagens (p.ex. compressão de imagens etc.) têm demonstrado um certo interesse em estudar e aplicar algumas abordagens em que a medida da diferença tenha como base o nível de interesse visual humano.

O uso do IWMSE tem-se mostrado promissor e demanda mais investigações para desenvolvê-lo e validá-lo. Atualmente, as medidas de interesse local (intensidade, brilho, borda etc.) são utilizadas separadamente no IWMSE. Entretanto, apesar de seletivo, o sistema visual humano não se detém a uma medida de interesse de cada vez, ele observa as diferenças como um todo, com maior ou menor destaque para uma medida de energia em relação as outras. Um trabalho que já sendo iniciado visa exatamente tratar o IWMSE sob esse novo enfoque.

Paralelamente, estuda-se ainda a criação de uma nova métrica, com base no IWMSE, que leve em consideração, também, o foco de atenção e a visão lateral, sendo que este trabalho encontra-se ainda em um estágio inicial de desenvolvimento.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, A.C.; STUTZ, D.; CIDADE, G.A.G.; SILVA NETO, A.J. Uso de Graphics Processing Unit (GPU) na Restauração de Imagens de Microscopia de Força Atômica com Regularização de Tikhonov. In: Congresso Nacional de Matemática Aplicada e Computacional (CNMAC), 32, Set. 2009, Cuiabá: Anais... Cuiabá, 2009.

ALMEIDA, F.; DELGADO, C.; LÓPEZ, R.J.G; SANDE, F. A public parallel method for astronomical image restoration. In: Workshop on State-of-the-art in Scientific and Parallel Computing, Umea, Sweden, June 18-21, 2006. , Proceedings...

ANDREWS, H.C.; HUNT, B. R. Digital Image Restoration. Prentice-Hall. 1977.

ARZNER, K.; MAGUN, A. Fast Maximum Entropy Restoratio of Low-noise Solar Images. Astron. Astrophys, 324, p.735-742. 1997.

BANHAM, M. R.; KATSAGGELOS, A. K. Digital Image Restoration. IEEE Signal Process. Magazine. p.24-41. March 1997.

BEVILACQUA, A.; PICCOLOMINI, E.L. Parallel image restoration on parallel and distributed computers. Parallel Computing, v.26, p.495-506, March 2000.

BINNIG, G.; QUATE, E.C.F.; GERBER, C.H. Atomic Force Microscope. Phys. Rev. Lett., v.56, n.9, p.930-933, 1986.

BONNET, N.; DONGMO, S.; VAUTROT, P.; TROYON, M. A. Mathematical Morphology Approach to Image-Formation and Image-Restoration in Scanning Tunneling and Atomic-Force Microscopies. Paris: Editions Physique, 1994.

BREGMAN, L.M., The Relaxation Method of Finding the Common Point of Convex Sets and its Application to the Solution of Problems in Convex Programming. Zh. vÿchisl. Mat. mat. Fiz., v.7, n.3, p.620-631, 1967.

CARITA MONTERO, R.F.; ROBERTY, N.C.; SILVA NETO, A.J. Natural Base Construction for Absorption Coefficient Estimation in Heterogeneous Participating Media with Divergent Beams. In: International Conference on Inverse Problems in Engineering: Theory and pratice, 3, Port Ludlow, Washington, Proceedings...

CENSOR, Y.; ZENIOS, S.A. Parallel Optimization: theory, algorithms and applications. New York: Oxford University Press, 1997.

CHAN, A.; ASHTON, D.; LUSK, R.; GROPP, W. Jumpshot-4 Users Guide. University of Chicago. Mathematics and Computer Science Division, Argone National Laboratory. July, 2007.

CHAN, A.; GROPP, W.; LUSK, E. User's Guide for MPE: Extensions for MPI Programs. Mathematics and Computer Science Division, Argone National Laboratory. 1998.

CHINAGLIA, E. F. Caracterização nanoestrutual de filmes finos do grupo IV-B depositados por sputtering magnetron. 2002, Tese (doutorado), Instituto de Física, USP, São Paulo.

CIDADE, G.A.G., Desenvolvimento de metodologias de aquisição e processamento de imagens biológicas em microscopia de força atômica (AFM). 2000, Tese (doutorado), Instituto de Biofísica Carlos Chagas, Universidade Federal do Rio de Janeiro.

CIDADE, G.A.G.; ANTENEODO, C.; ROBERTY, N.C.; SILVA NETO, A.J. A Generalized Approach for Atomic Force Microscopy Image Restoration with Bregman Distances as Tikhonov Regularization Terms. Inverse Problems in Engineering, v.8, p.457-472, 2000.

CIDADE, G.A.G.; SILVA NETO, A.J.; ROBERTY, N.C. Restauração de Imagens com Aplicações em Biologia e Engenharia - Problemas Inversos em Nanociência e Nanotecnologia. São Carlos, SP: SBMAC, 2003. (Notas em Matemática Aplicada; v.1).

CIDADE, G.A.G., ROBERTY, N.C., SILVA NETO, A.J. Reconstrução de Imagens com Regularização de Tikhonov e Determinação do Parâmetro de Regularização Ótimo. In: Simpósio Interdisciplinar em Tecnologia e Saúde, 2, 1991, Rio de Janeiro, Anais... Rio de Janeiro: COPPE/UFRJ, 1998.

CSISZÁR, I. Why Least Squares and Maximum Entropy? An Axiomatic Approach to Inference for Linear Inverse Problems. The Annals of Statistics 19(4), p.2032-2066. 1991.

DAMERA-VENKATA, N.; KITE, T. D.; GEISLER, W. S.; EVANS, B. L. Image Quality Assessment Based on a Degradation Model. IEEE Trans. on Image Processing, v.9, n.4, April 2000.

DONGDONG, M.; JINZONG, L.; BING, Z.; FUZHEN, Z. Research on the Architectures of Parallel Image Processing Systems. Intelligent Information Technology Application, 2008. IITA '08. Second International Symposium on., v.3, p.146-150, Dec. 2008.

DONGMO, S., TROYON, M., VAUTROT, P., DELAIN, E., BONNET, N. Blind Restoration Method of Scanning Tunneling and Atomic Force Microscopy Images. J. Vac. Sci. Technol. B 14(2), p.1552-1556. 1996.

ESKICIOGLU, A. M. Quality Measurement for Monochrome Compressed Images in the Past 25 Years. In: IEEE International Conference on Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings... .v.4, p.1907-1910. June, 2000.

ESKICIOGLU, A. M.; FISHER, P. S. Image Quality Measures and Their Performance. IEEE Trans on Communication. v.43, p.2959–2965, Dec. 1995.

ESKICIOGLU, A. M.; FISHER, P. S. A survey of quality measures for gray scale image compression. In: Proc. 1993 Space and Earth Science Data Compression Workshop (NASA Conference Publication 3191), Snowbird, Utah, Apr. 2, 1993, p.49-61. Proceedings...

FERNÁNDEZ, G.J.; JACOBO-BERLLES, J.; BAUZÁ, M.; BORENSZTEJN, P.; MEJAIL, M. Parallelization of ARTUR algorithm using PVM. In: Proceedings of the 13th Brazilian Symposium on Computer Graphics and Image Processing, SIBGRAPHI, Gramado, Brazil, p. 331, 2000. Proceeding...

FONSECA, E. X. Implementação de Algoritmos Genéticos e Estratégias Híbridas Visando a Restauração de Imagens Obtidas em Escala Nanométrica com Microscópios de Força Atômica. Dissertação (Mestrado), UERJ. 2004.

FOSTER, I. Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering. Addison Wesley. 1994.

FURTADO, V. Avaliação do uso do funcional de Tikhonov com uma família de funções de regularização a um parâmetro para a restauração de imagens biológicas. 2002. Dissertação (mestrado), Pós-graduação em Modelagem Computacional, Instituto Politécnico de Nova Friburgo/UERJ, Nova Friburgo.

GALATSANOS, N.P.; KATSAGGELOS, A.K. Methods for Choosing the Regularizations Parameters and Estimating the Noise Variance in Image Restoration and Their Relation. IEEE Trans. on Image Process., v.1, n.3, p.322-336. July 1992.

GIL, M.C.C. Aplicação de Redes Neurais Artificiais na Restauração de Imagens Obtidas com Microscópios de Força Atômica. Dissertação (Mestrado), UERJ. 2005.

GOMES, J.; VELHO, L. Computação Gráfica: Imagens. Rio de Janeiro: IMPA: SBM. 1994. ISBN 85-244-0088-9.

GONZALEZ, R.C.; WINTZ, P. Digital Image Processing. 2. ed., Boston: Addison-Wesley, 2001.

GONZALEZ, R.C.; WOODS, R.E. Processamento de Imagens Digitais. São Paulo: Edgard Blucher, 2000.

GORDON, D. Parallel ART for image reconstruction in CT using processor arrays. The International Journal of Parallel, Emergent and Distributed Systems, v.21, n.5, p.365–380, October 2006.

GROETSH, C.W. The Theory of Tikhonov Regularization for Freedholm Equations of The First Kind. Pitman Advanced Publishing Program Marshfield. Mass. 1984.

HANSMA, P.K.; ELLINGS, V.B.; MARTI, O.; BRACKER, C.E. Scanning Tunneling Microscopy and Atomic Force Microscopy: Application to Biology and Technology. Science, v.242, p.209-242, 1988.

HUNT, B.R. The Application of Constrained Least Squares Estimation to Image Restoration by Digital Computer. IEEE Trans. Comput. v.22. n.2. p.805-812. Sept. 1973.

JENG, F.C.; WOODS, J.W. Inhomogeneous Gaussian Image Models for Estimation and Restoration. IEEE Trans. Acoust., Speech, Signal Proc., v.36, n.8, p.1305-1312, 1988.

KANG, M.G.; KATSAGGELOS, A.K. General Choice of the Regularization Functional in Regularized Image Restoration. IEEE Trans. Image Process., v.4, n.5, p.594-602, 1995.

KARAYIANNIS, N. B.; VENETSANOPOULOS, A. N. Regularization Theory in Image Restoration: The Regularizing Operator Approach. Optical Engineenring, 28(7), p.761-780, 1989.

KASAS, S.; GOTZOS, V.; CELIO, M.R. Observation of Living Cells Using the Atomic Force Microscope. Biophys. J., v.64, p.539-544, 1993.

KATSAGGELOS, A.K. Iterative Image Restoration Algorithm. Optical Engineering Special issue on Visual Comunications and Image Processing, v.28, n.7, p.735-748, July 1989.

KATSAGGELOS, A.K.; BIEMOND, J.; MERSEREAU, R.M.; SCHAFER, R.W. A General Formulation of Constrained Iterative Restoration Algorithms. IEEE Proc. ICASSP, Tampa, FL, p.700-703, March 1985.

KATSAGGELOS, A.K.; BIEMOND, J.; SCHAFER, R.W.; MERSEREAU, R.M. A Regularized Iterative Image Restoration Algorithm. IEEE Trans. Acoust., Speech, Signal Proc., v.39, p.914-929, April 1991.

KE, J.; BURTSCHER, M.; SPEIGHT, E. Reducing Communication Time through Message Prefetching. In: The 2005 International Conference on Parallel and Distributed Processing Techniques and Applications, p.557-563. June 2005.

KNUTSSON, H.E.; WILSON, R.; GRAINLAND, G.H. Anisotropic Nonstationary Image Estimation and its Applications: Part I – Restoration of Noisy Images. IEEE trans. Commun. COM-31, p.388-397. 1983.

KOKARAM, A.C.; PERSAD, N.; LASENBY, J.; FITZGERALD, W.J.; MCKINNON, A.; WELLAND, M. Restoration of Images from Scanning-Tunneling Microscope. Applied Optics, v.34, n.23, p.5121-5132, 1995.

KREYSZIG, E. Advanced Engeneering Mathematics. 3. ed. New York: Wiley International. 1972.

KUMAR, V.; GRAMA, A.; GUPTA, A.; KARYPIS, G. Introduction to Parallel Computing: Design and Analysis of Algorithm. California: Benjamin/Cummings. 1994. ISBN 0-8053-3170-0.

LI, W.; ZHANG, D.; LIU, Z.; QIAO, X. A Parallel Algorithm for Image Information Restoration. In: High Performance Computing in the Asia-Pacific Region, May 2000. v.2, p.790-793. Proceedings...

LIM, J.S. Two-Dimensional Signal and Image Processing. Prentice-Hall, 1990.

MACKWORTH, N.; MORANDI, A. The Gaze Selects Informative Details within Pictures. Perception and Psychophysics, 2:547-552, 1967.

MANNHEIMER, W.A. Microscopia dos Materiais: Uma Introdução. Rio de Janeiro: E-papers Serviços Editoriais. 2002. ISBN 85-87922-54-8.

MARIANI, T.; MUSIO, A.; FREDIANI, C.; SBRANA, I.; ASCOLI, C. An Atomic Force Microscope for Cytilogical and Histological Investigations. J. Microsc. v.176, p.121-131, 1994.

MILLER, K. Least-squares Method for Ill-posed Problems with a Prescribed Bound. SIAM J. Math. Anal. 1, 52-74. 1970.

MING, J.; SHUI-FA, S.; FANGMIN, D.; BANGJUN, L. Image Quality Evaluation Based on HVS and Energy_Weighted Subband of Wavelet Transform. IEEE Comp. Soc. p.491-495. Proceedings. Second International Symposium on Intelligent Information Technology Application. 2008. DOI 10.1109.

MPI FORUM. MPI: A Message-Passing Interface Standard. The International Journal of Supercomputer Applications and High Performance Computing, 8(3/4):165-414, 1994.

MOREIRA, J.E.; REESE, T.S.; TORGERSEN, T.C. Freeze-substituition as a Preparative Technique for Immunoelectonmicroscopy: Evaluation by Atomic Force Microscopy. Microscopy Research and Technique, v. 33, p. 251-261, 1996.

NOONAN, Joseph P.; NATARAJAN, Premkumar. A General Formulation for Iterative Restoration Methods. IEEE Transactions on Signal Processing, v.45, n.10, p.2590-2593. October 1997.

ORAINTARA, S.; KARL, W.C.; CASTANON, D.A.; NGUYEN, T.Q. A Method for Choosing the Regularization Parameter in Generalized Tihkonov Regularized Linear Inverse Problems. IEEE Image Processing. Proceedings. International Conference on. v.1. n.2000. p.93-96. 2000.

ORTEGA, J.M.; RHEINBOLDT, W.C. Iterative Solution of Nonlinear Equations in Several Variables. Academic Press, New York. 1970.

PACHECO, P.S. Parallel Programing with MPI. San Francisco: M. Kaufmann, 1997.

PAIK, J.K.; KATSAGGELOS, A.K. Parallel Iterative Image Restoration Algorithms. IEEE Circuits and Systems, 1989. Proceedings of the 32nd Midwest Symposium on, v.1. p.63-66.

PEDRINI, H.; SCHWARTZ, W.R. Análise de Imagens Digitais: Princípios, Algoritmos e Aplicações. São Paulo: Thompson, 2008.

PICCOLOMINI, E.L.; ZAMA, F. Parallel Image Restoration with Domain Decomposition. Real-Time Imaging, v.7, p.47-57, 2001.

PRESS, W.H.; TEUKOLSKY, S.A.; VETTERLING, W.T.; FLANNERY, B.P. Numerical Recipes in Fortran 77 - The Art of Scientific Computing. 2. ed. New York: Cambridge Univ. Press, v.1, 1996.

PUETER, R.C.; GOSNELL, T.R. and YAHIL, A. Digital Image Reconstruction: Deblurring and Denoising. Annu. Rev. Astrophys. v.43, p.139-194, 2005.

QUINN, M. J. Parallel Computing: Theory and Pratice. 2. ed. McGrown Hill. 1994.

RIBEIRO, S.J.M.; HILL, C.E.M.; CIDADE, G.A.G.; SOEIRO, F.J.C.P.; SILVA NETO, A.J. Algoritmos genéticos aplicados em restauração de imagens de microscopia de força atômica (AFM). In: Encontro de Modelagem Computacional, 4, Nova Friburgo, RJ, 2001, Anais...

RIBEIRO, S.J.M.; SILVA, J.C.P. *Genetic Algorithms and Biological Images Restoration: Preliminary report*. Springer-Verlag Berlin, Germany, 2002.

ROMUALDO, K.V. Proposição e Avaliação de Indicadores de Desempenho para Algoritmos de Restauração de Imagens. 2006. Dissertação (mestrado), Pós-graduação em Modelagem Computacional, Instituto Politécnico de Nova Friburgo/UERJ, Nova Friburgo.

SALTZ, J.H.; NAIK, E.V.K.; NICOL, D.M. Reduction of the Effects of the Communication Delays in Scientific Algorithms on Message Passing MIND Architectures. SIAM J. Sci. Stat. Comput., v.8, n.1, 1987.

SANTOS, A.C. Restauração de Imagens de Microscopia Óptica com o Funcional de Regularização de Tikhonov Visando a Avaliação de Superfícies Metálicas. 2008. Dissertação (mestrado), Pós-graduação em Modelagem Computacional, Instituto Politécnico de Nova Friburgo/UERJ, Nova Friburgo.

SERPRO. Glossário de Termos Técnicos em Processamento de Dados. 2. ed. v.1. Rio de Janeiro. 1976.

SCHLEMER, E. Sistemas Distribuídos. Gravataí/RS. Disponível em: http://www.ulbra.tche.br/~elgios/sisdis/sisdisCompleto2003-versao_Compativel.pdf. Acesso em: 30/05/2003.

SHANNON, C. E.. A Mathematical Theory of Communication. The Bell System Technical Journal, v.27, p.379–423, 623–656, July, October, 1948.

SILVA NETO, A.J.; MOURA NETO, F.D. Escolha de Modelos - Problemas Inversos em Engenharia. In: CNMAC, 12. Santos, 1999. Notas de aula do mini-curso técnico MC05.

SMITH, S. The Scientific and Engineer´s Guide to Digital Signal Processing. 2. ed. Califórnia Tech. Pub. 1999. Disponível em: http://www.dspguide.com/. Acesso em 15/09/2007.

SNIR, M.; OTTO, S.; HUSS-LEDERMAN, S.; WALKER, D.E.; DONGARRA, J. MPI: The Complete Reference. Cambridge: The MIT Press, 1996.

STUTZ, D. Restauração de Imagens em Escala Nanométrica com Funcional de Regularização e Tikhonov e Computação Paralela. 2004. Dissertação (mestrado), Pós-Graduação em Modelagem Computacional, Instituto Politécnico de Nova Friburgo/UERJ, Nova Friburgo.

STUTZ, D.; SILVA NETO, A.J.; CIDADE, G.A.G. Parallel Computation Approach for the Restoration of AFM Images based on the Tikhonov Regularization Method. Microscopy & Microanalysis, New York - USA, v.11, p.22-25, 2005.

STUTZ, D.; SILVA NETO, A.J.; FARIAS, R.C. Information weighted Mean Square Error (IWMSE): Uma medida de comparação de imagens baseada na percepção. In: Encontro de Modelagem Computacional, 10, Anais. Nova Friburgo, RJ. 2007.

STUTZ, D.; SILVA NETO, A.J.; FARIAS, R.C. Método Iterativo de Busca do Parâmetro de Regularização Ótimo em um Problema de Restauração de Imagens. In: Encontro de Modelagem Computacional, 11, Anais. Volta Redonda, RJ. 2008.

STUTZ, D.; SILVA NETO, A.J.; FARIAS, R.C. Processo de Busca do Parâmetro de Regularização Ótimo em um Problema de Restauração de Imagens. In: Encontro Acadêmico de Modelagem Computacional, 2, Anais. Petrópolis: LNCC. 2009.

TELATAR, Z. Adaptative Filter Design for Image Deblurring by Using Multi-criteria Blurred Image Information. Science Direct. Digital Image Processing 15 (2005) 4-18. September 2004.

TERRAÇÃO, E. F. Restauração de imagens em escala nanométrica com morfologia matemática e estratégias híbridas. Dissertação (Mestrado), UERJ. 2005.

TIKHONOV, A.N.;ARSENIN, V.Y. Solutions of Ill-Posed Problems. Wiley: New York. 1977.

TOMPA, D.; MORTON, J.; JERNIGAN, E. Perceptually Based Image Comparison. IEEE Image Processing, 2000. Proceedings... 2000 International Conference on. v.1, p.489-492, 2000.

TOPPER, T. Selection Mechanisms in Human and Machine Vision. PhD thesis, University of Waterloo, Waterloo Ontario Canada, 1991.

TOPPER, T.; JERNIGAN, M. On the Informativeness of Edges. IEEE Internat. Conf. on Systems, Man and Cybernetics, 3:909-914, 1989.

WALLIS, R. Approach to Space Variant Restoration and Enhancement of Images. In: Proc. Symp. Current Math., Problems in Image Science, Naval Postgraduate School, Monterey, Calif. 1976.

WEISMAN, A.D.; DOUGHERTY, E.R.; MIZES, H.A; MILLER, R.J.D., Nonlinear Digital Filtering of Scanning-Probe-Microscopy Images by Morphological Pseudo-Convolutions. J. Appl. Phys., v.71, n.4, p.1565-1578, 1992.

WILSON, D.L.; KUMP, K.S.; EPPELL, S.J.; MARCHANT, R.E. Morphological Restoration of Atomic-Force Microscopy Images. Washington, DC: Amer Chemical Soc, 1995.

WONG, A.; VOGEL, M. Resolution-dependent Information Mesures for Image Analysis. IEEE Trans. on Systems, Man and Cybernetics, SMC-7(1):49-61, Jan 1977.

ZERVAKIS, M.E.; KATSAGGELOS, A.K.; KWON, T.M. A Class of Robust Entropic Functionals for Image Restoration. IEEE Trans. Image Process. 4(6), p.752-773, 1995.

GLOSSÁRIO

Aceleração – Medida que captura e expressa a melhoria de desempenho de um programa quando é feita alguma modificação no hardware/software ou quando se aplica alguma técnica computacional diferente e/ou otimizada em sua implementação. *Inglês: speedup*.

AFM – Microscópio de Força Atômica. *Abrev. Atomic Force Microscope*.

Área de sobreposição – (*overlap area*) Área da partição que sobrepõe à partição vizinha

Área sobreposta – Área de uma partição que é sobreposta pela partição vizinha.

Buffer – Área de armazenamento de dados temporário.

Conflitos de controle – Momento em que há a necessidade de se tomar uma decisão com base nos resultados de uma instrução, enquanto outras instruções estão sendo executadas em paralelo. Ex.: Instrução de desvio condicional: se o computador tiver que desviar, deverá interromper as instruções do *pipeline.*

Conflito de dados – Momento em que uma instrução em paralelo fica aguardando pelo resultado de uma instrução anterior para continuar.

Custo – Medida com base na soma dos tempos que cada um dos processadores gasta para se resolver um dado problema em paralelo. Normalmente, usado em análises comparativas de desempenho de algoritmos paralelos e sequenciais.

Eficiência – fração do tempo de execução que os processadores gastam para realizar o seu trabalho e fornece uma medida conveniente para se avaliar a qualidade em termos de rendimento dos programas em paralelo, independente do tamanho do problema abordado.

Idle-time – Tempo de espera que uma unidade de processamento fica aguardando uma comunicação.

Instrução – Operação ou comando para ser executado pelo computador.

IWMSE – Método de avaliação desenvolvido por Stutz et al. (2007) que atribui um valor a diferença entre duas imagens de modo a expressar a qualidade com base na percepção. *Abrev. Information Weighted Mean Square Error.*

IWMSE/G – Valor do IWMSE calculado usando a intensidade de cinza como medida de energia de interesse local.

IWMSE/S – Valor do IWMSE calculado usando magnitudes de borda como medida de energia de interesse local.

IWMSE/V – Valor do IWMSE calculado usando a variância local como medida de energia de interesse local.

Janela de verificação – Momento em que os processos em paralelo param a execução normal do programa para fazer uma verificação de convergência.

Linha de particionamento – Linhas que dividem uma matriz de dados (p.ex. imagem digital) em número de blocos de tamanhos aproximadamente iguais.

MIMD – Arquitetura de computadores em paralelo em que as unidades de processamento em paralelo executam múltiplas instruções de forma paralela, manipulando múltiplos dados. *Abrev. Multiple Instruction Stream Multiple Data Stream.*

MPI – Biblioteca de definições e funções utilizada em programas C ou Fortran, que possibilita o desenvolvimento de sistemas paralelos, usando memória distribuída. É implementada numa arquitetura em que a comunicação e a troca de dados entre os processadores é feita via troca de mensagens (*message passing*). *Abrev*. *Message Passing Interface*.

MSE – Erro médio quadrático. *Abrev*. *Mean Square Error.*

Overlapping partition – Abordagem onde uma matriz de dados é dividida (particionada) em conjunto de blocos menores e que se sobrepõem, sendo cada um dos blocos enviado a uma unidade de processamento diferente rodando em paralelo.

Parallel overhead – Tempo extra necessário para a coordenação de tarefas em paralelo em oposição ao trabalho que seria realmente necessário para executar a mesma tarefa serialmente.

Partição – Resultado da divisão de uma imagem maior em várias partes ou blocos menores disjuntos (*striped partition*) ou que se sobrepõem (*overlapping partition*), distribuídos às unidades de processamento diferentes.

Pel – *Ver pixel.*

Pixel – Menor unidade representativa (elemento ou ponto) de uma imagem digital. *Sin*. Pel. *Abrev*. *Picture Elements.*

PSF – Função de Espalhamento Pontual. Função que descreve a transformação da energia de um plano para outro. *Abrev*. *Point Spread Function.*

Rank – Número sequencial atribuído à uma unidade de processamento, de modo a identificá-la de um conjunto de unidades de processamento rodando em paralelo.

Realimentação – Técnica que consiste em pegar uma imagem restaurada em uma iteração anterior e usá-la como referência no próximo ciclo de restauração.

Real-time – Tempo real. Diz-se que um programa ou processo é *real-time* quando este dá resposta em tempo real.

Segmento – Consiste em uma etapa do processo em paralelo que é responsável pela restauração de uma partição da imagem. Em alguns casos, o segmento pode ser usado também para designar o par: unidade de processamento+partição.

SFM – Microscópio de varredura de força. *Abrev. Scanning Force Microscopy*.

SIPSF – Função de espalhamento espaço-invariante. *Abrev. Space-Invariante* PSF.

SNR – Relação sinal-ruído em decibéis. *Abrev*. *Signal Noise Ration*.

Speedup – *Ver aceleração*.

SPM – Microscópio de Varredura por Sonda. *Abrev*. *Scanning Probe Microscope.*

SPMD – Modelo de programação em paralelo, onde as unidades de processamento paralelas recebem e executam uma cópia de um mesmo programa de forma concorrente, porém manipulando múltiplos dados. *Abrev. Single Program Multiple Data*.

STM – Microscópio de Varredura por Tunelamento. *Abrev*. *Scanning Tunneling Microscope*.

Striped partition – Abordagem onde uma matriz de dados é dividida (particionada) em blocos disjuntos, sendo cada um dos blocos enviado a uma unidade de processamento diferente, rodando em paralelo.

SVPSF – Função de espalhamento espaço-variante. *Abrev. Space-Variante* PSF.

Tarefa – Atividade de um programa com início e fim bem definidos.

Tarefa paralela – Tarefa capaz de ser operada paralelamente com outras tarefas.

Tip – Ponteira de medição do microscópio.

APÊNDICES

Apêndice A – Desenvolvimento do método de solução de equações lineares SOR (*Sucessive OverRelaxation*)

Reescreve-se a Eq. (45)

$$\Delta\hat{x}_{rs}^{t,c+1} = -\frac{1}{\left(\partial F_{rs}\big/\partial\hat{x}_{mn}\right)\Big|_{\substack{t,c\\m=r\\n=s}}}\left\{F_{rs}\Big|_{\hat{x}^{t,c}} + \sum_{\substack{m=0\\m\neq r}}^{M}\sum_{\substack{n=0\\n\neq s}}^{M}\frac{\partial F_{rs}}{\partial\hat{x}_{mn}}\Big|_{\hat{x}^{t,\tilde{c}}}.\Delta\hat{x}_{mn}^{t,\tilde{c}}\right\} \tag{A.1}$$

onde $\Delta\hat{x}_{rs}^{t,0} = 0$, c e t são o contadores de iterações, $\tilde{c} = \begin{cases} c+1, & (m<r)\ ou\ (m=r\ \ e\ \ n\leq s) \\ c\quad , & caso\ contrário \end{cases}$

e $r,s = 0,...,M$.

Acrescentando o próprio elemento da diagonal principal à Eq. (A.1), de forma que esta se mantém, obtém-se

$$\hat{x}_{rs}^{t,c+1} = -\frac{1}{\left(\partial F_{rs}\big/\partial\hat{x}_{mn}\right)\Big|_{\substack{t,c\\m=r\\n=s}}}\left\{F_{rs}\Big|_{\hat{x}^{t,c}} + \sum_{\substack{m=0\\m\neq r}}^{M}\sum_{\substack{n=0\\n\neq s}}^{M}\frac{\partial F_{rs}}{\partial\hat{x}_{mn}}\Big|_{\hat{x}^{t,\tilde{c}}}.\Delta\hat{x}_{mn}^{t,\tilde{c}}\right\} + \Delta\hat{x}_{rs}^{tc} - \Delta\hat{x}_{rs}^{tc} \tag{A.2}$$

Desenvolvendo a Eq. (A.2), de forma a incluir o elemento da diagonal em um dos somatórios

$$\hat{x}_{rs}^{t,c+1} = -\frac{1}{\left(\partial F_{rs}\big/\partial\hat{x}_{mn}\right)\Big|_{\substack{t,c\\m=r\\n=s}}}\left\{F_{rs}\Big|_{\hat{x}^{t,c}} + \sum_{\substack{m=0\\m\neq r}}^{M}\sum_{\substack{n=0\\n\neq s}}^{M}\frac{\partial F_{rs}}{\partial\hat{x}_{mn}}\Big|_{\hat{x}^{t,\tilde{c}}}.\Delta\hat{x}_{mn}^{t,\tilde{c}}\right\} + \Delta\hat{x}_{rs}^{t,c} - \frac{\left(\partial F_{rs}\big/\partial\hat{x}_{mn}\right)\Big|_{\substack{t,c\\m=r\\n=s}}.\Delta\hat{x}_{mn}\Big|_{\substack{t,c\\m=r\\n=s}}}{\left(\partial F_{rs}\big/\partial\hat{x}_{mn}\right)\Big|_{\substack{t,c\\m=r\\n=s}}}$$

(A.1)

$$\hat{x}_{rs}^{t,c+1} = -\frac{1}{\left(\partial F_{rs} \big/ \partial \hat{x}_{mn}\right)\Big|_{\substack{t,c \\ m=r \\ n=s}}} \left\{ F_{rs}\Big|_{\hat{x}^{t,c}} + \sum_{m=0}^{M}\sum_{n=0}^{M} \frac{\partial F_{rs}}{\partial \hat{x}_{mn}}\Big|_{\hat{x}^{t,\tilde{c}}} . \Delta\hat{x}_{mn}^{t,\tilde{c}} \right\} + \hat{x}_{rs}^{t,c} \tag{A.3}$$

Chega-se, finalmente, à seguinte equação, em que o termo de correção da estimativa de $\Delta\hat{x}_{rs}^{t,c}$ se destaca

$$\Delta\hat{x}_{rs}^{t,c+1} = \Delta\hat{x}_{rs}^{t,c} - \frac{1}{\left(\partial F_{rs} \big/ \partial \hat{x}_{mn}\right)\Big|_{\substack{t,c \\ m=r \\ n=s}}} \left\{ F_{rs}\Big|_{\hat{x}^{t,c}} + \sum_{m=0}^{M}\sum_{n=0}^{M} \frac{\partial F_{rs}}{\partial \hat{x}_{mn}}\Big|_{\hat{x}^{t,\tilde{c}}} . \Delta\hat{x}_{mn}^{t,\tilde{c}} \right\} \tag{A.4}$$

Empregando-se um fator de relaxação ω no termo de correção de $\Delta\hat{x}_{rs}^{t,c}$, obtém-se o método iterativo SOR (*Sucessive OverRelaxation*) para a solução do sistema de equações lineares, a partir do método de Gauss-Seidel empregado na Eq. (A.1).

$$\Delta\hat{x}_{rs}^{t,c+1} = \Delta\hat{x}_{rs}^{t,c} - \frac{\omega}{\left(\partial F_{rs} \big/ \partial \hat{x}_{mn}\right)\Big|_{\substack{t,c \\ m=r \\ n=s}}} \left\{ F_{rs}\Big|_{\hat{x}^{t,c}} + \sum_{m=0}^{M}\sum_{n=0}^{M} \frac{\partial F_{rs}}{\partial \hat{x}_{mn}}\Big|_{\hat{x}^{t,\tilde{c}}} . \Delta\hat{x}_{mn}^{t,\tilde{c}} \right\} \tag{A.5}$$

onde ω é o fator de relaxação do método SOR. Para $\omega = 1$, a Eq. (A.1) se mantém e o método empregado na solução do sistema de equação linear é o de Gauss-Seidel, caso contrário ($\omega \neq 1$), o método aplicado é o SOR.

Apêndice B – Desenvolvimento das expressões de F_{rs} e F_{mn} para o funcional de regularização.

Partindo do funcional geral de minimização

$$Q(\hat{x}) = \sum_{i=0}^{M}\sum_{j=0}^{M}\left(y_{ij} - \sum_{k=-N}^{N}\sum_{l=-N}^{N} b_{kl}\,\hat{x}_{i+k,\,j+l} \right)^2 + \alpha S \tag{B.1}$$

tem-se que a derivada-parcial F_{rs} no ponto (r,s) é dado por

$$F_{rs} = \frac{\partial Q(\hat{x})}{\partial \hat{x}_{rs}} = 2\sum_{i=0}^{M}\sum_{j=0}^{M}\left[\left(y_{ij} - \sum_{k=-N}^{N}\sum_{l=-N}^{N} b_{kl}\,\hat{x}_{i+k,\,j+l} \right)\left(-\sum_{k=-N}^{N}\sum_{l=-N}^{N} b_{kl}\,\frac{\partial \hat{x}_{i+k,\,j+l}}{\partial \hat{x}_{rs}} \right) \right] + \alpha\,\frac{\partial S}{\partial \hat{x}_{rs}} \tag{B.2}$$

Sendo

$$\frac{\partial \hat{x}_{ij}}{\partial \hat{x}_{rs}} = \delta(i,r)\,\delta(j,s) \tag{B.3}$$

tem-se, então, que

$$\begin{array}{lll} \dfrac{\partial \hat{x}_{i+k,\,j+l}}{\partial \hat{x}_{rs}} & = & \delta(i+k,r)\,\delta(j+l,s) \quad \therefore \quad \displaystyle\sum_{k=-N}^{N}\sum_{l=-N}^{N} b_{kl}\,\frac{\partial \hat{x}_{i+k,\,j+l}}{\partial \hat{x}_{rs}} \;=\; b_{r-i,\,s-j} \\ r = i+k, & & k = r-i, \\ s = j+l & & l = s-j \end{array} \tag{B.4}$$

onde $\delta(.)$ representa o delta de Kronecker.

Aplicando-se a Eq. (B.4) na Eq. (B.2), tem-se

$$F_{rs} = -2\sum_{i=0}^{M}\sum_{j=0}^{M}\left[\left(y_{ij} - \sum_{k=-N}^{N}\sum_{l=-N}^{N} b_{kl}\,\hat{x}_{i+k,\,j+l} \right) b_{r-i,\,s-j} \right] + \alpha\,\frac{\partial S}{\partial \hat{x}_{rs}} \tag{B.5}$$

Tomando-se o funcional de regularização S descrito na Eq. (36), reproduzido aqui na equação (B.6),

$$S = D_q(\hat{x}, \bar{x}) = \frac{1}{1+q} \sum_{i=0}^{M} \sum_{j=0}^{M} \left\{ \hat{x}_{ij} \left[\frac{\hat{x}_{ij}^q - \bar{x}_{ij}^q}{q} \right] - \bar{x}_{ij}^q (\hat{x}_{ij} - \bar{x}_{ij}) \right\} \tag{B.6}$$

e derivando-a em função de $\hat{x}_{rs}$, tem-se

$$\frac{\partial S}{\partial \hat{x}_{rs}} = \frac{1}{1+q} \sum_{i=0}^{M} \sum_{j=0}^{M} \left\{ \left(\frac{\partial \hat{x}_{ij}}{\partial \hat{x}_{rs}} \left(\frac{\hat{x}_{ij}^q - \bar{x}_{ij}^q}{q} \right) + \hat{x}_{ij} \frac{\partial}{\partial \hat{x}_{rs}} \left(\frac{\hat{x}_{ij}^q - \bar{x}_{ij}^q}{q} \right) \right) - \frac{\partial \left(\bar{x}_{ij}^q (\hat{x}_{ij} - \bar{x}_{ij}) \right)}{\partial \hat{x}_{rs}} \right\} \tag{B.7}$$

$$\frac{\partial S}{\partial \hat{x}_{rs}} = \frac{1}{1+q} \left[\frac{\hat{x}_{rs}^q - \bar{x}_{rs}^q}{q} + \hat{x}_{rs} \frac{q \hat{x}_{rs}^{q-1}}{q} - \bar{x}_{rs}^q \right] = \frac{\hat{x}_{rs}^q - \bar{x}_{rs}^q}{q} \tag{B.8}$$

Aplicando-se a Eq. (B.8) na Eq. (B.5), tem-se

$$F_{rs} = -2 \sum_{i=0}^{M} \sum_{j=0}^{M} \left[\left(y_{ij} - \sum_{k=-N}^{N} \sum_{l=-N}^{N} b_{kl} \hat{x}_{i+k, j+l} \right) b_{r-i, s-j} \right] + \frac{\alpha}{q} \left(\hat{x}_{rs}^q - \bar{x}_{rs}^q \right) \tag{B.9}$$

Uma vez que $b_{r-i,s-j} = 0$ quando $|r-i| > N$ ou $|s-j| > N$, tem-se que

$$-N \le r - i \le N \Rightarrow r - N \le i \le r + N \tag{B.10}$$

e

$$-N \le s - j \le N \Rightarrow s - N \le j \le s + N \tag{B.11}$$

Pode-se redefinir, então, os índices dos somatórios da Eq. (B.9)

$$F_{rs} = -2 \sum_{i=r-N}^{r+N} \sum_{j=s-N}^{s+N} \left[\left(y_{ij} - \sum_{k=-N}^{N} \sum_{l=-N}^{N} b_{kl} \hat{x}_{i+k, j+l} \right) b_{r-i, s-j} \right] + \frac{\alpha}{q} \left(\hat{x}_{rs}^q - \bar{x}_{rs}^q \right) \tag{B.12}$$

Para $q \to 0$, tem-se que o funcional S construído com distâncias de Bregman, Eq. (36), representa o funcional de mínima energia, Eq. (34). Então, para o caso especial $q = 0$, substitui-se o funcional S, usando a Eq. (B.6), por

$$S = \frac{1}{2}\sum_{i=0}^{M}\sum_{j=0}^{M}\left(\hat{x}_{ij} - \bar{x}_{ij}\right)^2 \tag{B.13}$$

Assim, quando $q = 0$, tem-se

$$F_{rs} = -2\sum_{i=r-N}^{r+N}\sum_{j=s-N}^{s+N}\left[\left(y_{ij} - \sum_{k=-N}^{N}\sum_{l=-N}^{N} b_{kl}\hat{x}_{i+k,j+l}\right)b_{r-i,s-j}\right] + \alpha\left(\hat{x}_{rs} - \bar{x}_{rs}\right) \tag{B.14}$$

- ***Derivada de segunda ordem***

Fazendo: $F_{mn} = \dfrac{\partial F_{rs}}{\partial \hat{x}_{mn}} = \dfrac{\partial^2 Q(\hat{x})}{\partial \hat{x}_{rs}\partial \hat{x}_{mn}}$

$$F_{mn} = -2\sum_{i=r-N}^{r+N}\sum_{j=s-N}^{s+N}\left[\left(-\sum_{k=-N}^{N}\sum_{l=-N}^{N} b_{kl}\frac{\partial \hat{x}_{i+k,j+l}}{\partial x_{mn}}\right)b_{r-i,s-j}\right] + \frac{\alpha}{q}\frac{\partial\left(\hat{x}_{rs}^{q} - \bar{x}_{rs}^{q}\right)}{\partial x_{mn}} \tag{B.15}$$

onde

$$\begin{array}{lll} \dfrac{\partial \hat{x}_{i+k,j+l}}{\partial \hat{x}_{mn}} & = & \delta(i+k,m)\delta(j+l,n) \quad \therefore \quad \displaystyle\sum_{k=-N}^{N}\sum_{l=-N}^{N} b_{kl}\frac{\partial \hat{x}_{i+k,j+l}}{\partial \hat{x}_{mn}} \quad = \quad b_{m-i,n-j} \\ m = i + k, & & k = m - i, \\ n = j + l & & l = n - j \end{array} \tag{B.16}$$

$$F_{mn} = 2\sum_{i=r-N}^{r+N}\sum_{j=s-N}^{s+N} b_{m-i,n-j}b_{r-i,s-j} + \alpha\hat{x}_{rs}^{q-1}\delta(r,m)\delta(s,n) \tag{B.17}$$

Fazendo os somatórios em função de k e l, tem-se

$$F_{mn} = 2\sum_{k=-N}^{N}\sum_{l=-N}^{N} b_{m-r-k,n-s-l} b_{kl} + \alpha \hat{x}_{rs}^{q-1}\delta(r,m)\delta(s,n) \tag{B.18}$$

Como $b_{m-r-k,n-s-l} = 0$ quando $|m-r-k| > N$ ou $|n-s-l| > N$, tem-se que

$$F_{mn} = 0 \text{ quando } |m-r| > 2N \text{ ou } |n-s| > 2N \tag{B.19}$$

Neste caso, pode-se limitar ainda mais os índices dos somatórios da Eq. (B.18) fazendo

$$k = máx(-N + (r-m), -N), \ldots, \min(N - (r-m), N) \tag{B.20}$$

e

$$l = máx(-N + (s-n), -N), \ldots, \min(N - (s-n), N) \tag{B.21}$$

Apêndice C – Pseudocódigo da nova versão do programa paralelo de restauração

```
/***********************************************
// Declaração das variáveis
/***********************************************
  // Áreas de trabalho
  B       // Operador de espalhamento pontual (PSF)
  BX      // Partição da matriz de convolução B*X[i]
  Y       // Imagem a ser restaurada
  Y[i]    // Partição da imagem que será restaurada
  YBX     // Partição da matriz diferença Y[i]-BX[i]
  Xtm0    // Partição da imagem estimada de X[i] no instante t=0,1,2,...
  Xtm1    // Partição da imagem estimada de X[i] no instante t+1
  Xref    // Partição da imagem de referência
  Xrec    // Partição da imagem de restaurada
  Frs     // Partição de dados contendo a deriva primeira
  Fmn     // Partição de dados contendo a deriva segunda

  // Outras variáveis
  np            // Número total de processadores rodando em paralelo
  my_rank       // Número do rank do processo corrente
  uproc_acima   // Numero do rank do processo do segmento acima
  uproc_abaixo  // Numero do rank do processo do segmento abaixo
  ImRoot        // Identifica se o processo corrente é o root
  first_part    // Identifica se o processo corrente é o primeiro segmento
  last_part     // Identifica se o processo corrente é o último segmento
  Qmin          // Valor mínimo da função Q
  Qmin_parc     // Valor parcial da função Q
  Qmin_Xtm1     // Valor mínimo da função Q para a estimativa X no instante t+1
  t             // Contador de iterações
  nIter         // Número máximo de iterações

/***********************************************
// Procedimentos iniciais
/***********************************************

  // Inicialização do processamento paralelo
  /*********************************************

  // Função de inicialização do processamento paralelo
  MPI_Init(...);

  // Acha o rank do processador corrente (my_rank)
  MPI_Comm_rank(..., my_rank);

  // Acha o numero total de processadores (np)
  MPI_Comm_size(..., np);

  // Obtêm as unidades de processamentos acima e abaixo de acordo
  //com rank (my_rank) do processo corrente
  uproc_acima  := pega_processador_acima(my_rank);
  uproc_abaixo := pega_processador_abaixo(my_rank);

  // Inicializa variáveis de controle
  root   := 0               // Estabelece que o processo-0 assumirá o papel de
"root"
  ImRoot := (my_rank=root);  // Identifica o processo root


  // Verificação de argumentos iniciais
  /*********************************************
  - Faz um conjunto de verificações dos argumentos: faixa de valores, parâmetros
    opcionais/obrigatórios etc.
```

```
  // Distribuição dos fragmentos da imagem
  ***********************************************/

  // O root faz a leitura da imagem, particiona e distribui as
  //partições para os outros segmentos.
  IF (ImRoot)
    - Lê a imagem que deverá ser restaurada (Y);

    - Particiona Y em np partições de acordo com estratégia de
      distribuição de cargas de modo a minimizar desbalanceamentos;

    - Distribui as partições (Y[i]) para cada unidade de processamento
(Broadcast),
      de acordo com o rank de cada uma delas; MPI_Bcast(...);
  ELSE
    - O processo aguarda pelo recebimento (sincrono) da partição (Y[i]),
      enviada pelo root, que irá restaurar em paralelo;
  END_IF


  //// Alocacoes dinâmicas de memória
  ************************************************/
  - Reserva espaços de memória necessários para as áreas de trabalho;

  - Faz a carga de dados inicial dessas áreas
     B    := calcula_PSF()    // Constrói o operador de borramento
     Xtm0 := Y[i]             // Inicia a partição X no instante t=0
                              // com uma cópia da partição Y[i];
     Xref := imagem de referência // Carrega Xref com a imagem referência;
     Xrec := Y[i]                 // Faz inicialmente Xrec igual a Y[i];
     ....

  - Faz o cálculo da convolução b *  b              usado nos cálculos em
paralelo
                                 kl   m-r-k, n-s-l

  // Calcula a convolução B * Xtm0
  BX = conv(B, Xtm0)

  // Calcula o mínimo da partição (mínimo parcial)
  Qmin_parc = minQuad(Y, Xtm0, Xref, ConvBX, Alfa, q)

  // Calcula em paralelo o valor mínimo da função Q (Qmin) no instante t=0
  MPI_Allreduce(Qmin_parc, Qmin, ...)


  /***********************************************
  //// Loop principal do processo de restauração
  ***********************************************/
  FOR t := 0 TO nIter

    // Faz o prefetching message das áreas S0 e S1 da partição Y-BX
    MPI_Irecv(YBX(S0), uproc_acima  )
    MPI_Irecv(YBX(S1), uproc_abaixo )


    // Faz o cálculo em paralelo das diferenças YBX
    FOR ...
      YBX := Y - BX  // Calcula pontualmente as diferenças
    END_FOR


    // Envia (assíncrono) as áreas A0 e A2 para atualizar S0 e S1 da partição YBX
    MPI_Isend(YBX(A0), uproc_acima,  ...) // Envio da área-A0 p/ a partição YBX
acima
    MPI_Isend(YBX(A2), uproc_abaixo, ...) // Envio da área-A1 p/ a partição YBX
abaixo
```

```
    // Recebimento síncrono das áreas S0 e S1 da partição YBX
    MPI_Wait(YBX(S0)...)  // Recebimento síncrono da área-S0
    MPI_Wait(YBX(S1)...)  // Recebimento síncrono da área-S1


    // Faz o prefetching message das áreas S0 e S1 da partição X no instante t+1
    MPI_Irecv(Xtm1(S0), uproc_acima,  ...);
    MPI_Irecv(Xtm1(S1), nproc_abaixo, ...);


    // Faz o calculo em paralelo das partições Frs e Fmn
    FOR ...
      Frs := ...  // Cálculo da derivada primeira
      Fmn := ...  // Cálculo da derivada segunda
      Xtm1 := pXtm0 - Gama * (Frs/Fmn)  // Cálculo da nova estimativa de X
                                        //(X no instante t+1)
    END_FOR


    // Atualização das áreas S0 e S1 da partição Xtm1
    MPI_Isend(Xtm1(A0), nproc_acima,  ...)  // Envio assíncrono da área A0 p/ a
                                            // partição Xtm1 acima
    MPI_Isend(Xtm1(A2), nproc_abaixo, ...)  // Envio assíncrono da área A2 p/ a
                                            // partição Xtm1 abaixo


    // Recebimento síncrono das áreas S0 e S1 da partição Xtm1
    MPI_Wait(Xtm1(S0)...)  // Recebimento síncrono da área-S0
    MPI_Wait(Xtm1(S1)...)  // Recebimento síncrono da área-S1


    // Calcula a convolução B * Xtm1
    BX = conv(B, Xtm1)

    // Calcula o mínimo da partição (minimo parcial)
    Qmin_parc = minQuad(Y, Xtm1, Xref, ConvBX, Alfa, q)

    // Calcula em paralelo o valor da função Q (Qmin) da nova
    // estimativa X no instante t+1
    MPI_Allreduce(Qmin_parc, Qmin_Xtm1, ...)


    // Verifica se Qmin_Xtm1 é mínimo
    IF (Qmin_Xtm1 < Qmin)
      Qmin := Qmin_Xtm1  // Armazena o novo mínimo valor da função Q
      Xrec := Xtm1       // Armazena a nova estimativa em Xrec e aceita-a como
solução
    ENDIF


    // Verifica os critérios convergência foram atingidos
    IF (critérios de convergência foram atingidos?)
      - Encerra o processo de restauração e sai do laço.
    END_IF


    // Atribui valores para a próxima iteração
    ************************************************/

    // Verifica se a realimentação deve ser aplicada
    IF (Aplica realimentação?)
      Xref := Xtm1
    END_IF

    // Comuta áreas de trabalho
    Aux   := Xtm0;
    Xtm0  := Xtm1;
```

```
      Xtm1  := Aux;

    END_FOR


    //// Encerramento
    ***********************************************/
      // Agrupa distribuições da imagem restaurada no root
      MPI_Gatherv(Xrec, root, ...);

      - Salva a imagem restaurada Xrec em um novo arquivo de saída

      - Libera alocações de memória

      // Encerra o processamento em paralelo
      MPI_Finalize();

  FIM
```

Apêndice D – Relação completa dos gráficos de tempo e desempenho dos estudos de caso I a V

- **Estudo de caso I – 128x128**

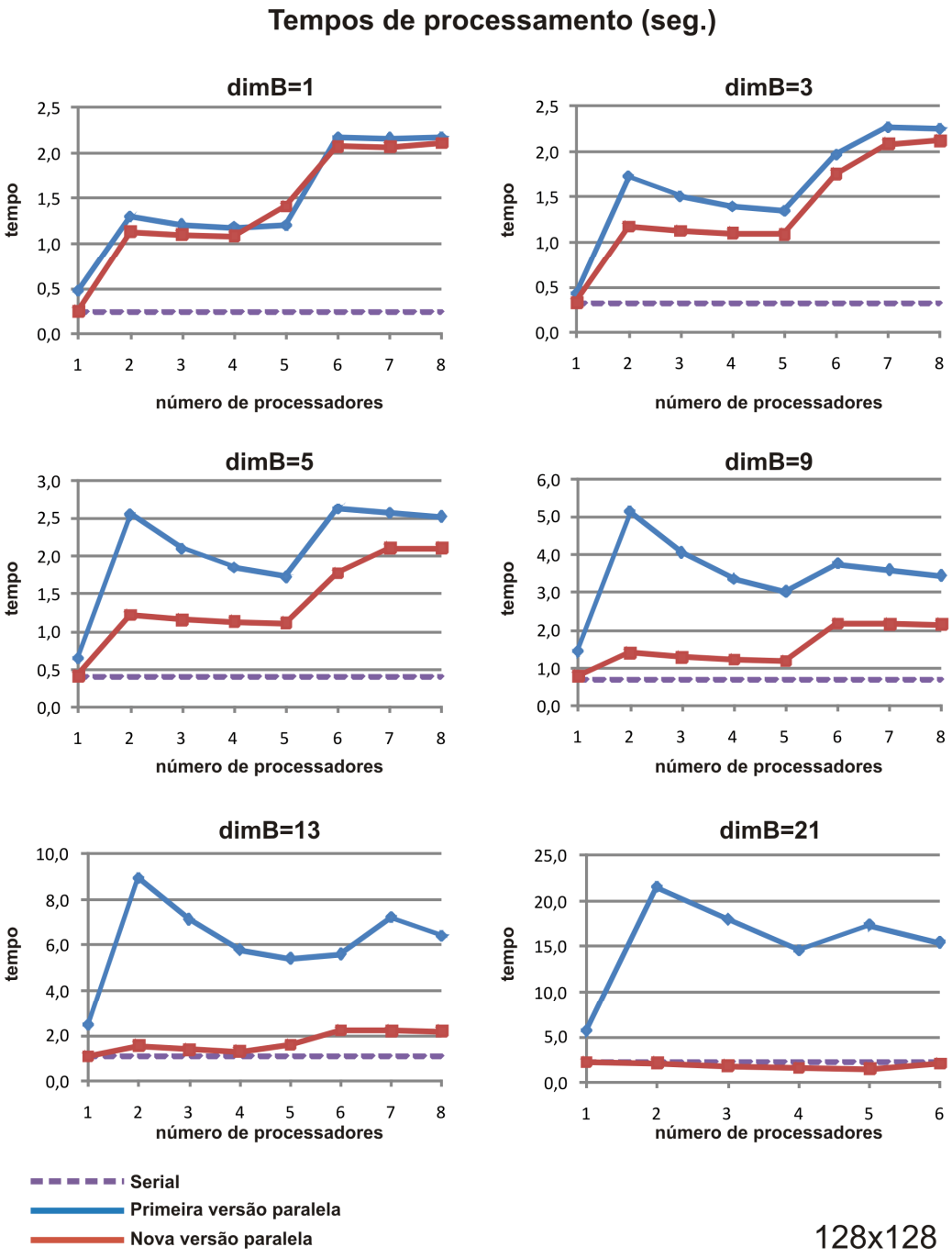


Figura 12.1 - Tempos de execução obtidos com a restauração de uma imagem de 128x128 pixels (Caso I).

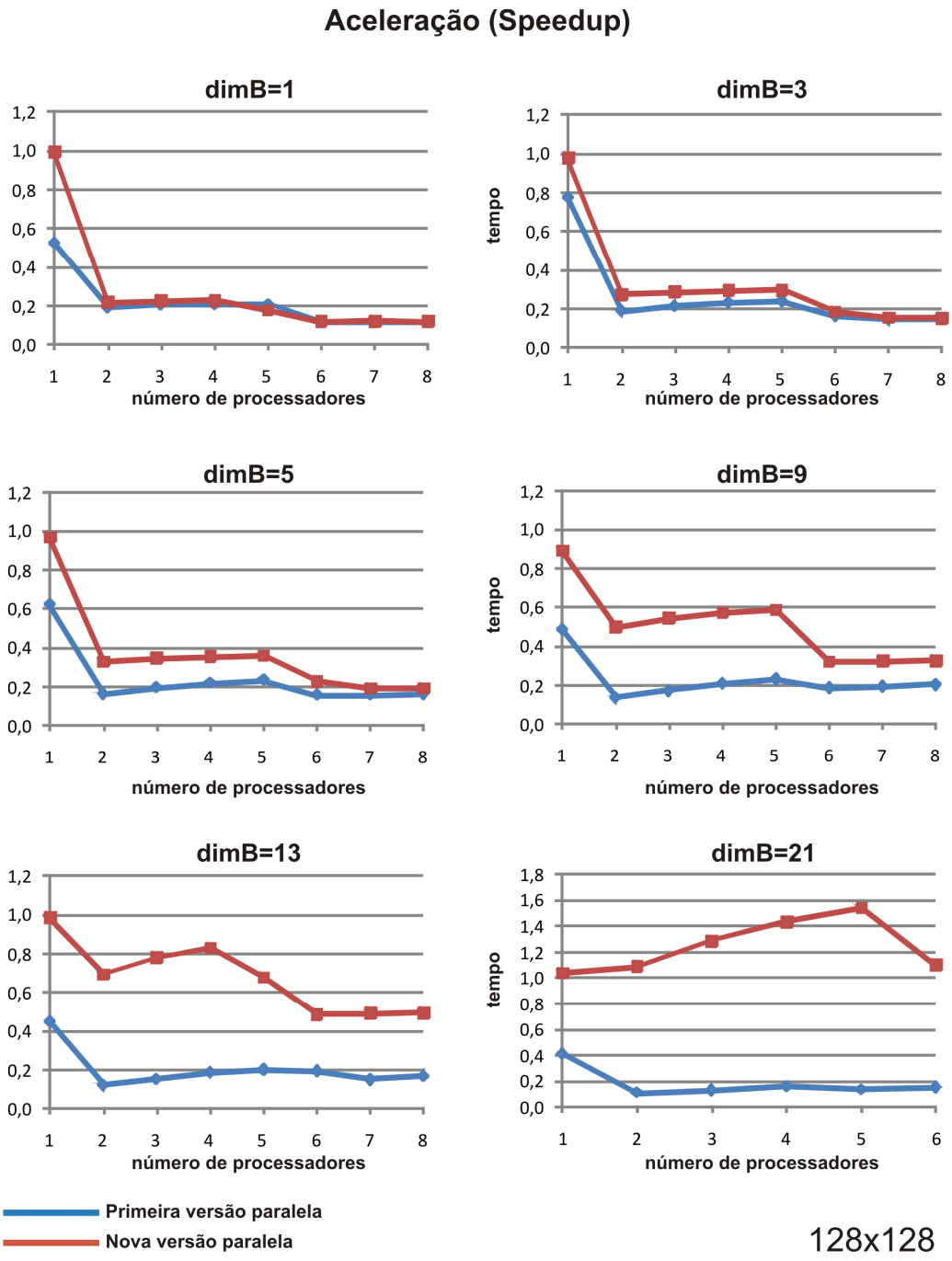


Figura 12.2 - Caso I: 128x128. Speedups.

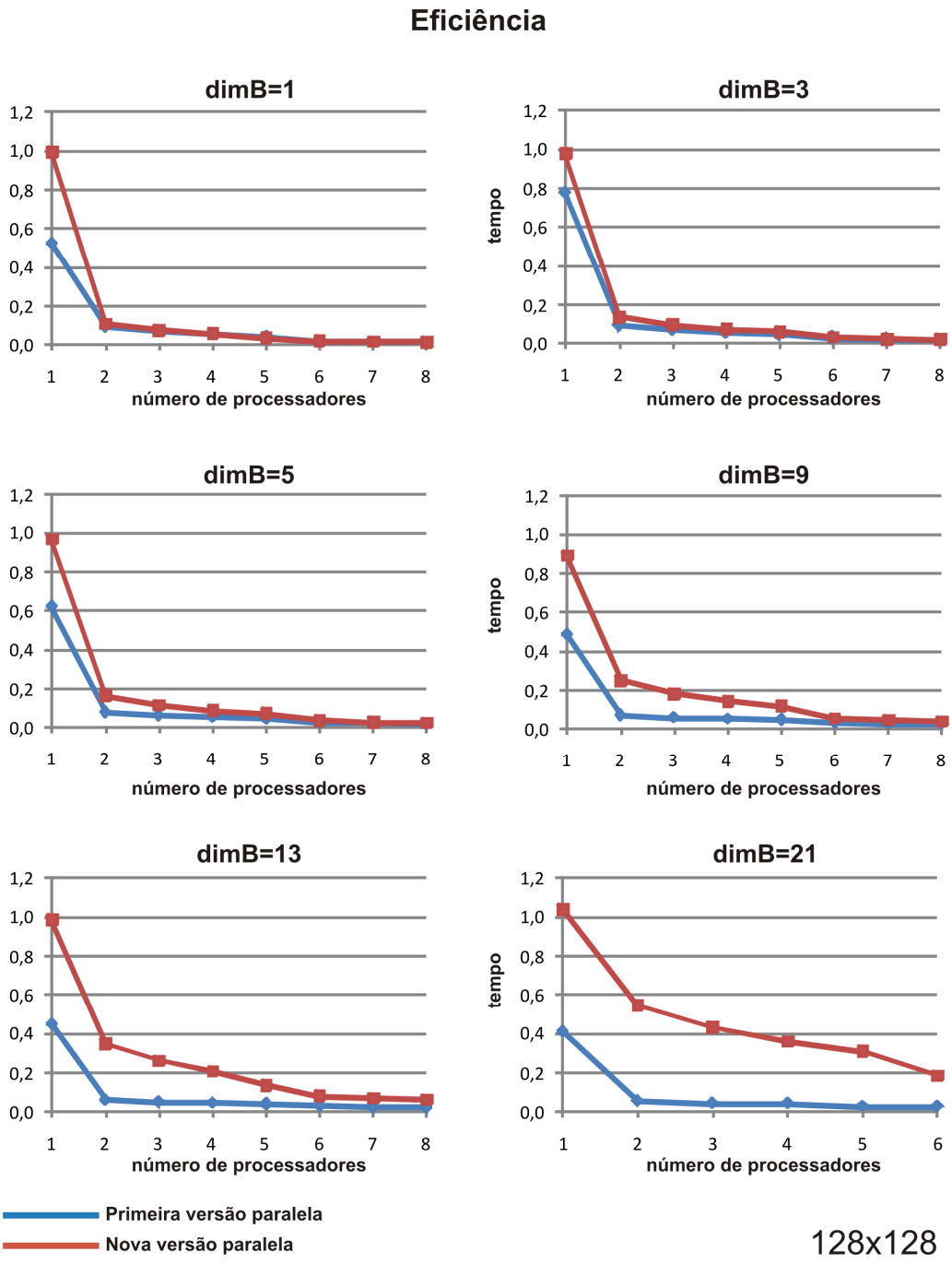


Figura 12.3 - Caso I: 128x128. Eficiências.

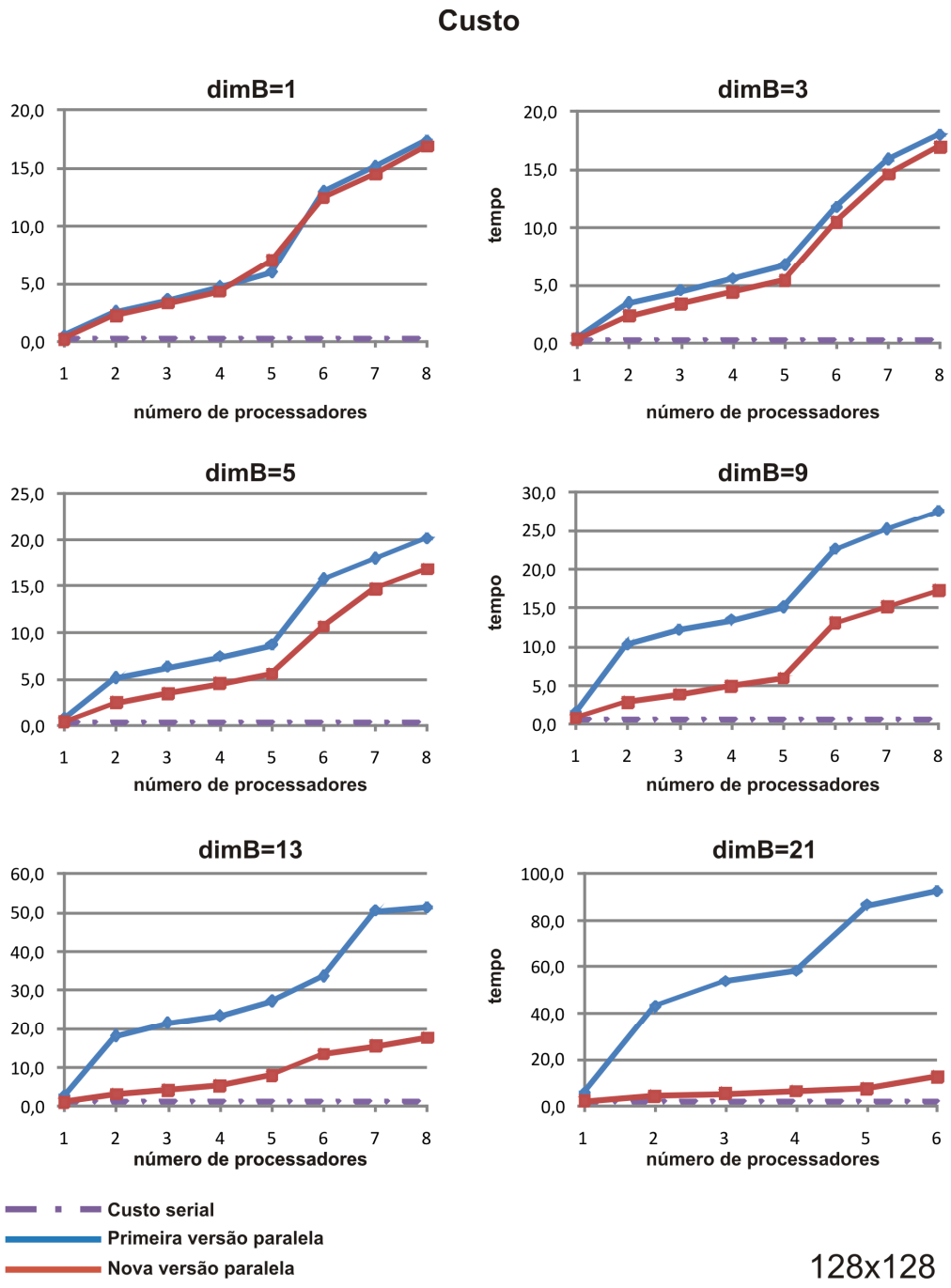


Figura 12.4 - Caso I: 128x128. Custos.

- **Estudo de caso II – 256x256**

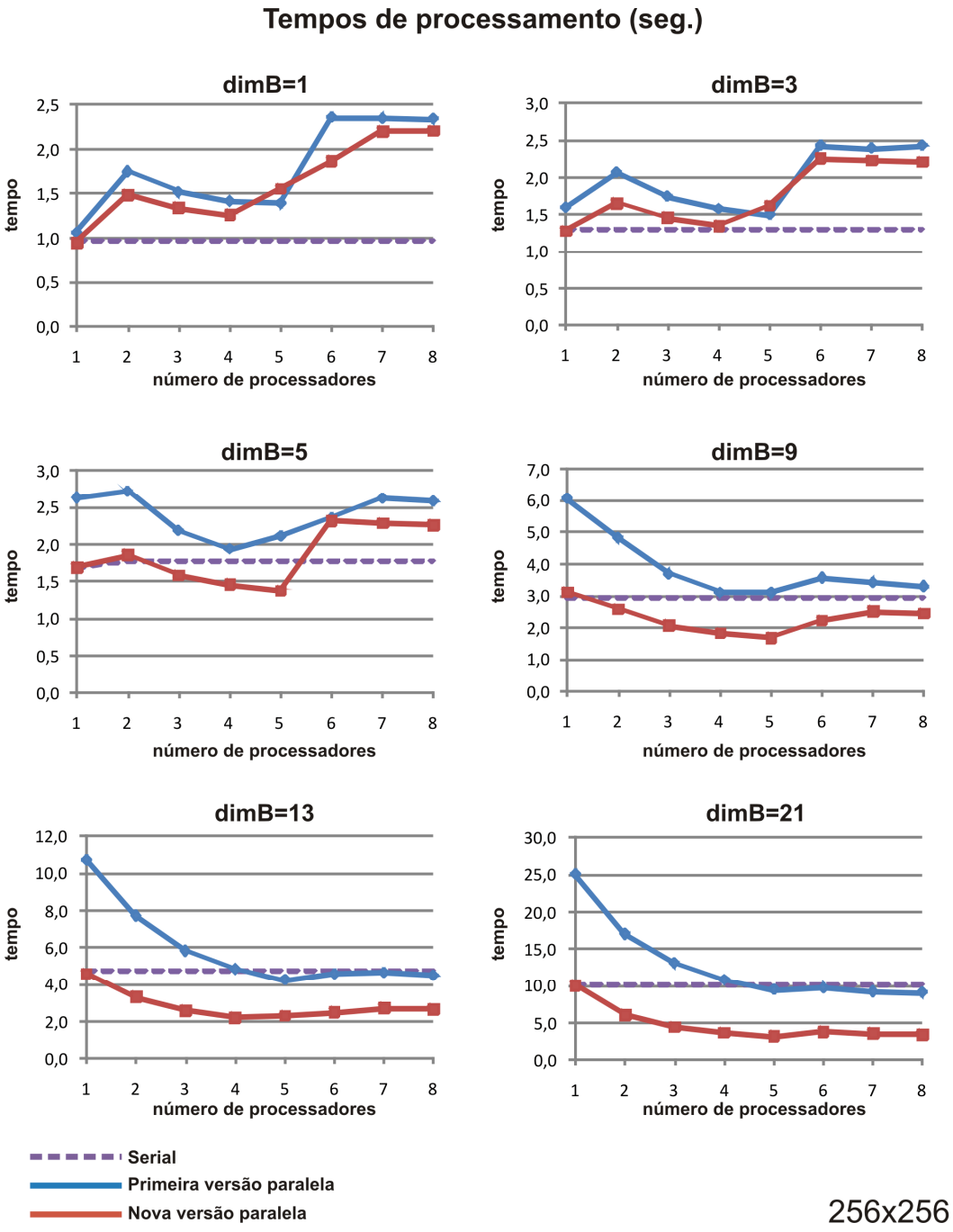


Figura 12.5 - Caso II: 256x256. Tempos de execução.

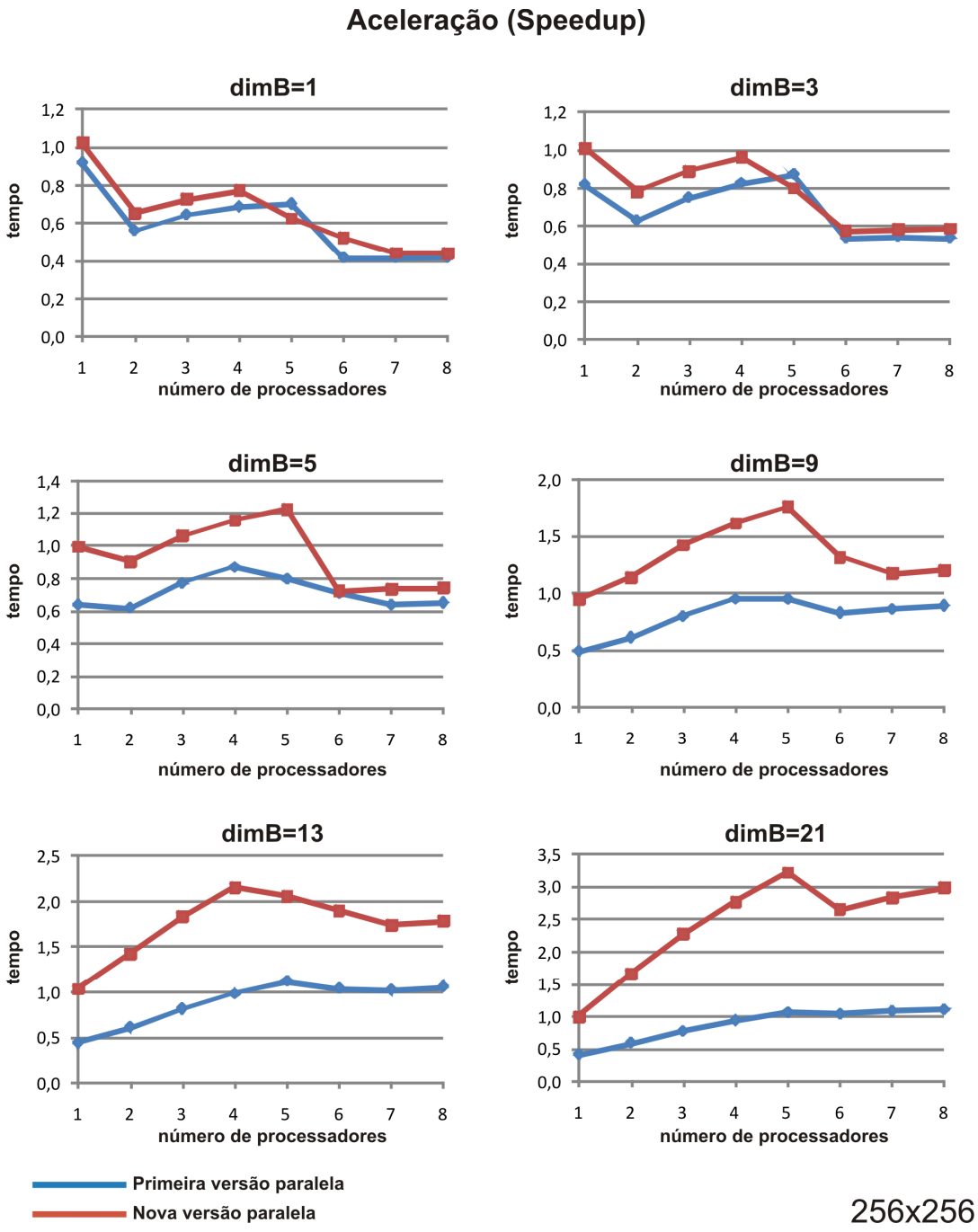


Figura 12.6 - Caso II: 256x256. Speedups.

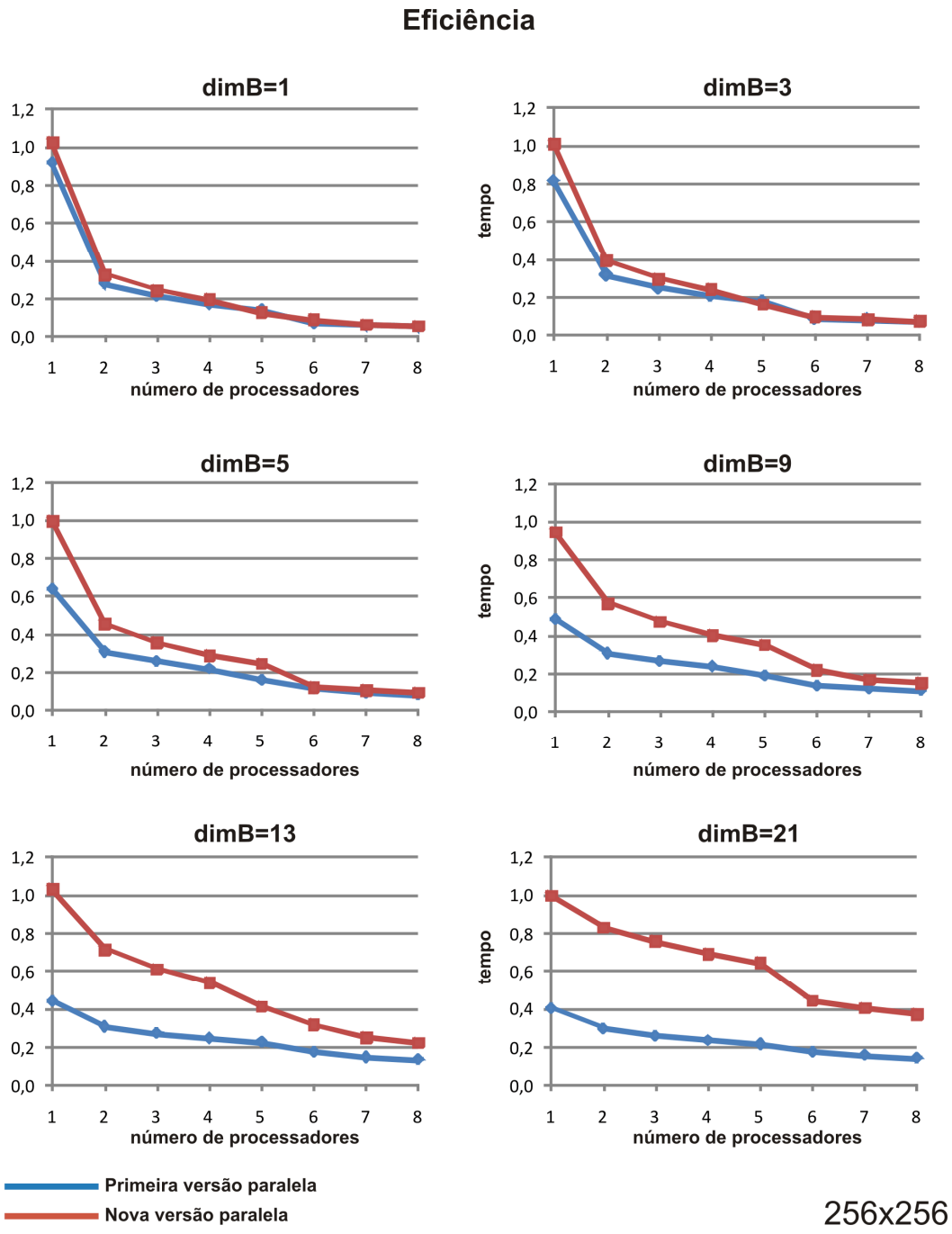


Figura 12.7 - Caso II: 256x256. Eficiências.

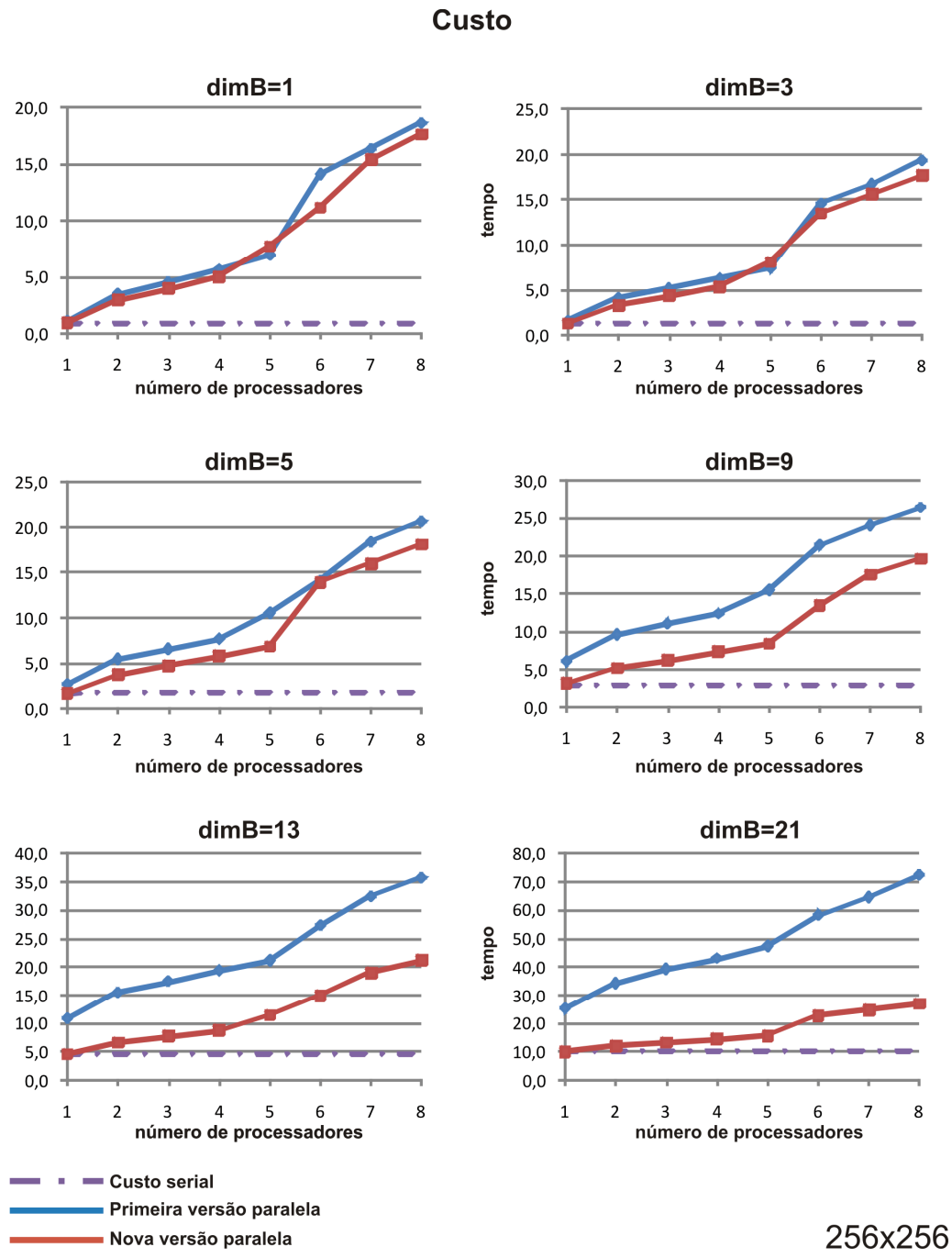


Figura 12.8 - Caso II: 256x256. Custos.

- **Estudo de caso III – 512x512**

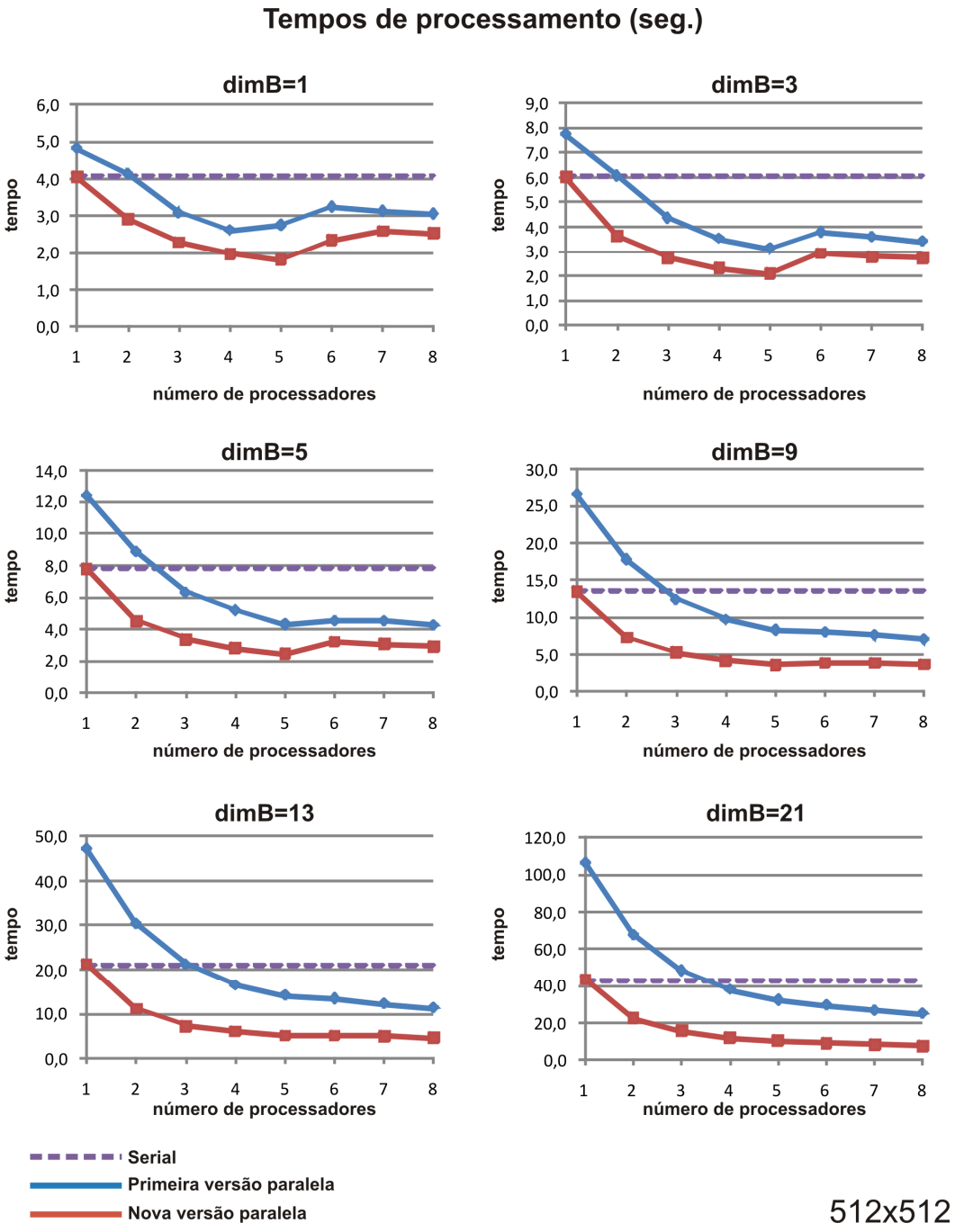


Figura 12.9 - Caso III: 512x512. Tempos de execução.

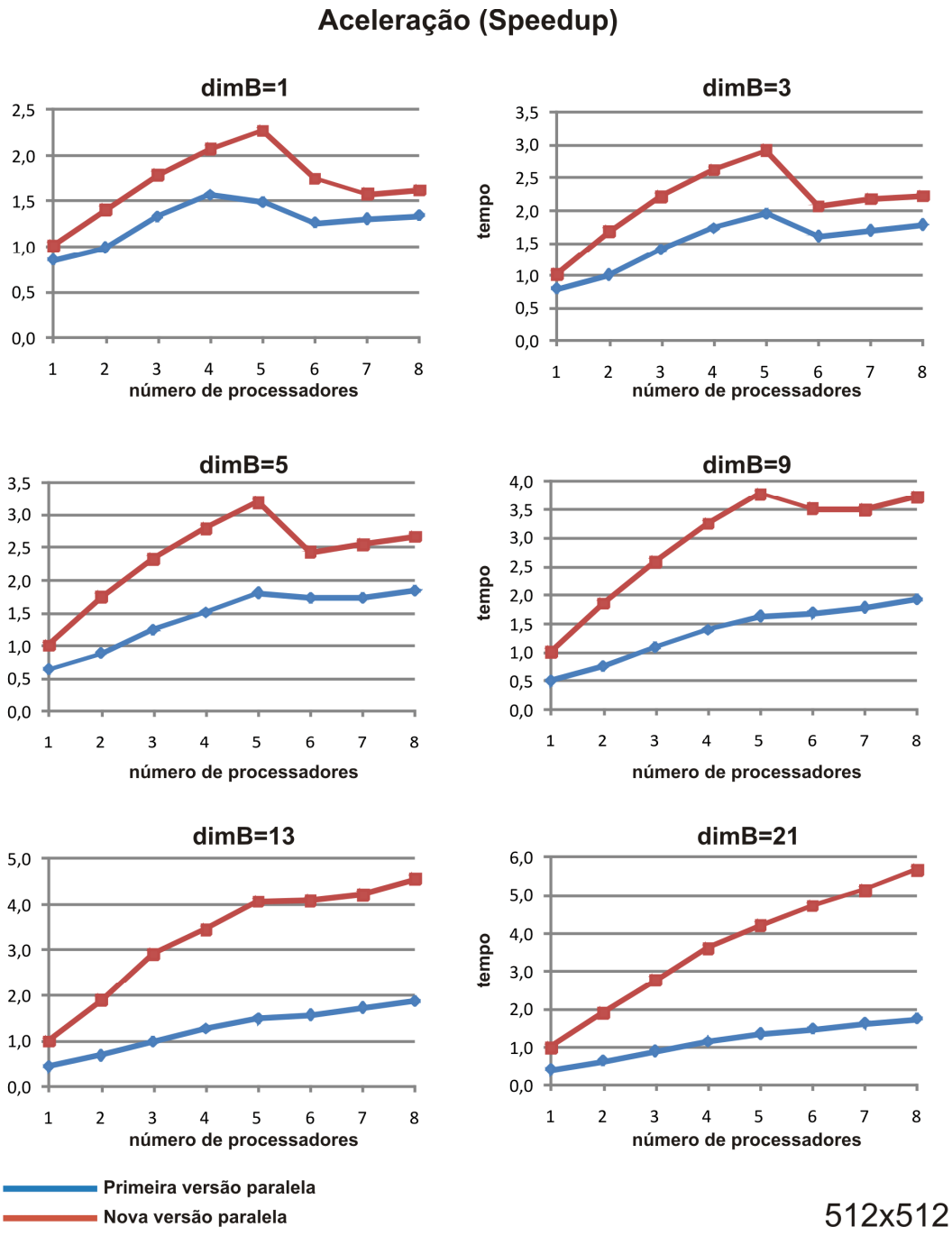


Figura 12.10 -Caso III: 512x512. Speedups.

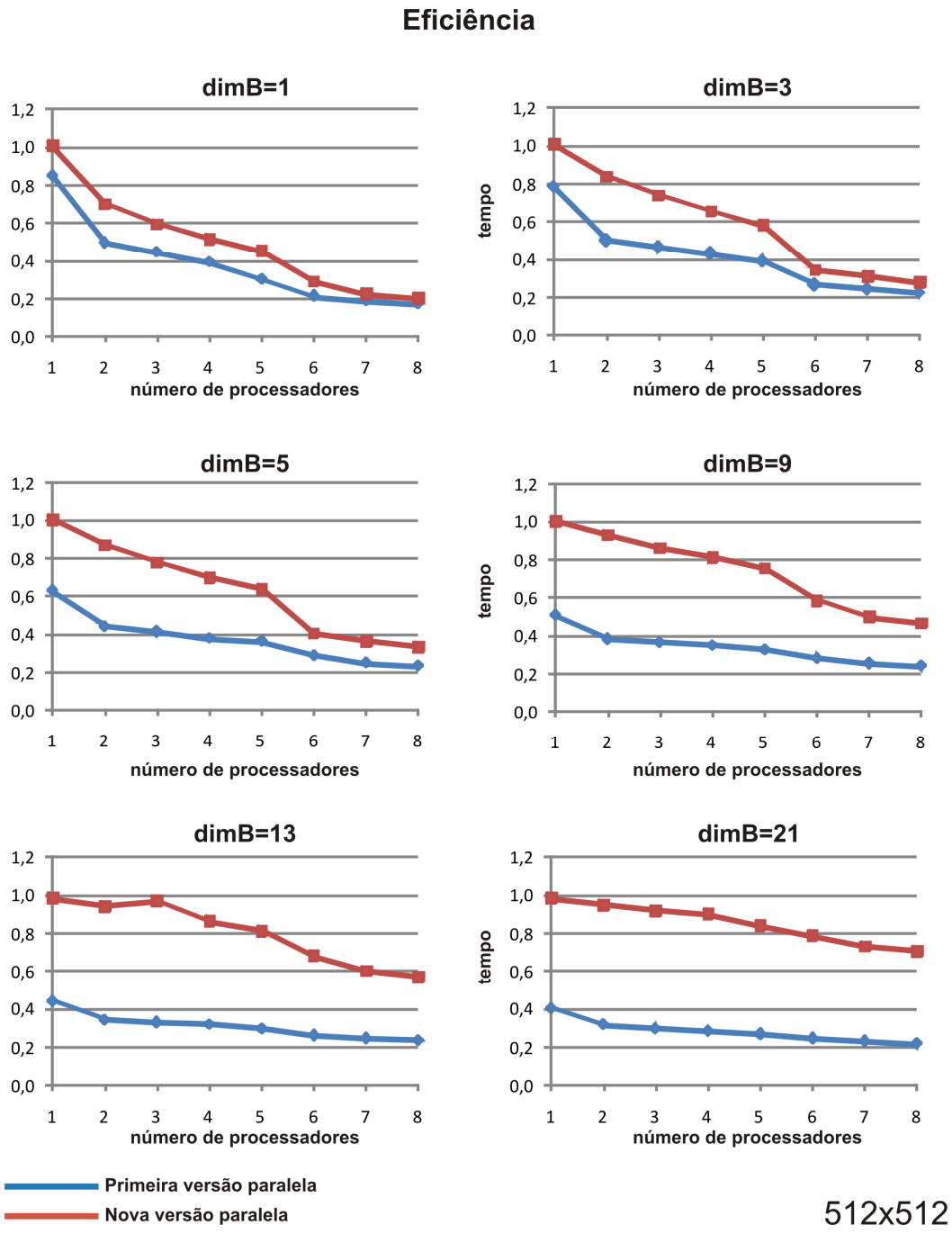


Figura 12.11 -Caso III: 512x512. Eficiências.

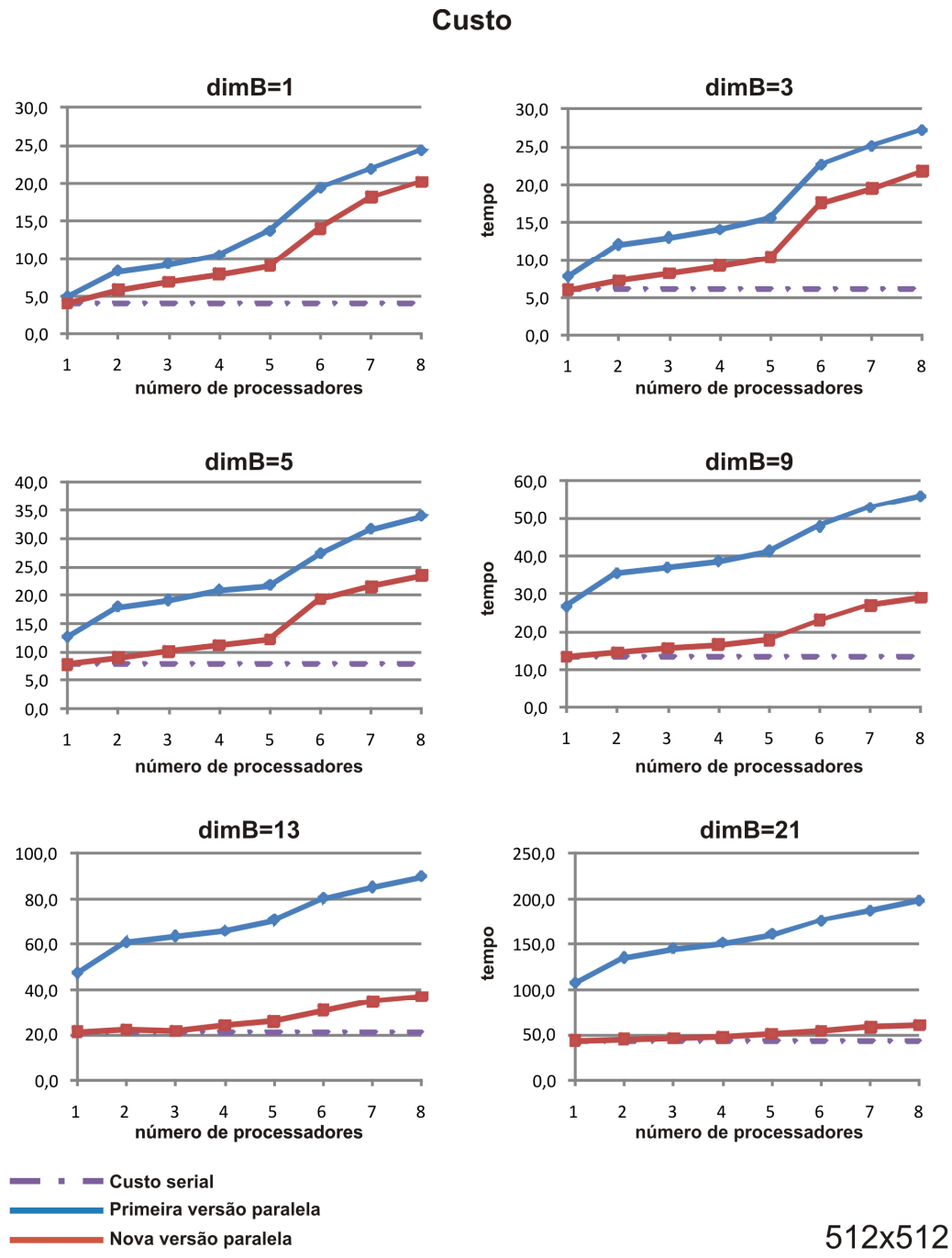


Figura 12.12 -Caso III: 512x512. Custos.

- **Estudo de caso VI – 1024x1024**

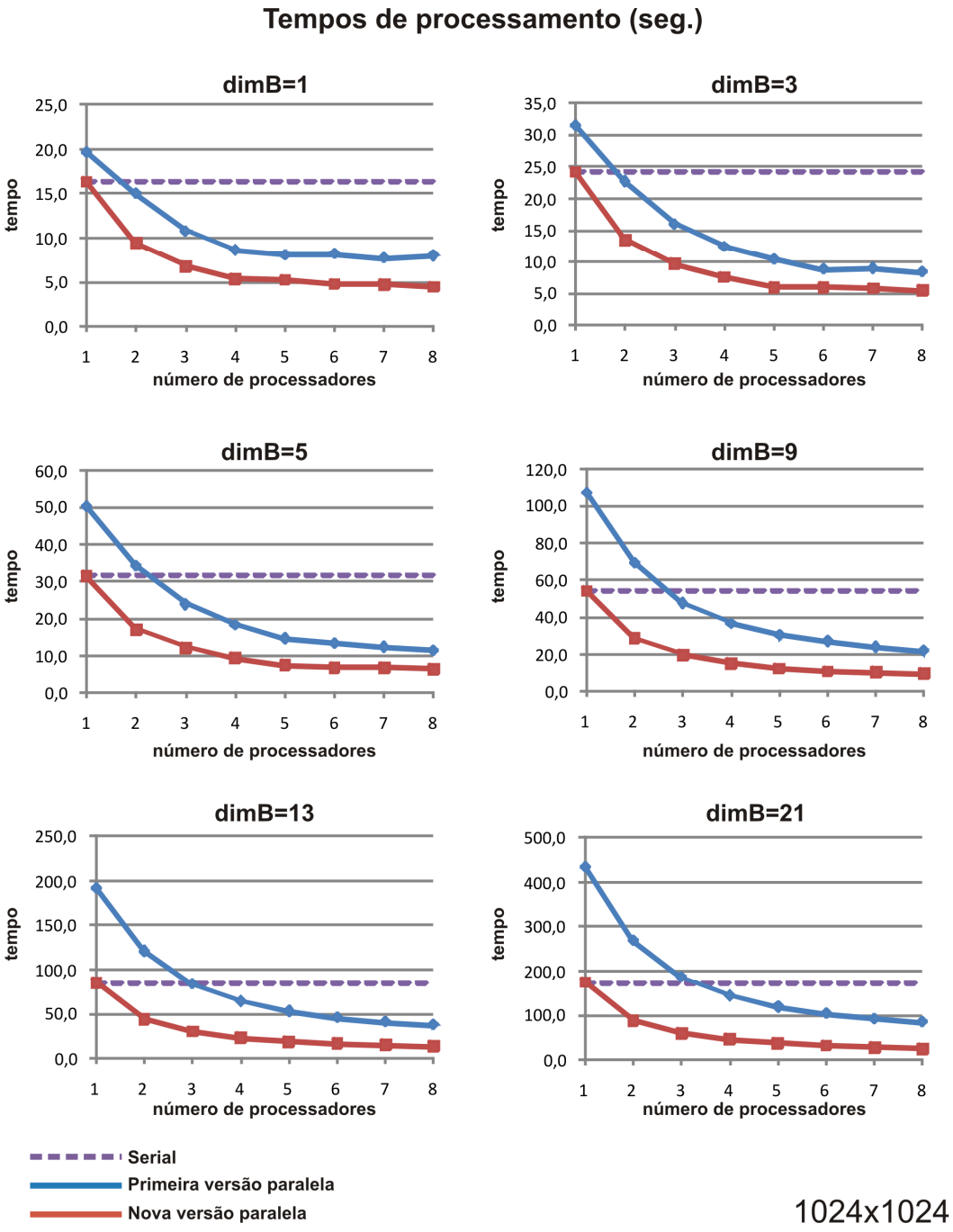


Figura 12.13 -Caso IV: 1024x1024. Tempos de execução.

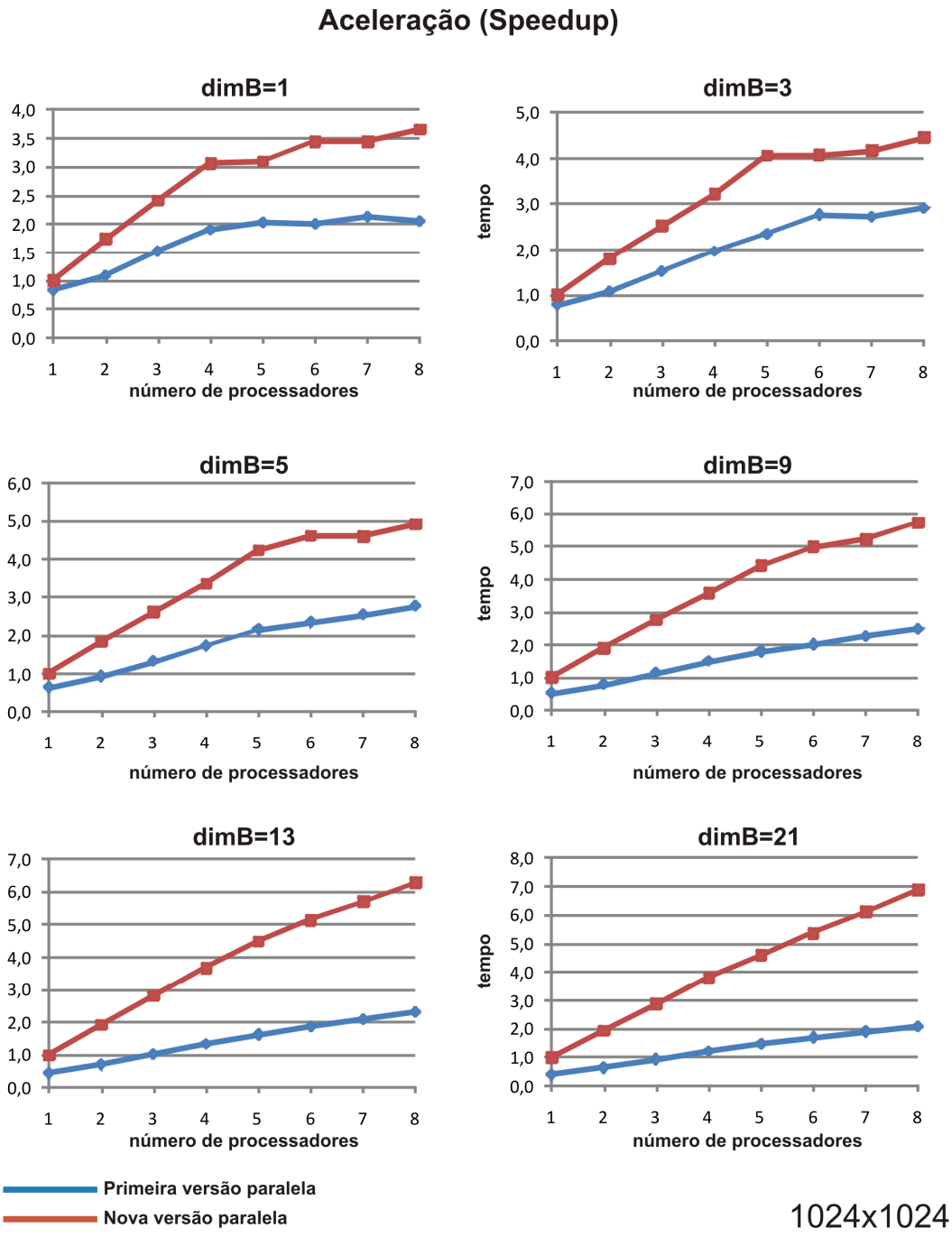


Figura 12.14 -Caso IV: 1024x1024. Speedups.

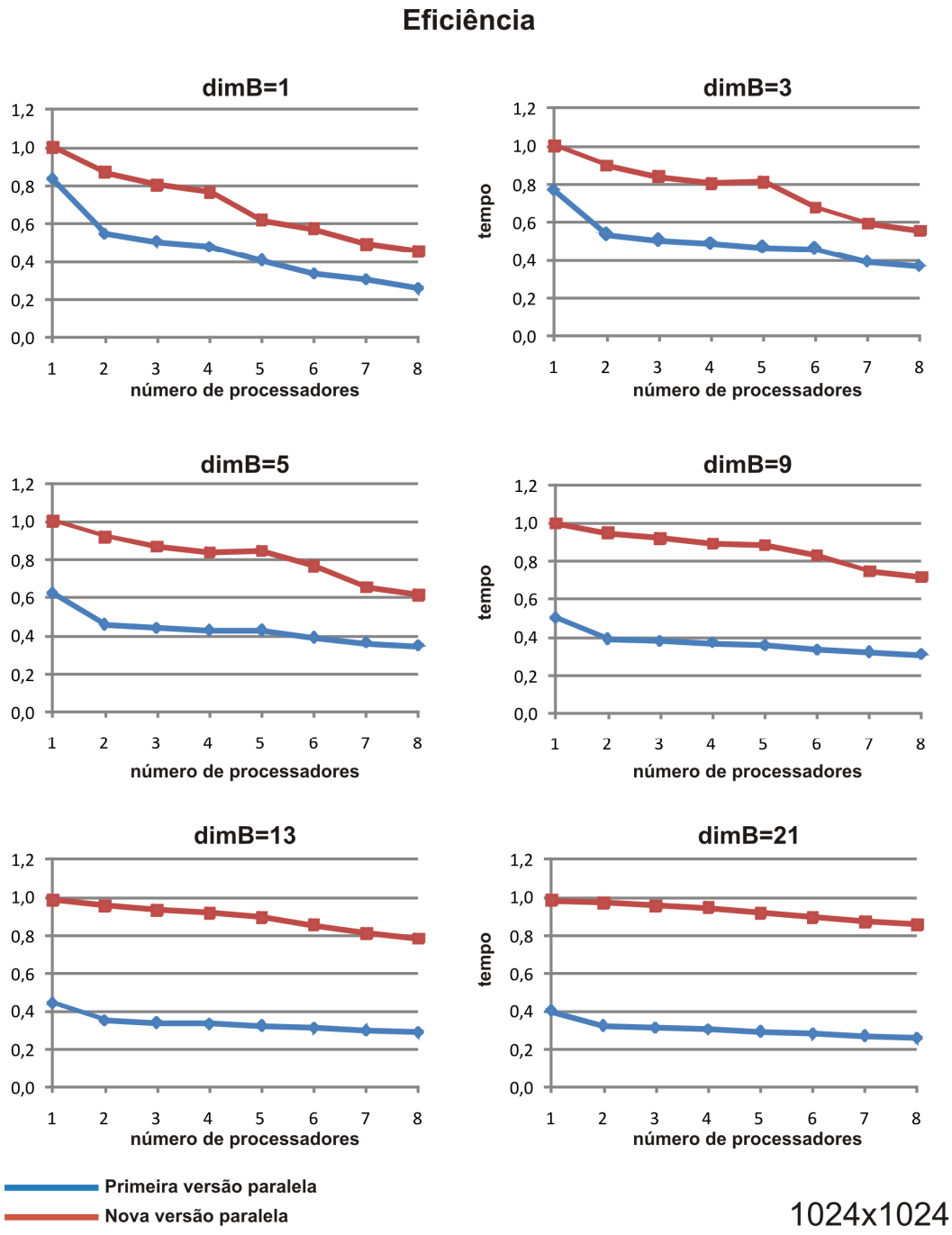


Figura 12.15 -Caso IV: 1024x1024. Eficiências.

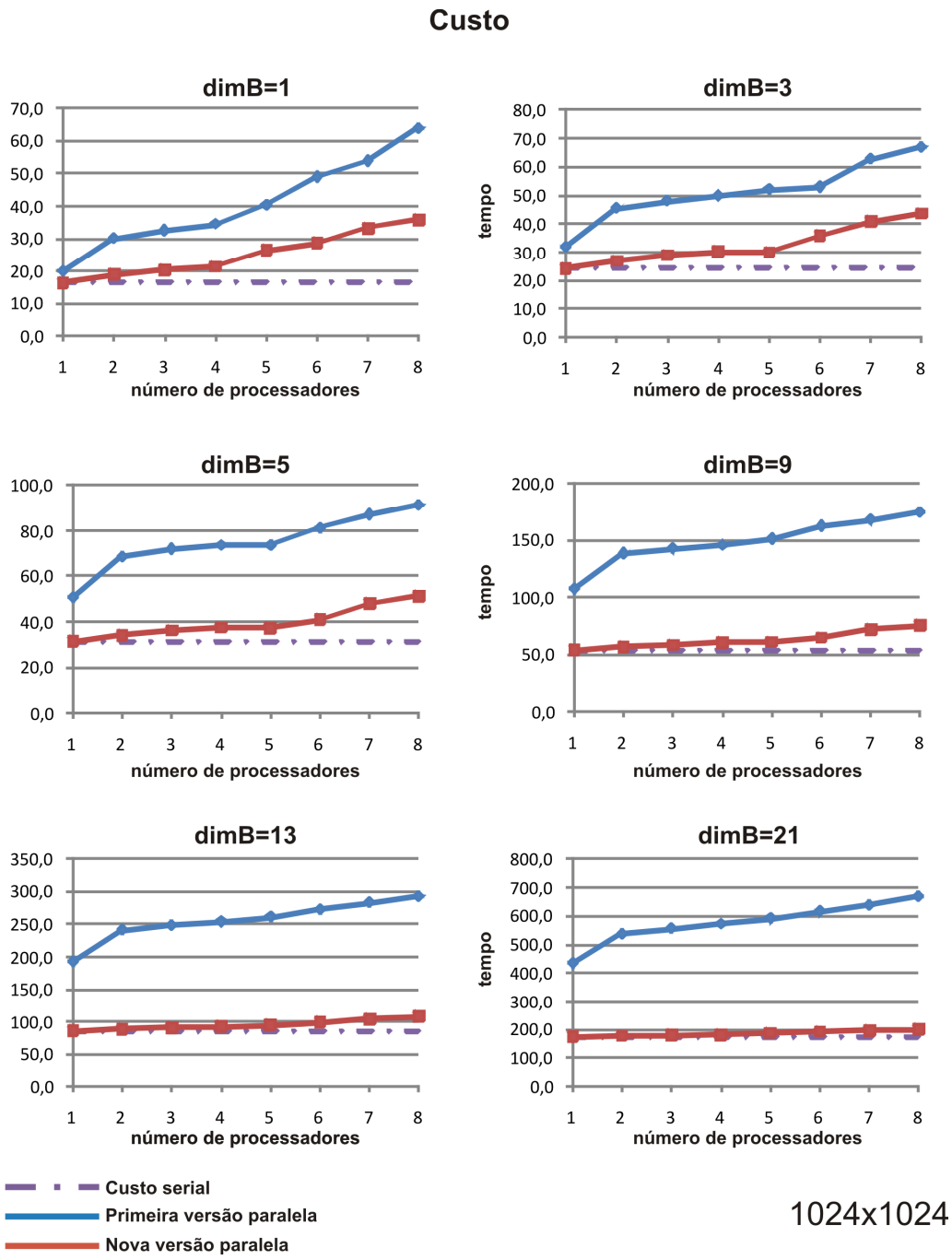


Figura 12.16 -Caso IV: 1024x1024. Custos.

- **Estudo de caso V – 2048x2048**

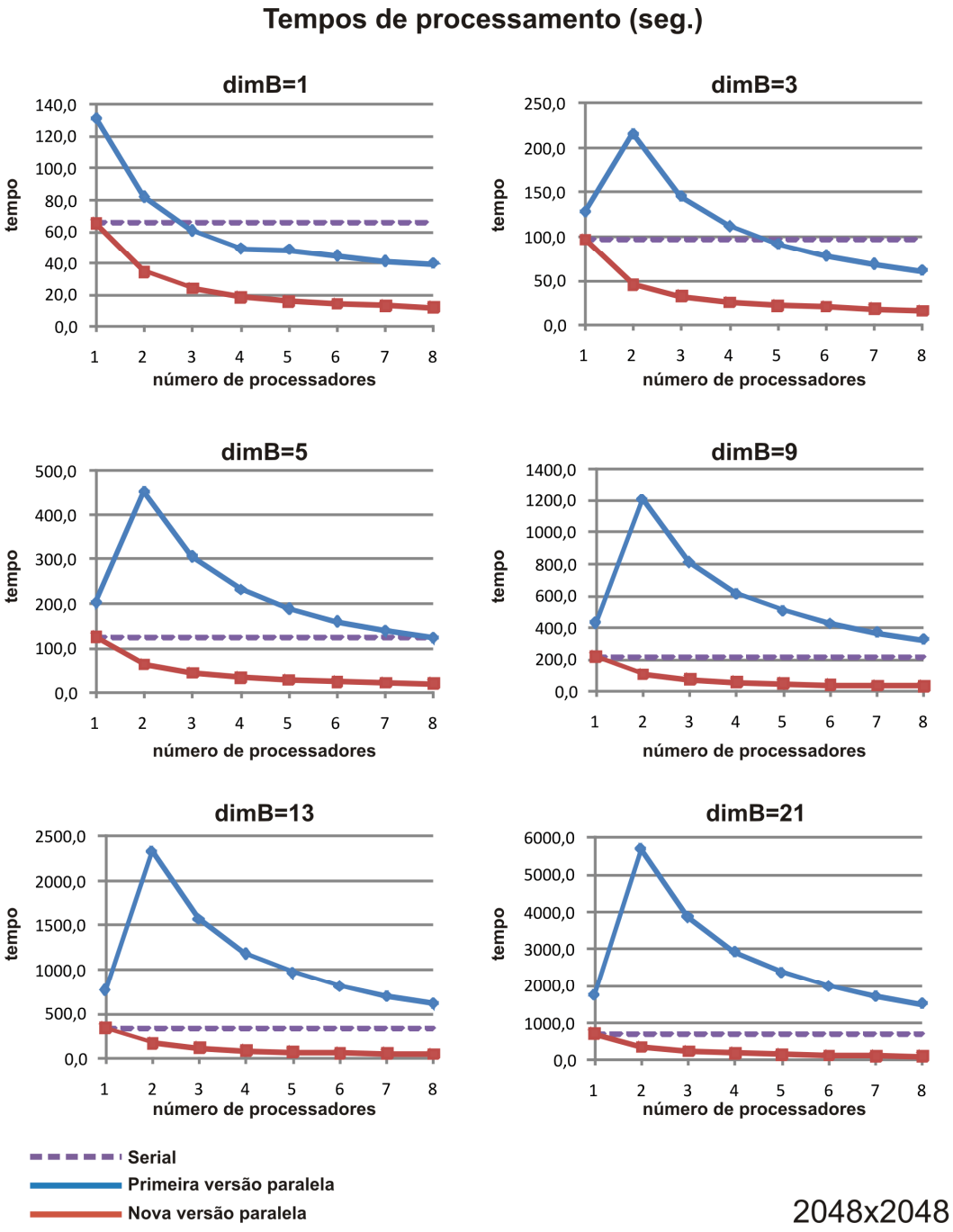


Figura 12.17 -Caso V: 2048x2048. Tempos de execução.

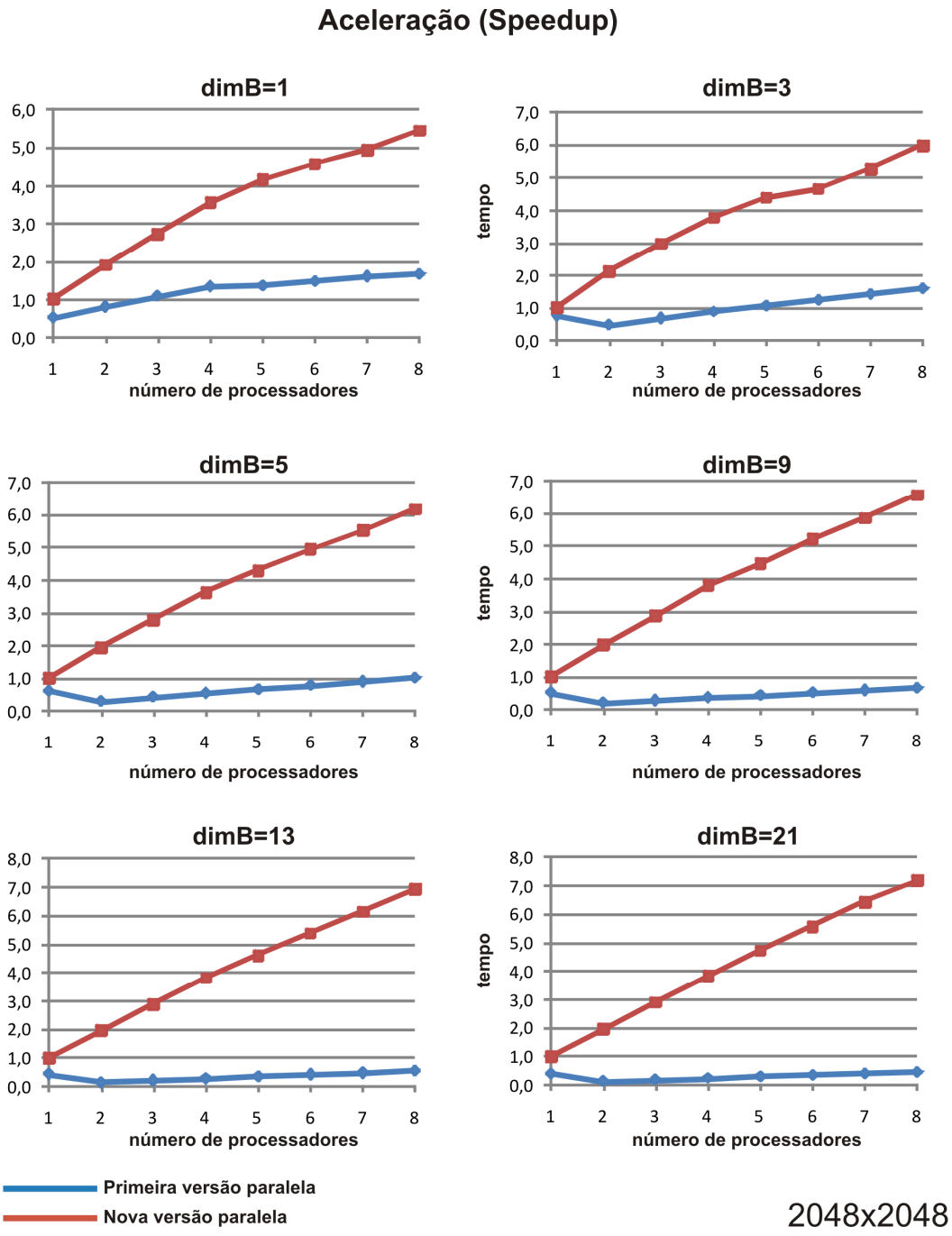


Figura 12.18 -Caso V: 2048x2048. Speedups.

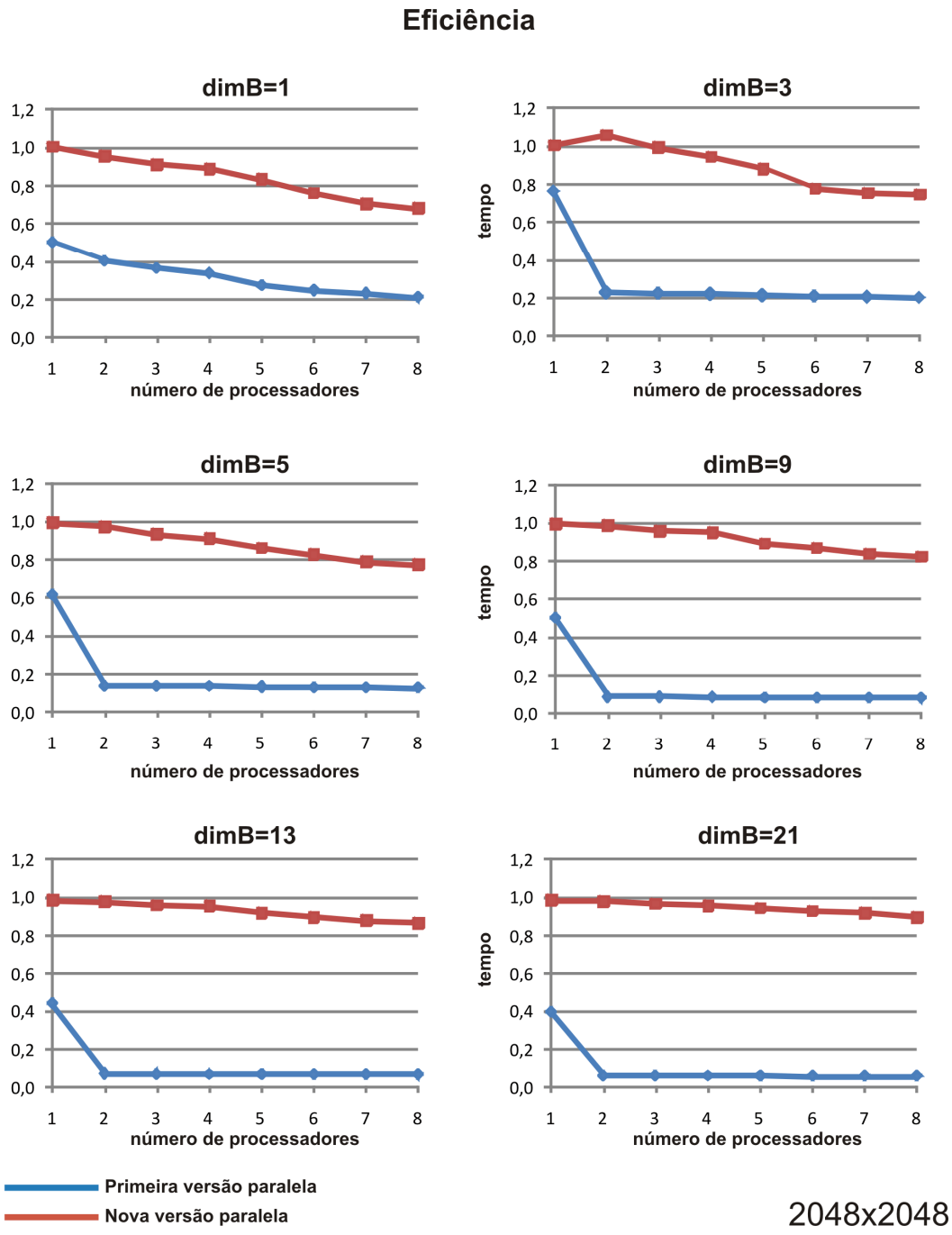


Figura 12.19 -Caso V: 2048x2048. Eficiências.

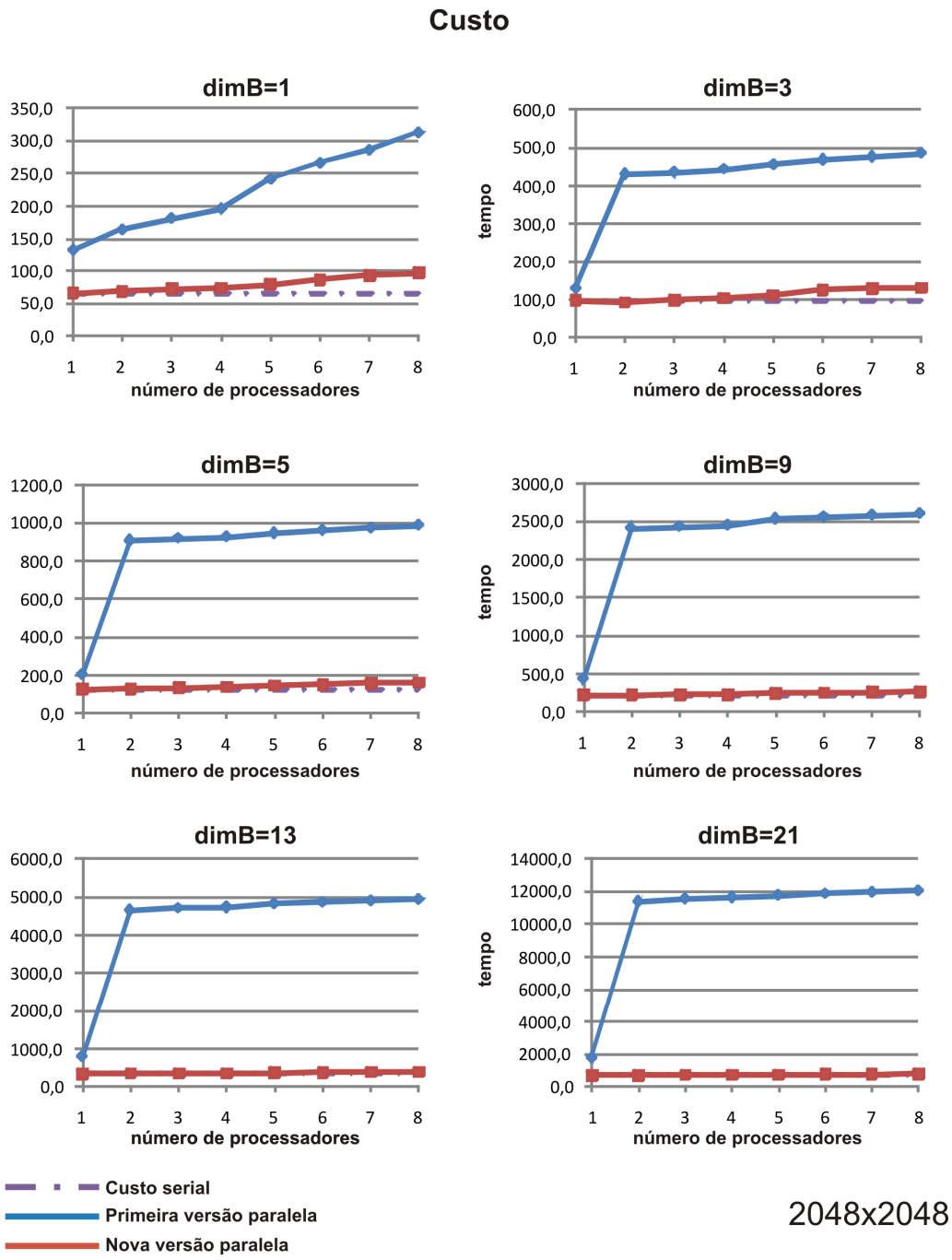


Figura 12.20 -Caso V: 2048x2048. Custos.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo