MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-8534-TDI/785

PARALELIZAÇÃO DE UM CLASSIFICADOR CONTEXTUAL DE IMAGENS

Rui Nelson Taborda de Almeida

Dissertação de Mestrado em Computação Aplicada, orientada pelo Dr. Celso Luiz Mendes, aprovada em 20 de junho de 2000.

INPE
São José dos Campos
2002

621.376.5

ALMEIDA, R. N. T.

Paralelização de um classificador contextual de imagens / R. N. T. Almeida. – São José dos Campos: INPE, 2000.

126p. – (INPE-8534-TDI/785).

1.Classificação de imagens. 2.Processamento de imagens. 3.Processamento paralelo. 4.Programação paralela. 5. Desempenho de sistemas computacionais. I. Título.

Aprovado pela Banca Examinadora em cumprimento a requisito exigido para a obtenção do Título de **Mestre** em **Computação Aplicada.**

Dr. Airam Jônatas Preto — Presidente

Dr. Celso Luiz Mendes — Orientador

Dr. Antônio Miguel Vieira Monteiro — Membro da Banca

Dr. João Argemiro de Carvalho Paiva — Membro da Banca

Dr. Cláudio Roland Sonnenburg — Membro da Banca - Convidado

Candidato (a): Rui Nelson Taborda de Almeida

São José dos Campos, 20 de junho de 2000.

AGRADECIMENTOS

Diversas pessoas contribuíram, de formas diferentes, para a realização deste trabalho. Por este motivo, agradeço:

A meus pais João e Olímpia, que durante toda a minha vida apoiaram e incentivaram todas as minhas realizações, e continuam a me ajudar a concretizar os meus diversos sonhos.

A minha irmã Katia pela motivação, apoio e paciência durante a escrita desta dissertação.

A minha noiva Lídia, pelo amor e companheirismo, que tornaram este estudo mais agradável e produtivo.

À toda a minha família, em especial a meu tio Artur, pelo constante incentivo ao cumprimento de mais esta etapa de minha vida.

Ao professor e amigo Dr. Celso Luiz Mendes, que me ajudou na proposta deste trabalho e acompanhou o seu desenvolvimento, desde os primeiros rascunhos, até a aprovação final, contribuindo com o seu valioso conhecimento.

Ao professor Guaraci J. Erthal, pela especial ajuda no desenvolvimento inicial dos programas deste trabalho.

Aos membros da banca examinadora, pela paciente análise de cada capítulo e pelas importantes sugestões e propostas de melhoramentos.

A Deus pela oportunidade, saúde e inteligência que me permitiu pertencer a este seleto grupo de pessoas, entituladas como "Mestres".

Muito Obrigado.

RESUMO

Apresentamos, inicialmente, conceitos básicos sobre processamento digital de imagens, com ênfase em classificação de imagens obtidas a partir de sensores multiespectrais. Escolhemos dois algoritmos de classificação, Máxima Verossimilhança (MAXVER) e Iterated Conditional Modes (ICM), para exemplificar a aplicação das metodologias propostas. Antes de apresentar estas metodologias, avaliamos a situação dos algoritmos mencionados em suas atuais implementações no Sistema para Processamento de Informações Georeferenciadas (SPRING). Descrevemos os aspectos relacionados a sistemas de processamento paralelo e o padrão de comunicação por troca de mensagens, Message Passing Interface (MPI). Adaptamos os programas atuais (seqüenciais) para serem executados independentemente do sistema SPRING. O objetivo desta adaptação foi otimizar os testes e a avaliação dos resultados. A partir dos códigos-fonte destes programas, desenvolvemos versões capazes de classificar imagens utilizando processamento paralelo, baseadas em troca de mensagens com MPI. O enfoque, durante este desenvolvimento, foi o de aumentar o desempenho durante a classificação das imagens com um código facilmente portável de um sistema paralelo para outro. Realizamos testes com os novos programas em equipamentos paralelos com arquiteturas diferentes entre si. Para estes testes, utilizamos tanto imagens com baixo como com grande volume de informações. Calculamos os tempos de processamento considerando aspectos tais como: algoritmo utilizado, comunicação, I/O, volume de informações, etc. Os programas paralelos foram avaliados quanto ao seu desempenho e eficiência. Comparamos as imagens geradas pelos programas paralelos com aquelas geradas pelos programas originais (seqüenciais), a fim de garantir a qualidade dos resultados. Pudemos comprovar que os processos de classificação de imagens podem ser otimizados, reduzindo o tempo de processamento consideravelmente. Além disso, os programas desenvolvidos podem ser utilizados em equipamentos paralelos com arquiteturas diferentes, sem que para isto sejam necessárias alterações nos códigos-fonte dos programas. Finalmente, concluímos que a utilização das metodologias apresentadas pode ser de grande benefício no desenvolvimento de sistemas de processamento de imagens obtidas por sensores orbitais.

PARALLEL IMPLEMENTATION OF CONTEXTUAL CLASSIFIER FOR REMOTE SENSING IMAGES

ABSTRACT

This work presents basic concepts about digital image processing, with emphasis on classification of images from multi-spectral sensors. We have chosen two classification algorithms (MAXVER and ICM), to exemplify the application of the proposed methodologies. Before showing these methodologies, we evaluated the situation of present algorithms, in the SPRING system. We described the aspects related to parallel systems and the standard of communication by Message Passing Interface (MPI). We adjusted the current programs (sequential) to be executed outside of the SPRING system. The objective was to optimize the tests and the evaluation of the results. From the code of these programs, we developed able versions to classify images using parallel processing, based on message passing interface with MPI. During the development of the programs, the objective was to increase the classification performance, using a portable code across parallel systems. The new programs were tested in parallel systems with different architectures. We used both images with low and with high volume of information. We calculated the times of processing in regard to aspects such as: selected algorithm, communication, I/O, information volume, etc. The parallel programs were evaluated in their aspects of performance and efficiency. To assess the quality of the results, we compared the resulting images of the parallel case with the resulting images of the sequential case. We confirmed that the classification can be optimized, with reduction of processing time. Furthermore, the developed programs can be used in parallel systems with different architectures, without changes in their original code. Thus, we concluded that the methodologies used in this work are very important to the development of systems for image processing.

SUMÁRIO

Pág.

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

O mundo tem passado por uma constante evolução tecnológica nos diferentes setores da ciência. Isso só foi possível com o desenvolvimento da computação. Atualmente, contamos com ferramentas computacionais para nos fornecerem uma grande variedade de informações a serem aplicadas em diversas áreas.

O Geoprocessamento é uma das áreas que têm sofrido uma grande transformação e evolução tecnológica. O objetivo do Geoprocessamento é coletar, armazenar, gerenciar, manipular e analisar informações espaciais com um determinado fim. Para que as informações referenciadas geograficamente possam ser aproveitadas, diferentes tecnologias estão envolvidas. Dentre elas, podemos destacar o processamento de imagens digitais.

De forma geral, o processamento de imagens tem sido usado em aplicações relacionadas a sensoriamento remoto, medicina, cartografia, indústria, manufatura, impressão e publicações, cosméticos e vestuário, além de uma grande quantidade de campos de pesquisa científica, incluindo astronomia, análise mineral, mecânica dos fluidos, análise radioativa, física de partículas e modelagem oceânica (Niblack, 1986). Em todos os casos é realizado um processamento computacional sobre imagens que são convertidas numa forma numérica. Estas imagens podem vir de diferentes fontes. Em áreas como medicina, teste de materiais e astronomia, as imagens podem ser obtidas de equipamentos de Raio-X, raios gama ou ultrassom. Em sensoriamento remoto, sensores em satélites produzem imagens digitais originárias de medidas de reflectância, infra-vermelho ou micro-ondas.

Para o observador, a informação em uma imagem digital, pode ser caracterizada através de propriedades dos objetos e padrões. Desta forma, a

extração da informação sobre uma imagem digital envolve a detecção e reconhecimento de padrões (Moik, 1980).

A análise de imagens da superfície da Terra começou com o uso de fotografias aéreas em meados de 1900. Câmeras de mapeamento aéreo e fotointerpretação foram ferramentas utilizadas até por volta de 1960, quando foram utilizados os primeiros sensores multi-espectrais. Paralelamente, houve um interesse no processamento quantitativo e na análise numérica dos dados obtidos destes sensores, além do desenvolvimento dos primeiros satélites da série Landsat, em 1972. Desde então, o processamento de imagens digitais tem se desenvolvido bastante.

O processamento digital de imagens é um campo fascinante devido às possibilidades que ele oferece. Entretanto, entender e analisar uma matriz de números tem sido comprovadamente uma tarefa difícil e de um custo computacional muito alto, especialmente quando consideramos imagens que representam uma grande área da superfície terrestre. A informática já contribuiu muito para otimizar este processo, mas ainda há muito a ser feito. A maioria das tarefas de extração da informação ainda requerem uma grande interatividade dos seres humanos, além do longo tempo de processamento em muitos dos computadores atuais.

O sistema visual humano tem uma extraordinária capacidade de reconhecimento de padrões. Por outro lado, nem sempre o olho humano é capaz de perceber todas as informações em uma imagem. Degradações radiométricas, distorções geométricas, e ruídos introduzidos durante a gravação, transmissão e exibição das imagens podem limitar seriamente o reconhecimento. Dentro do processamento de imagens digitais, a tarefa caracterizada como *pré-processamento*, tem como finalidade remover estas distorções e desta forma possibilitar a extração de mais informações.

Podemos destacar algumas operações básicas no tratamento digital de imagens:

- Remoção de manchas;
- Suavização de granulados e ruídos;
- Aumento de contraste ou outras propriedades visuais, para visualização;
- Aumento, redução ou rotação;
- Remoção de deformações ou distorções;
- Codificação para armazenamento ou transmissão;
- Segmentação e classificação em regiões.

As técnicas de processamento de imagens podem se dividir, basicamente, em dois grupos. O primeiro concentra a restauração quantitativa da imagem corrigindo degradação e ruído, registro de coordenadas e destaque de suas propriedades para interpretação humana. O segundo grupo se concentra na extração da informação. Esta área inclui detecção de objetos, segmentação e classificação das imagens em regiões com características diferentes, e determinação do relacionamento entre essas regiões. Neste grupo, as imagens podem ser convertidas em mapas temáticos, onde cada ponto (pixel) está associado a um tema ou classe previamente definido, sob a forma de números ou representações gráficas. Este trabalho se concentra em procedimentos deste segundo grupo, mais especificamente na classificação de imagens.

O processamento de imagens oferece vantagens quando falamos em flexibilidade na manipulação das informações. Contudo, existem desvantagens que precisam ser observadas. Uma desvantagem, é a velocidade frente ao custo. Certas operações, por geralmente serem baseadas em volumosas operações matemáticas, são extremamente lentas e necessitam de recursos computacionais mais sofisticados e eficientes. Determinados processos, antes problemáticos, têm sofrido melhoras devido aos constantes avanços da computação com tecnologia associada a um baixo custo. Operações comuns já

podem ser realizadas em computadores pessoais ou "workstations" de menor porte com grande eficiência. Porém, ainda existem outras que inviabilizam a utilização dos equipamentos mencionados, requerendo equipamentos de maiores porte e custo.

O processamento digital de imagens requer sistemas especializados que possuam funções para as diferentes operações sobre a imagem, armazenamento de dados, exibição e comunicação com o operador.

O Instituto Nacional de Pesquisas Espaciais (INPE) tem uma larga experiência no desenvolvimento de sistemas de processamento de imagens e geoprocessamento. Atualmente, existe o Sistema para Processamento de Informações Georeferenciadas (SPRING), ver *http://www.dpi.inpe.br/spring*. Este sistema pode ser executado em ambientes UNIX e Windows. O sistema SPRING, originalmente, foi desenvolvido para ser utilizado em estações de trabalho. O objetivo era desenvolver um sistema que proporcionasse eficiente integração dos dados com facilidade de uso. Alguns dos processos executados pelo SPRING são lentos, quando utilizados em computadores tradicionais. Entre esses processos, destacamos a *Classificação de Imagens*, que pode levar horas ou até dias em máquinas convencionais, dependendo do volume de informação. Mais tarde, apresentaremos uma descrição conceitual deste processo.

Como um sistema computacional, o SPRING se baseia em algoritmos para cada operação a ser realizada sobre a imagem. Este trabalho apresenta uma alternativa para otimização de dois dos algoritmos de classificação utilizados no sistema SPRING: os algoritmos de Máxima Verossimilhança (MAXVER) e *Iterated Conditional Modes* (ICM). Estes algoritmos são descritos mais adiante neste trabalho.

O tempo de processamento dos programas, muitas vezes, é limitado pelos sistemas computacionais convencionais, baseados em um processador. Mesmo que os processadores evoluam tecnologicamente, após determinados intervalos de tempo (meses, anos, ...), sempre continuarão a existir necessidades imediatas que não poderão ser resolvidas por um único processador num tempo hábil. Para resolver algumas dessas necessidades, é necessário um conjunto dos mesmos processadores trabalhando ao mesmo tempo por um objetivo comum. A este tipo de processamento, chamamos de processamento paralelo.

Neste trabalho, os algoritmos de classificação MAXVER e ICM, do SPRING, são implementados em paralelo, visando a redução do tempo de processamento a um custo relativamente baixo.

Durante anos, desenvolvedores e estudiosos do assunto se reuniram em grupos para tentar estabelecer padrões para programação em sistemas paralelos. Um desses grupos, chamado de *Message-Passing Interface Forum*, foi o responsável por estabelecer um padrão para troca de mensagens entre processadores. Eles definiram e criaram grupos de funções capazes de efetuar troca de mensagens e suportar paralelismo. Segundo Pacheco (1997), a esse grupo padronizado de funções foi dado o nome de *Message-Passing Interface* (MPI).

No desenvolvimento dos programas deste trabalho, utilizamos funções da biblioteca MPI. Mostramos com isso que com o desenvolvimento de padrões para programação em sistemas paralelos é possível portar facilmente programas, permitindo o paralelismo independentemente da arquitetura do computador.

Os programas de classificação MAXVER e ICM foram modificados a partir dos códigos originais do sistema SPRING, dispensadas todas as rotinas gráficas de

interface. Na fase inicial de desenvolvimento dos programas paralelos, foi utilizado, para testes, *cluster* de estações Sparc/SUN (rede local). Contudo, os resultados dos testes nas estações Sparc, não são apresentados neste trabalho pois o objetivo era a posterior avaliação dos programas em ambientes genuinamente paralelos. Deste modo, os programas foram testados em estações pentium Pro e equipamento SP-2 da IBM, demonstrando a viabilidade de uma implementação portável entre diferentes ambientes paralelos.

1.1 APRESENTAÇÃO DOS CAPÍTULOS

Este trabalho está organizado em seis capítulos. O primeiro corresponde a esta introdução.

No Capítulo 2 apresentamos as definições para os processos de classificação de imagens. Definimos os algoritmos trabalhados (MAXVER e ICM). Também é feita uma análise sobre como os algoritmos estão implementados atualmente no sistema SPRING e como eles se comportam. É importante realçar que para todos os diagramas de objetos, estados e fluxo de dados apresentados, utilizamos simbologia da *Object Modeling Technique* (OMT), demonstrada por Rumbaugh (1991). Apresentamos ainda conceitos sobre sistemas de processamento em paralelo. Ao final do Capítulo, abordamos a biblioteca de troca de mensagens MPI.

O Capítulo 3 descreve a metodologia utilizada para a paralelização dos algoritmos MAXVER e ICM. Apresentamos também os procedimentos para avaliação e validação das imagens classificadas.

Os resultados obtidos para as imagens e equipamentos testados podem ser observados no Capítulo 4.

No Capítulo 5 avaliamos os resultados obtidos no que diz respeito à forma como os algoritmos foram paralelizados e o problema da classificação propriamente dito. Ainda neste capítulo, avaliamos as diferenças nas imagens geradas com o processo em paralelo.

Finalmente no Capítulo 6, comentamos o conteúdo deste trabalho, fazendo uma avaliação sobre os resultados positivos e negativos. Também sugerimos pesquisas para trabalhos futuros.

CAPÍTULO 2

DESCRIÇÃO DO PROBLEMA

Neste capítulo apresentamos conceitos básicos sobre processamento de imagens, em especial os processos de classificação. Descrevemos os algoritmos de classificação utilizados neste trabalho, Máxima Verossimilhança (MAXVER) e *Iterated Conditional Modes* (ICM). O Sistema SPRING é apresentado e é feita uma análise de como estão desenvolvidos os seus módulos de classificação.

Os conceitos básicos sobre sistemas de processamento paralelo e sobre o padrão de comunicação por troca de mensagens *Message-Passing Interface* (MPI), são aqui apresentados.

2.1 CLASSIFICAÇÃO MULTI-ESPECTRAL DE IMAGENS

Uma imagem pode ser processada basicamente sob dois aspectos. O *qualitativo*, cujo objetivo principal é melhorar a qualidade da imagem para interpretação visual, e o *quantitativo*, cujo objetivo é extrair conjuntos de informações de regiões específicas da imagem (Schowengerdt, 1983).

Este trabalho enfoca uma técnica de análise quantitativa, chamada *classificação multi-espectral*. Abordamos este tipo de classificação por ser, atualmente, a mais utilizada no mapeamento de imagens de sensoriamento remoto.

Neste capítulo, portanto, é importante enfatizar alguns aspectos sobre classificação multi-espectral.

Primeiramente, é fundamental mencionar que o objetivo da classificação é mapear áreas que tenham características semelhantes. Desta forma, os pontos ou regiões de uma imagem são associados a classes de um conjunto pré-definido pelo usuário do sistema. A descrição do ponto ou região classificada é simplesmente o nome da classe a que pertence.

A saída de um processo de classificação é um tipo de imagem digital, especificamente um mapa de *pixels* (amostras ou pontos da imagem) classificados. Para apresentação, a classe de cada pixel pode ser representada por um símbolo gráfico ou por uma cor.

A informação espectral em uma cena (imagem), pode ser gravada como imagens chamadas *multi-espectrais*. Isto significa dizer que é gravado um conjunto de imagens da mesma cena, cada uma utilizando um diferente filtro espectral. Uma imagem multi-espectral pode ser representada por um conjunto de propriedades medidas ou computadas para cada banda. Uma propriedade pode ser um nível de cinza, uma medida de textura, um coeficiente de uma transformação ortogonal (ex: Fourier, Karhunen-Loéve), ou a descrição do tamanho e forma de uma região da imagem. Ao conjunto de valores para propriedades de um ponto ou região, chamamos de *padrão*.

As imagens utilizadas em sensoriamento remoto geralmente têm seus padrões analisados e classificados em áreas de plantação com as diferentes culturas, água, áreas urbanas, tipos de vegetação, tipos de solos, etc.

O processo de classificação comprime os dados da imagem, reduzindo o grande número de níveis de cinza, de cada banda, para um número menor de classes. Concluímos que a classificação causa uma redução na informação numérica, no sentido que transforma um grande número de níveis de cinza em cada banda espectral, em um pequeno número de classes em uma única imagem.

O pixel de uma imagem multi-espectral, além de suas coordenadas espaciais, que indicam sua posição na imagem, também pode ser associado a diferentes valores de acordo com o sensor de cada banda (coordenada espectral). Isto se deve ao fato de que as diferentes superfícies têm uma reflectância espectral diferente, ou ainda, variação nas características de emissão termal.

Consideramos, portanto, que o valor de um pixel *p* pode ser representado por um vetor *v*.

$$v = [v_1, v_2, ..., v_n] \tag{2.1}$$

onde:

v_i = valor de *p* na *i-ésima* banda espectral.

Ao conjunto de valores espectrais dos pixels amostrados nas diferentes bandas da imagem (considerando-os como identificadores de uma mesma classe), chamamos de *assinatura espectral*.

A assinatura espectral de uma classe pode ser obtida a partir do conjunto de amostras, considerando a variabilidade espectral possível, para uma mesma classe.

Dado o conjunto de padrões de uma imagem e baseado nas assinaturas espectrais das classes, os diferentes algoritmos de classificação empregam teorias de decisão estatísticas ou técnicas geométricas para associar um determinado padrão a uma classe específica. Para o método estatístico, as classes são definidas através de parâmetros de distribuição de probabilidade. No método geométrico, as classes são representadas por coeficientes de funções discriminantes.

A caracterização das assinaturas das classes de interesse do usuário, é um aspecto crítico para o sucesso do processo de classificação e deve ocupar uma boa parte do tempo de análise. Para que um programa seja capaz de classificar uma imagem, ele deve dispor de um conjunto de pixels (amostras) para cada classe. Existem, basicamente, dois caminhos para determinar as amostras e consequentemente as assinaturas das classes de interesse: o treinamento supervisionado e o não supervisionado. Neste trabalho, nos dedicamos especialmente à metodologia de classificação supervisionada, pois é empregada nos algoritmos de classificação estudados. Quanto à classificação não supervisionada, apresentamos apenas uma descrição superficial do seu funcionamento.

2.1.1 Classificação Supervisionada

Para o treinamento supervisionado, o usuário utiliza conhecimento prévio obtido em trabalho de campo, fotointerpretação e outras fontes, para associar amostras de pixels às classes de interesse. As assinaturas dessas classes são calculadas pelo programa e utilizadas para reconhecer pixels com assinaturas semelhantes em toda a imagem.

A classificação supervisionada tem sido o procedimento mais utilizado na análise quantitativa dos dados de sensoriamento remoto. Para tal, existe uma grande variedade de algoritmos que geralmente se baseiam em modelos de distribuição de probabilidade para as classes de interesse.

Independentemente do algoritmo escolhido, o usuário deve seguir alguns passos básicos para executar este tipo de classificação.

Primeiramente, o usuário deve definir as classes em que a imagem poderá ser classificada. Por exemplo: água, áreas urbanas, regiões de plantações, campos de pastagem e etc.

Em seguida, devem ser selecionadas áreas representativas da imagem para cada classe a ser localizada. É importante que cada área selecionada possua amostras de regiões homogêneas da classe em questão. Entretanto, as variações para esta mesma classe também devem ser amostradas. Normalmente, mais de uma área de treinamento é selecionada para cada classe. Para auxiliar o processo de identificação das amostras, são utilizados resultados de trabalhos de campo, fotografias aéreas, mapas da região e etc.

Em muitos casos, as características da imagem não permitem a nítida identificação de suas classes. Neste caso, devem ser utilizadas algumas técnicas de realce de regiões a fim de identificar melhor as suas assinaturas.

2.1.2 Classificação Não-Supervisionada

No treinamento não supervisionado, o usuário utiliza um algoritmo que localiza ocorrências de concentrações numa amostra heterogênea de pixels. Essas concentrações, chamadas *clusters*, são assumidas como representações de classes e são utilizadas para o cálculo das assinaturas das classes. O usuário, entretanto, pode decidir que clusters correspondem às suas classes de interesse ou não.

As duas técnicas (supervisionada e não-supervisionada) podem ser utilizadas como complementares. Um fator que determina a utilização de uma ou outra é o conhecimento do usuário sobre a região.

Os dados de treinamento são utilizados pelos algoritmos de classificação como parâmetros para o cálculo da probabilidade de cada classe no espaço de amostras. Uma vez definidas as probabilidades das classes e consequentemente suas assinaturas espectrais, os algoritmos de classificação devem ser capazes de gerar relatórios tabulares ou mapas temáticos representando o resultado da classificação.

Os processos de classificação ainda podem ser subdivididos em dois grandes grupos:

- Pixel a pixel: A classificação de um certo pixel baseia-se apenas na sua própria informação espectral;
- Contextual: Neste caso, a classificação de um pixel leva em conta não somente o seu valor espectral, mas também os valores dos seus vizinhos.

Neste trabalho, abordamos dois métodos de classificação presentes no sistema SPRING. O primeiro deles é o da Máxima Verossimilhança (MAXVER). Trata-se de um método *supervisionado* do tipo *pixel a pixel,* e que atua de forma estatística sobre a imagem. O outro é chamado de Iterated Conditional Modes (ICM), que é um método *supervisionado* e *contextual*, baseado num processo iterativo onde a classe de cada pixel vai sendo atribuída, a cada iteração, em função do valor espectral do pixel e das classes de pixels vizinhos.

2.2 CLASSIFICAÇÃO MAXVER

O método de classificação chamado Máxima Verossimilhança, ou somente MAXVER, é o mais comumente utilizado na classificação de imagens de sensoriamento remoto. É um método estatístico, onde cada classe é modelada segundo uma distribuição *gaussiana*. O critério empregado para classificação

baseia-se na regra de decisão de Bayes. A maioria dos sistemas de processamento de imagens implementa esta metodologia (Richards, 1993).

Para descrever este método, vamos começar com um exemplo. Suponha que nós temos uma banda de uma imagem de uma região costeira, contendo terra e água. Nós precisamos classificar cada pixel como “classe 1= terra” ou “classe 2= água”. O resultado será um mapa de classes, ou imagem, onde cada pixel está associado a um valor, 1 ou 2, de acordo com a sua classificação (terra ou água).

Para associar cada pixel a uma das classes, precisamos de uma “regra de decisão”. Por exemplo:

- Associe todos os pixels da metade esquerda da imagem como classe *água*;
- Associe todos os pixels da metade direita da imagem como classe *terra*.

Este é um exemplo de uma regra trivial, arbitrária e sem uma aparente justificativa. Podemos definir uma outra regra onde o valor do pixel é considerado independente da área onde está localizado. Por exemplo:

- Associe todos os pixels que possuem um valor menor que 20 a *água*. Associe os demais a *terra*.

Esta é uma regra simplesmente baseada num limiar. O problema está no momento em que o número de classes e de bandas aumenta. Neste caso, é necessária a definição de regras adequadas ao volume de informações.

O método *Bayesiano* é uma forma de estabelecer regras de decisão. Um classificador baseado neste método é do tipo supervisionado.

As amostras de treinamento servem para estabelecer a distribuição de probabilidade de cada uma das classes. Cada classe W_k é modelada por uma função densidade de probabilidade $p(X|W_k)$ do tipo *gaussiana*. Um determinado pixel da imagem com valor espectral $X=X_0$ deve ser associado à classe mais provável, isto é, àquela que maximize $P(W_k|X_0)$, que é a probabilidade de que a classe de X_0 seja W_k. Intuitivamente, cada classe está associada a uma curva de distribuição de probabilidade; dependendo da posição espectral do pixel em relação a estas curvas, é determinada a classe a que ele pertence. Segundo a regra de Bayes:

$$P(W_k \mid X) = \frac{P(X \mid W_k).P(W_k)}{P(X)} \qquad (2.2)$$

O valor $P(X)$ é independente das classes, logo, basta maximizar o numerador de (2.2). Supondo-se que, inicialmente, as classes tenham igual probabilidade, ou seja, $P(W_1) = P(W_2) = ... = P(W_j)$, temos que o único termo a ser maximizado é $P(X|W_k)$, que, segundo o modelo *gaussiano* tem a seguinte expressão:

$$P(X \mid W_k) = \frac{1}{(2\pi)^{n/2} |C_k|^{1/2}} \bullet \exp\left(-\frac{1}{2} (X - M_k)^t C_k (X - M_k) \right) \qquad (2.3)$$

onde:

- $\boldsymbol{n}$ é o número de atributos do pixel (número de bandas espectrais da imagem);
- $\boldsymbol{M_k}$ é o vetor-média da classe W_k;
- $\boldsymbol{C_k}$ é a matriz de covariância da classe W_k.

A expressão em (2.3) é maximizada quando minimiza-se o termo:

$$\exp\left(-\frac{1}{2}(X-M_k)^t C_k (X-M_k)\right) \quad \Leftrightarrow \quad \ln|C_k| + R_k^{\,2} \tag{2.4}$$

onde $R_k^2 = (X - \boldsymbol{M_k})^t\, \boldsymbol{C_k}\, (X-\boldsymbol{M_k})$ é o quadrado da "distância de Mahalanobis" do pixel de valor X à média $\boldsymbol{M_k}$, relativa à k-ésima classe.

Embora, em princípio, a regra de decisão apresentada indique que se deva associar o pixel de valor X à classe W_k tal que ($ln\ |\boldsymbol{C_k}| + R_k^2$) seja minimizado, é possível que, para um determinado pixel, esta expressão ainda atinja valores altos para todas as classes. Intuitivamente, tal pixel está longe do ponto médio de todas as classes (em termos da distância de Mahalanobis). Nestes casos, é comum utilizar-se limiares de decisão L_k tal que, se para todas as classes W_k ocorrer:

$$\left(\ln|C_k| + R_k^{\,2}\right) > L_k \tag{2.5}$$

então o pixel não é classificado, ou é dito rejeitado, podendo ser associado a uma classe especial W_0.

Os valores de $\boldsymbol{M_k}$ e $\boldsymbol{C_k}$ para cada classe são obtidos numa etapa preliminar, denominada fase de treinamento, onde o usuário indica ao sistema pixels representativos de cada classe (amostras). Para G amostras em uma determinada classe W_k, tem-se:

$$M_k = \frac{1}{G}\sum_{i=1}^{G} Xi \tag{2.6}$$

$$C_k = \frac{1}{G}\sum_{i=1}^{G}(X_i - M_k)\bullet(X_i - M_k)^t \tag{2.7}$$

De acordo com o comentado anteriormente, a classificação de um pixel com valor *X* consiste em procurar pela classe W_k que minimize a expressão (2.4). Sendo o valor $ln|\boldsymbol{C_k}|$ uma constante para cada classe, e portanto independente do valor do pixel em questão, basta calcular-se o termo contendo R_k, e em seguida somá-lo a $ln|\boldsymbol{C_k}|$, que já pode ter sido calculado e armazenado previamente.

Existem duas possibilidades de cálculo do termo R_k. A primeira vem de sua própria definição e é dada por:

$$R_k^{\,2} = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \bullet (x_i - m_i) \bullet (x_j - m_{ij}) \qquad (2.8)$$

onde:

- *n* é o número de atributos do pixel (número de bandas espectrais da imagem);
- x_i é o nível de cinza do pixel de valor *X* relativo ao i-ésimo atributo (i-ésima coordenada de *X*);
- m_i é a i-ésima coordenada do vetor-média $\boldsymbol{M_k}$ relativo à k-ésima classe;
- c_{ij} é o elemento (i,j) de $\boldsymbol{C_k^{-1}}$.

A outra possibilidade é através da transformação de coordenadas no espaço de atributos, pela qual pode-se fazer com que a nova matriz de covariância seja igual à matriz identidade, o que irá reduzir a distância de Mahalanobis à distância euclidiana (Velasco, 1978). Esta transformação é tal que:

$$X' = X - \mathbf{M_k}, \qquad X'' = \mathbf{T_k} \,.\, X' \qquad (2.9)$$

Com esta transformação, R_k passa a ser dado por:

$$R_k^{\,2} = X''_1{}^2 + X''_2{}^2 + \ldots + X''_n{}^2 \qquad (2.10)$$

É possível encontrar uma transformação tal que T_k seja triangular inferior, fazendo

$$\mathbf{C_k} = L_k \,.\, L_k^t, \qquad \mathbf{T_k} = L_k^{-1} \qquad (2.11)$$

O pixel transformado passa a ter, então, as seguintes coordenadas:

$$\begin{aligned} x'_1 &= t_{11} \,.\, (x_1 - m_1) \qquad (2.12)\\ x'_2 &= t_{21} \,.\, (x_1 - m_1) + t_{22} \,.\, (x_2 - m_2)\\ &\ldots\\ x'_n &= t_{n1} \,.\, (x_1 - m_1) + \ldots + t_{nn} \,.\, (x_n - m_n) \end{aligned}$$

Pelas expressões acima, observa-se que o custo computacional do cálculo de x''_1 é menor que o de x''_2, que o de x''_2 é menor que o de x''_3, e assim por diante. O valor R_k também pode ser dado pela expressão geral:

$$R_k^{\,2} = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} t_{ij} \bullet \left(x_j - m_j \right) \right)^2 \qquad (2.13)$$

Se durante o cálculo de R_k^2 ocorrer, para determinado atributo i, que o valor acumulado em R_k^2 ultrapasse o limiar de decisão L_k, então o cálculo pode ser interrompido, com a implicação de que o pixel atual é rejeitado pela classe W_k. Desta forma, o tempo de processamento pode variar de um pixel para outro, dependendo da quantidade de cálculos necessários à classificação de cada um deles.

O resultado deste processo é uma imagem classificada, na qual cada pixel da imagem original está associado a uma das classes W_k (ou à classe genérica W_0). Uma avaliação simples da qualidade deste processo de classificação pode ser realizada através da verificação das classes atribuídas aos diversos pixels selecionados como amostras na fase de treinamento.

2.3 CLASSIFICAÇÃO ICM

Novos modelos para classificação de imagens têm sido propostos. Estes novos modelos, chamados *contextuais*, levam em consideração a dependência espacial entre as classes de uma imagem. Isto quer dizer que um pixel é classificado através de método estatístico que considera os seus pixels vizinhos.

Entretanto, o uso destes novos modelos implica em um custo computacional superior aos modelos de classificação pixel a pixel. Contudo, o aumento do custo computacional é justificável, pois geralmente estas metodologias tendem a apresentar resultados mais homogêneos e muitas vezes são utilizadas para correção de imagens com ruídos (pixels não identificados corretamente no momento da aquisição da imagem).

Segundo Orgambide (1993), um dos principais métodos de classificação contextual é o *Iterated Conditional Modes* (ICM). Ele implementa uma técnica que pode levar a uma maior precisão na classificação, pois permite incorporar informações de dependência espacial dos pixels observados dentro de cada classe. Outra grande vantagem do algoritmo ICM sobre outros classificadores contextuais é o fato de ter a sua formulação baseada no método de classificação MAXVER, que é muito difundido.

Por definição, o algoritmo ICM é determinístico. Isto quer dizer que para um mesmo dado, sempre é produzido o mesmo resultado.

O algoritmo ICM pode ser definido como: dada a estimação da imagem *x* na iteração *k*, representada por $\chi(k)$, a nova estimação, $\chi(k+1)$, deve ser gerada atualizando, para toda coordenada *s,* o valor do pixel x_s pelo valor χ_s que maximiza a expressão:

$$P\left(x_s \middle| y, x_{S\setminus\{s\}}\right) \tag{2.14}$$

Este processo se repete até ser encontrado um equilíbrio ou convergência.

O suporte de uma imagem pode ser denotado como *S*. Por definição $S \subset Z^2$ é finito e da forma $S = S_1 * S_2$, onde $S_1=\{1,...,m\}$ e $S_2 = \{1,...,n\}$; intuitivamente, S é o conjunto de posições onde a imagem tem seus pontos definidos. A função indicadora de um conjunto *A* pode ser denotada $F_A(x)$, onde:

$$F_A(x) = \begin{Bmatrix} 1 & se & x & \in & A \\ 0 & caso & contrário & & \end{Bmatrix} \tag{2.15}$$

Conforme Georgii (1988), o vetor aleatório $X=[X_s]_{s \in S}$ é chamado de modelo de Potts-Strauss, se tem por contra-domínio, cada sítio $s \in S$, os valores {1,...,*K*}, $K \geq 2$, e se a sua distribuição conjunta é:

$$P(X = x) = \frac{1}{Z_b}\exp\left(\beta \sum_{\|s-u\|=1} F_{\{x_s\}}(x_u) \right), \tag{2.16}$$

com $\beta \in \Re$, $Z_b = \sum_{x \in q} \exp\left\{\sum_{\|s-u\|=1} F_{x_s}(x_u)\right\}$, e o conjunto das $K^{\#S}$ configurações possíveis $\theta = \{1,...,K\}^S$.

Adotando este modelo como distribuição para o algoritmo ICM, o método consiste em se substituir, na iteração t, χ_s (t) pela classe ℓ, que satisfaça $\max^{-1}_{\ell\in\{1,\ldots,K\}}\{f_\ell(y_s)\bullet \boldsymbol{b}\, v_s(\ell)\}$, onde $v_s(\ell)=\#\{u\in\partial_s : x_u=\ell\}$ e $\partial_s=\{u\in S:\|s-u\|=1\}$. Ao conjunto ∂_s, chamamos de *vizinhança* do ponto *s*, e os seus elementos *vizinhos* do ponto *s*.

Este algoritmo ainda é alvo de muitas pesquisas e vários detalhes não estão completamente especificados. Detalhes deste algoritmo, como sequência de pixels visitados, estimação inicial $\chi(0)$ e possíveis critérios de parada, além de conexão das ICM com técnicas de relaxação estocástica e resultados experimentais, podem ser encontradas em Besag (1986, 1989).

Para classificação com o algoritmo ICM, utilizamos os seguintes dados de entrada:

- bandas originais da imagem a ser classificada;
- imagem temática da região, gerada a partir de um classificador pontual (utilizamos o MAXVER);
- limiar de classificação;
- porcentagem de pixels modificados a cada iteração (critério de parada).

A seguir, apresentamos uma descrição simplificada da implementação deste algoritmo:

1) Lê bandas originais da imagem, banda temática (gerada por MAXVER) e parâmetros de entrada;

2) A cada iteração *m*, os pixels da imagem são visitados baseando-se numa subdivisão em quatro partições (conhecido como método de Gauss-Seidel). A figura 2.1 exemplifica o percorrimento nas subdivisões, em uma região de 6x6 pixels. Os números correspondem à sequência em que os pixels são visitados.

1	**10**	**2**	**11**	**3**	**12**
19	**28**	**20**	**29**	**21**	**30**
4	**13**	**5**	**14**	**6**	**15**
22	**31**	**23**	**32**	**24**	**33**
7	**16**	**8**	**17**	**9**	**18**
25	**34**	**26**	**35**	**27**	**36**

Fig. 2.1 – Grades e sequência de visitas a pixels da imagem para classificação.

3) Para cada imagem classificada ***A***, resultante da iteração *m-1*, a iteração *m* gera uma nova imagem classificada ***B*** tal que:

$$v_{i,j}(k) = \beta \cdot f(a_{i-1,j} ; a_{i,j-1} ; a_{i,j+1} ; a_{i+1,j} ; k) \qquad (2.17)$$

$b_{i,j} = k$ tal que: $\left(\ln|C_k| + R_k^{\,2}\right) - v_{i,j}(k)$ é *mínimo* (2.18)

onde:

- *k* é a classe analisada;
- *f* é uma função de maioria, que retorna o número de pixels vizinhos (assumindo vizinhança 4) do pixel a_{ij}, com classe igual a *k*;
- ***b*** é o parâmetro de atratividade, que incide sobre o valor retornado pela função *f*;
- $v_{ij}(k)$ é o valor correspondente à informação contextual de uma classe *k*, para o pixel observado na coordenada (*i,j*) em *A*;
- $a_{i,j}$ é a classe atribuída a um pixel com coordenadas (*i,j*) em ***A***;

- $b_{i,j}$ é a nova classe atribuída a um pixel com coordenadas (i,j) em ***B.*** Esta imagem será tratada como "A" em uma nova iteração.

OBS: Em (2.18) o termo $\left(\ln|C_k| + R_k^{\,2}\right)$, é obtido do algoritmo de classificação MAXVER, que é base para a classificação ICM.

O parâmetro de atratividade $\boldsymbol{b}$, pode ser fixo (definido no início da execução do algoritmo) ou ser estimado a cada iteração *m* (Orgambide, 1993). Em caso afirmativo, $\boldsymbol{b}$ pode ser obtido através da equação de pseudo-verossimilhança (Jensen e Moller, 1989). O valor de $\boldsymbol{b}$ é utilizado para ponderar o retorno da função de maioria *f*. Desta forma, é inserida a informação contextual à classificação, como podemos observar em (2.18). Se o valor de $\boldsymbol{b}$ for "zero", concluímos que o algoritmo torna-se pontual.

4) A cada iteração *m*, após classificação de todos os pixels, verifica-se o percentual de pixels modificados em relação a *m-1*. Se o número de pixels for menor ou igual ao percentual definido como critério de parada, encerra execução. Caso contrário, executa nova iteração *m+1*. O processo ainda pode ser encerrado caso seja ultrapassado o número máximo de iterações, previamente estabelecido.

O número de iterações necessárias em uma classificação ICM é variável. Ele está relacionado aos dados de cada imagem e ao número de classes definidas.

Originalmente, o algoritmo ICM foi desenvolvido para processamento seqüencial. Sua utilização para classificação de imagens é uma opção viável, porém ainda requer um custo computacional alto, demorando muito tempo para um grande volume de dados. Este trabalho propõe adaptação do algoritmo para processamento em paralelo (conforme será mostrado no Capítulo 3). A

execução do algoritmo em paralelo visa reduzir significativamente o tempo de processamento e com isso garantir uma maior utilização deste tipo de classificação, sem que seja necessário um grande investimento computacional.

2.4 ESQUEMA DE CLASSIFICAÇÃO UTILIZADO NO SPRING

2.4.1 O Sistema SPRING

Durante vários anos, os pesquisadores do instituto Nacional de Pesquisas Espaciais (INPE) têm se empenhado no desenvolvimento de aplicações para Geoprocessamento e Processamento de Imagens. O “Sistema para Processamento de Informações Georeferenciadas” (SPRING) é um dos resultados do trabalho iniciado na década de 80.

O SPRING é um sistema que possui um banco de dados orientado a objetos para o gerenciamento de informações geográficas. Além disso, ele incorpora ferramentas de processamento de imagens. Tudo isto com uma interface gráfica amigável e interativa, baseada em *X Window System* com padrão de apresentação OSF/MOTIF (SPRING, 1998).

Os principais objetivos do sistema SPRING, conforme a própria documentação do sistema, são:

- Integrar as tecnologias de Sensoriamento Remoto e Sistemas de Informação Geográfica;
- Utilizar modelo de dados orientado a objetos, que melhor reflete a metodologia de trabalho de estudos ambientais e cadastrais;
- Fornecer ao usuário um ambiente interativo para visualizar, manipular e editar imagens e dados geográficos.

O SPRING foi desenvolvido utilizando uma abordagem orientada a objetos. A linguagem de desenvolvimento escolhida foi o C++ para ambiente operacional UNIX e, na sua versão inicial, só podia ser executado em estações de trabalho com arquitetura RISC. Com a evolução dos *Personal Computers* (PCs) e a crescente demanda para este tipo de sistemas, o SPRING foi migrado para ambiente Windows, a fim de atender um maior número de usuários a um custo menor.

Neste trabalho consideramos somente a versão UNIX do SPRING. Isto foi feito porque, atualmente, os padrões de comunicação paralela estão mais difundidos e conseqüentemente mais testados nestes ambientes. Desta forma, podemos garantir uma melhor avaliação dos resultados e das técnicas empregadas. Entretanto, os conceitos abordados neste trabalho certamente poderão ser incorporados a trabalhos futuros envolvendo outras plataformas operacionais.

2.4.2 Módulo de Classificação

No SPRING, classes de objetos tais como imagens, polígonos, linhas e textos, são modelados em C++. Com isso o sistema permite que o conceito de herança seja empregado entre os diferentes objetos. O conceito de herança em sistemas orientados a objetos permite que características existentes numa determinada classe sejam reaproveitadas em classes derivadas.

O módulo de classificação de imagens, no SPRING, é composto por uma hierarquia de classes. O diagrama que representa o relacionamento entre as classes do módulo de classificação pode ser observado no apêndice A deste trabalho.

2.4.3 Versões para Testes

Para que os algoritmos de classificação MAXVER e ICM utilizados pelo SPRING fossem melhor avaliados, optamos pela separação dos mesmos em relação ao resto do produto. Desta forma, geramos dois aplicativos isolados, que podem ser executados fora do ambiente gráfico original, em forma de linha de comando.

Cada um dos aplicativos é responsável por uma das classificações estudadas (MAXVER e ICM). Estes aplicativos são inteiramente baseados nos códigos originais utilizados no SPRING, diferindo somente no que diz respeito à não utilização de linguagem C++ e bibliotecas para recursos gráficos, mas apenas a linguagem C padrão. Isto foi necessário para manter a compatibilidade com as bibliotecas MPI instaladas nos equipamentos de testes.

Foram criadas estruturas de dados para substituir as classes existentes no código C++ original. As estruturas de dados criadas correspondem às classes:

- Image;
- Matrix;
- MyClass;
- TrainingData.

Os aplicativos adotam o modelo Gaussiano multivariado para caracterização das classes espectrais. Neste modelo não foi prevista a existência de áreas não classificadas; na prática, isto pode ser obtido com a utilização de um limiar de classificação convenientemente escolhido. Também consideramos que as classes são equiprováveis (Orgambide, 1993).

Os aplicativos para testes são executados por linha de comando e utilizam os seguintes dados de entrada:

1) Conjunto de bandas armazenadas em disco no formato RAW (arranjo linear em disco obtido pelo encadeamento dos pixels da imagem obtidos ao longo de cada linha da esquerda para a direita e percorrendo as linha de cima para baixo). Consideramos um arquivo de imagem para cada banda espectral;
2) O Classificador ICM utiliza, além das bandas originais da imagem, a banda temática gerada pelo classificador MAXVER;
3) Arquivo com dados de treinamento das classes (formato SPRING), de onde são extraídos o vetor média e a matriz de covariância de cada classe;
4) Arquivo com controles de convergência dos algoritmos:

   - Limiar (MAXVER e ICM);
   - Porcentagem de pixels modificados (ICM).

Como saída dos aplicativos, temos:

1) Arquivo de imagem temática (classificada), no formato RAW;
2) Arquivo texto (testeset.dat) contendo os parâmetros das classes obtidos no arquivo de treinamento.

O código destes aplicativos é totalmente independente da estrutura de dados utilizada no SPRING, exceto pela dependência do arquivo de dados de treinamento que, para este trabalho, continua a ser gerado no próprio sistema SPRING.

Todas as áreas de dados utilizadas nos programas foram inicialmente alocadas estaticamente. Para imagens pequenas não houve problema, entretanto, quando utilizamos imagens maiores (ex: Cena TM-LANDSAT), foi necessária a alteração para alocação dinâmica de memória.

Basicamente, os dois aplicativos utilizam uma mesma função para montagem das matrizes de transformação a serem usadas no cálculo da *Distância de Mahalanobis*. Eles diferem no que diz respeito à função chamada para a classificação propriamente dita: um chama a função para classificação MAXVER e o outro para ICM.

Os diagramas simplificados dos modelos de objetos, dinâmico e funcional, para os elementos envolvidos no processo de classificação MAXVER e ICM podem ser observados no apêndice A deste trabalho.

2.5 SISTEMAS PARALELOS

Quando observamos que a constante necessidade de maior eficiência em processamento computacional não pode ser satisfeita somente por um processador durante um determinado tempo, torna-se fácil aceitar que a tendência da computação, em geral, é a utilização de sistemas paralelos. Isto ocorre porque é necessário um período de tempo para desenvolvimento de um processador mais rápido e eficiente, e nem sempre é possível esperar.

Para satisfazer algumas das necessidades imediatas que ainda não podem ser resolvidas em tempo hábil por somente um processador, é necessário um conjunto dos mesmos processadores trabalhando por um objetivo comum (Dowd e Severance, 1998).

O processo de classificação de imagens não é diferente. Com mais recursos de aquisição e armazenamento, quanto mais informações puderem ser processadas simultaneamente, em intervalos de tempo aceitáveis, melhor.

Ao modificar os algoritmos de classificação apresentados, adaptando-os para sistemas paralelos e utilizando um padrão de comunicação entre processadores, desejamos otimizar e avaliar o processo, além de permitir a sua

execução em diferentes arquiteturas paralelas, sem que isto implique em grandes modificações no código original. Para tanto, é importante apresentarmos alguns conceitos e características básicas de sistemas paralelos.

2.5.1 Características Básicas

Um computador paralelo é simplesmente um computador, ou uma coleção destes, com múltiplos processadores que podem trabalhar juntos para solucionar um mesmo problema. Para que isto ocorra, é necessário considerar alguns aspectos:

1) Decidir e implementar a rede de interconexão entre os processadores e módulos de memória;
2) Projetar e implementar os sistemas de software e hardware;
3) Desenvolver algoritmos e estruturas de dados para resolver o problema;
4) Dividir os algoritmos e estruturas de dados em sub-problemas;
5) Identificar a comunicação que será necessária entre os subproblemas;
6) Associar os subproblemas aos processadores e módulos de memória.

Com estes seis itens definidos, é possível que um sistema seja executado em paralelo para resolver um problema comum (Pacheco, 1997).

2.5.2 Sistemas Paralelos de Memória Compartilhada e Distribuída

Os sistemas paralelos são subdivididos, de acordo com a forma de acesso aos seus módulos de memória, em sistemas de memória compartilhada (multiprocessadores) e de memória distribuída (multicomputadores). Nos sistemas paralelos de memória compartilhada, os processadores e os módulos

de memória são interconectados por uma rede, conforme ilustrado na figura 2.2. Nesta classe de equipamentos, cada processador pode acessar qualquer módulo de memória diretamente. Para minimizar o tráfego no acesso às diversas memórias, é comum o uso de memórias *cache* junto a cada processador.

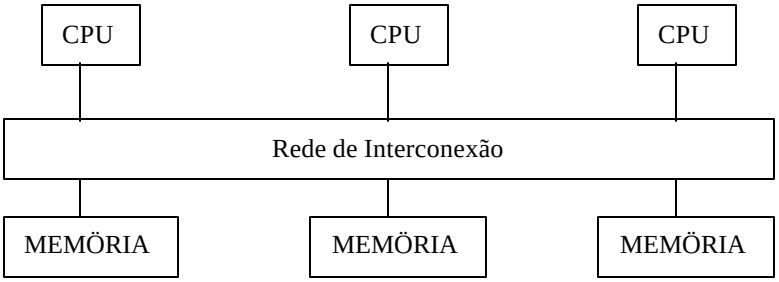


Fig. 2.2 - Esquema genérico para sistemas paralelos de memória compartilhada.

Em sistemas paralelos de memória distribuída, cada processador tem seu próprio módulo de memória, e somente este processador pode acessá-la diretamente. O acesso aos dados que estão na memória de um processador remoto precisa ser feito através de uma troca de mensagens entre os processadores envolvidos. Por exemplo, se o processador "A" precisa ler um dado que está na memória do processador "B", é preciso ocorrer uma troca de mensagem tal que "B" leia o dado de sua memória, envie-o através de uma mensagem para o processador "A", e que o processador "A" receba tal mensagem, utilizando-se então do dado recebido.

Um esquema genérico de sistemas de memória distribuída pode ser representado pela figura 2.3.

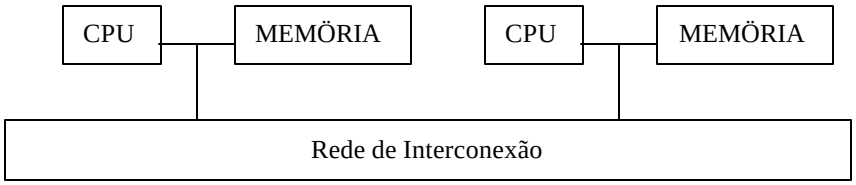


Fig. 2.3 - Esquema genérico para sistemas paralelos de memória distribuída.

A interconexão ideal em sistemas paralelos é aquela em que cada nó (conjunto processador + memória) possui comunicação direta com todos os outros:

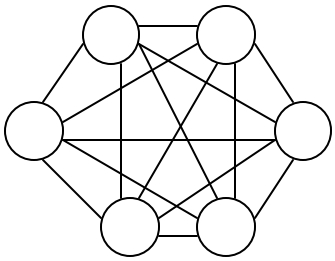


Fig. 2.4 - Modelo genérico de interconexão ideal para sistemas paralelos.

Como podemos observar na figura 2.4, a comunicação nesta arquitetura não envolve nenhum tipo de retardo, pois qualquer nó pode se comunicar com qualquer outro ao mesmo tempo que outras comunicações estão ocorrendo. Porém, seria economicamente proibitivo construir uma máquina com estas características e com um número considerável de processadores. Na prática, os sistemas possuem um grau bem menor de conectividade entre os vários nós. As redes de interconexão reais podem variar desde um simples barramento, comum em multiprocessadores, até sofisticadas redes de múltiplos estágios.

2.6 TROCA DE MENSAGENS UTILIZANDO MPI

O aumento do uso de programas paralelos tem sido incentivado pelo desenvolvimento de padrões de programação. Os desenvolvedores podem escrever códigos portáveis entre os diferentes equipamentos e esperar um retorno para o desenvolvimento de grandes aplicações paralelas.

Até bem pouco tempo atrás, era necessário aprender uma nova linguagem de programação, ou conhecer os detalhes de outro *hardware*, quando necessitávamos migrar um programa paralelo para outro equipamento. Entretanto, este aspecto vem sendo superado graças ao esforço de vários grupos de desenvolvedores e pesquisadores em estabelecer padrões para a programação de sistemas paralelos.

Um destes grupos, chamado de *High Performance Fortran Forum,* desenvolveu um conjunto de extensões para a linguagem Fortran 90, facilitando a confecção de programas baseados no processamento dos dados em paralelo. Neste caso, o paralelismo ocorre no sentido em que os dados (vetores ou matrizes) são distribuídos pelos processadores e estes executam o mesmo trabalho para as suas porções de dados. Tal conjunto de extensões definiu uma nova linguagem, denominada *High Performance Fortran*, ou apenas HPF. Esta linguagem fornece várias primitivas para a distribuição destes dados e, consequentemente, para o seu processamento em paralelo. É uma linguagem muito poderosa e bem projetada. Entretanto, diversos algoritmos ainda não podem ser convenientemente implementados utilizando-se os compiladores HPF atuais, devido à pouca maturidade destes compiladores (Loveman, 1993).

Um segundo grupo, chamado de *Message-Passing Interface Forum,* pregava a adoção de um padrão para a comunicação por troca de mensagens. Era uma abordagem diferente na busca por um padrão para programar sistemas paralelos. Ao invés de especificar uma nova linguagem, optou-se por

especificar uma biblioteca de funções que pudessem ser chamadas por programas escritos em C ou Fortran. O objetivo era formar um pequeno grupo de funções, para troca de mensagens, que pudessem ser usadas para suportar paralelismo. Tal grupo padronizado de funções recebeu a denominação de *Message-Passing Interface* (MPI).

O método de programação através de troca de mensagens é um método muito poderoso e abrangente para conseguir paralelismo. Esta técnica permite criar programas paralelos mais eficientes, pois o programador tem a capacidade de estabelecer explicitamente como deve ocorrer a comunicação entre os vários processadores, e pode, com isto, otimizar a execução do programa.

Em contrapartida, esta metodologia também acarreta uma maior dificuldade para o desenvolvimento de programas paralelos, pois o programador fica totalmente responsável pela correta inserção das funções de comunicação em locais apropriados do código-fonte. Muitos chamam MPI de “Assembly da computação paralela”. Contudo, a história tem mostrado que sofisticados programas utilizando troca de mensagens podem ser desenvolvidos rapidamente (Pacheco, 1997). A cada dia que passa, mais sistemas utilizando MPI são criados. Mais e mais algoritmos são encapsulados em bibliotecas baseadas em MPI, e este encapsulamento significa ter um programa onde podemos simplesmente chamar uma função para executar um algoritmo em paralelo (Gropp, 1994).

Outra grande vantagem do uso de MPI é o aspecto de portabilidade. Um determinado código-fonte, com comunicação baseada em MPI, pode ser executado em qualquer sistema no qual aquele padrão seja suportado. Atualmente, quase todos os fabricantes de sistemas paralelos suportam tal padrão.

Pelos motivos apresentados, as versões paralelas dos algoritmos apresentados neste trabalho foram desenvolvidas utilizando a metodologia de troca de mensagens com MPI. O objetivo era tornar os programas mais eficientes e portáveis, satisfazendo o objetivo da avaliação em equipamentos com arquiteturas diferentes.

2.7 RESUMO

Neste capítulo, vimos que imagens podem ser processadas sob dois aspectos, o qualitativo e o quantitativo, que é onde está o enfoque deste trabalho. Observamos o processo de mapeamento de áreas que tenham características semelhantes, chamado de classificação. Neste trabalho levamos em consideração a classificação de imagens multi-espectrais, obtidas por sensores orbitais. Analisamos as características das imagens multi-espectrais relevantes ao processo de classificação. Apresentamos ainda as características e diferenças básicas entre as metodologias de classificação supervisionadas e não-supervisionadas, além da caracterização do que é uma classificação *pixel a pixel* e *contextual*.

Observamos a descrição do algoritmo de classificação MAXVER, que é o mais comumente utilizado na classificação de imagens de sensoriamento remoto. Em seguida, pudemos observar a descrição do algoritmo de classificação contextual ICM, onde concluímos que, sendo um algoritmo estatístico com maior complexidade, implica num maior custo computacional, porém com resultados mais satisfatórios. Foi apresentada a situação atual dos módulos de classificação (MAXVER e ICM) do sistema SPRING.

Ainda neste capítulo, pudemos observar que a necessidade de um processamento computacional mais poderoso, num curto espaço de tempo, nos leva a buscar soluções alternativas, como o processamento em paralelo.

Neste ponto, vimos que a classificação de imagens pode ser processada em paralelo. Apresentamos os conceitos básicos de arquiteturas de sistemas paralelos, incluindo a definição de sistemas paralelos de memória compartilhada e distribuída. Comentamos, ainda, sobre o sistema de comunicação por troca de mensagens, MPI. Este é um padrão de comunicação que permite que programas sejam desenvolvidos para diferentes arquiteturas paralelas, com o mínimo de alteração possível.

No próximo capítulo, serão apresentados os esquemas de paralelização utilizados para os algoritmos de classificação MAXVER e ICM. Será feita uma análise de como os programas originais do sistemas SPRING foram modificados e adaptados para o processamento paralelo. Além disso, apresentaremos a metodologia para avaliação das imagens resultantes do processo de classificação.

CAPÍTULO 3

ABORDAGEM DE PARALELIZAÇÃO DO CLASSIFICADOR

Este capítulo inicia-se com uma abordagem dos conceitos de programação paralela implementados e comuns aos dois programas de classificação (MAXVER e ICM). Em seguida, são levantados os aspectos específicos no desenvolvimento de cada um dos programas.

Finalizando, são apresentadas as metodologias utilizadas para verificação da correção da imagem gerada e os procedimentos para avaliação de desempenho e eficiência dos programas de classificação em paralelo.

3.1 ESQUEMA GERAL DE PARALELIZAÇÃO

Em sistemas com suporte a MPI, os processadores coordenam suas atividades, enviando e recebendo mensagens. Isto é o básico que uma biblioteca MPI possui (uma função que envie mensagens e outra que as receba). A versão de MPI utilizada neste trabalho assume que os processadores são alocados estaticamente. Isto significa que o número de processadores envolvidos na execução dos programas é definido no início do processamento, permanecendo fixo até o final, sem que processadores adicionais sejam inseridos.

Ao se desenvolver um programa paralelo, geralmente temos como estratégia diminuir o tempo de execução, aproveitando ao máximo os recursos dos processadores. Em algumas situações, isto talvez não signifique distribuir o mesmo volume de dados para cada processador (pois o tempo de processamento por dado pode ser variável); nestes casos, devemos procurar alguma outra forma de distribuição de dados tal que os “tempos de

processamento” dos vários processadores sejam similares, evitando, assim, processadores ociosos.

Uma vez que decidimos utilizar o modo de programação por troca de mensagens com MPI, tornou-se necessário definir quais técnicas, dentro deste padrão, seriam mais ou menos eficientes para os algoritmos em questão.

Para simplificar os programas, decidimos que somente um processador deve ser o responsável pela interação com o usuário; isto diz respeito à entrada de dados via teclado e saída no terminal padrão. Este processador também é o responsável pela leitura e escrita de informações das imagens em disco. Tal processador é o responsável pela distribuição dos dados aos demais processadores. O motivo para atribuir estas funções a somente um processador é garantir a *portabilidade*, por não haver, ainda, um padrão muito bem definido para I/O (*Input/Output*) em sistemas paralelos.

Para a definição das técnicas de paralelização empregadas, alguns aspectos foram considerados. Primeiramente, levamos em conta a questão dos sistemas de memória distribuída e compartilhada. Os programas desenvolvidos podem ser utilizados em qualquer um dos ambientes de memória, pois estão baseados no padrão MPI; contudo, isto não quer dizer que o tempo de processamento seja similar para os dois ambientes. Este tempo varia consideravelmente de um ambiente para outro, especialmente no caso da classificação ICM, onde a comunicação é mais intensa.

Outro fator que influencia o desempenho dos sistemas, é a forma como a memória *cache* dos equipamentos é utilizada. Por exemplo, no MAXVER, cada linha da imagem é processada uma única vez, e descartada em seguida. Neste caso não há muito aproveitamento dos *caches*. Entretanto, no caso do ICM, cada linha da imagem lida para a memória, é utilizada durante o processamento de três linhas de saída: a própria linha, a linha acima e a linha

abaixo (exemplo: supondo que a linha original é a de número 9, ela será usada para gerar os valores de saída das linhas 8, 9 e 10, com isto os dados são reutilizados duas vezes e há vantagem se for possível mantê-los no *cache*).

Nos sistemas de memória compartilhada, a forma mais comum de programação consiste em especificar quais variáveis são privativas de cada um dos processadores e quais são compartilhadas por todos eles. Em geral, a comunicação entre processadores é feita através do acesso às variáveis compartilhadas. O acesso ordenado a estas variáveis é obtido através de um conjunto de funções de sincronização. Sistemas de memória compartilhada, contudo, também podem ser programados segundo um paradigma de memória distribuída, com comunicação baseada em troca de mensagens. Neste caso, as informações de uma certa mensagem não precisam ser deslocadas, mas apenas acessadas por seus endereços iniciais de memória. Este processo é bem mais rápido do que enviar uma mensagem de uma região da memória de um processador para outro em sistemas de memória distribuída.

Atualmente, na maioria dos sistemas paralelos de memória distribuída, o tempo gasto no envio de uma mensagem pode ser dividido em duas partes: uma parte fixa, caracterizada pelo hardware e pelo software básico do sistema, e uma parte variável, proporcional ao comprimento da mensagem. Em mensagens curtas, o custo da parte fixa tende a preencher amplamente o tempo de comunicação.

O ideal para os programas desenvolvidos neste trabalho seria o processamento das imagens inteiramente em memória. Entretanto, quando consideramos imagens grandes, como cenas inteiras TM-Landsat, a memória não é suficiente para todas as bandas. Neste caso é necessário um particionamento da imagem para o processamento.

Nos programas desenvolvidos neste trabalho, a forma como a imagem é dividida tem grande influência no tempo de processamento, especialmente no caso do ICM, devido à troca de informação entre processadores para manter a classificação *contextual*. Segundo Bader (1995), se a imagem fosse dividida em blocos regulares (quadrados), cada processador precisaria trocar mensagens com até 8 processadores vizinhos (relativos aos blocos acima, abaixo, à esquerda, à direita e diagonais). Por outro lado, dividindo a imagem em blocos retangulares, com cada bloco cobrindo toda a sua dimensão horizontal, a necessidade de comunicação cai para somente dois processadores vizinhos, no máximo (um bloco acima e outro abaixo). É verdade que o volume de informação a ser transportado por vez é maior, mas, como na prática o tempo de comunicação depende mais do número de mensagens do que do tamanho de cada uma, esta metodologia tende a obter um desempenho significativamente maior do que com o esquema de divisão em blocos regulares. Logo, para obter maior desempenho e atender as limitações de memória, optamos por desenvolver os algoritmos baseados na divisão da imagem em faixas retangulares.

Outro fator importante a ser ressaltado é a forma como o trabalho é dividido entre os processadores. Devemos, portanto, entender a diferença entre divisão estática e dinâmica de trabalho.

3.1.1 Divisão Estática/Fixa de Trabalho

A principal característica para este tipo de divisão consiste em atribuir um volume de trabalho fixo para todos os processadores. O volume de trabalho atribuído aos diferentes processadores não precisa necessariamente ser igual para que a divisão seja considerada estática. Esta atribuição é feita logo no início do processamento e permanece constante até o final.

Considerando o processo de classificação, devemos verificar se esta é uma solução viável, pois pode sobrecarregar algum processador, já que o tempo de processamento pode variar de um pixel para outro. Com isto, o tempo de processamento desta implementação pode variar para diferentes regiões da imagem. O gráfico da figura 3.1 permite a visualização de um exemplo da problemática envolvida na divisão estática.

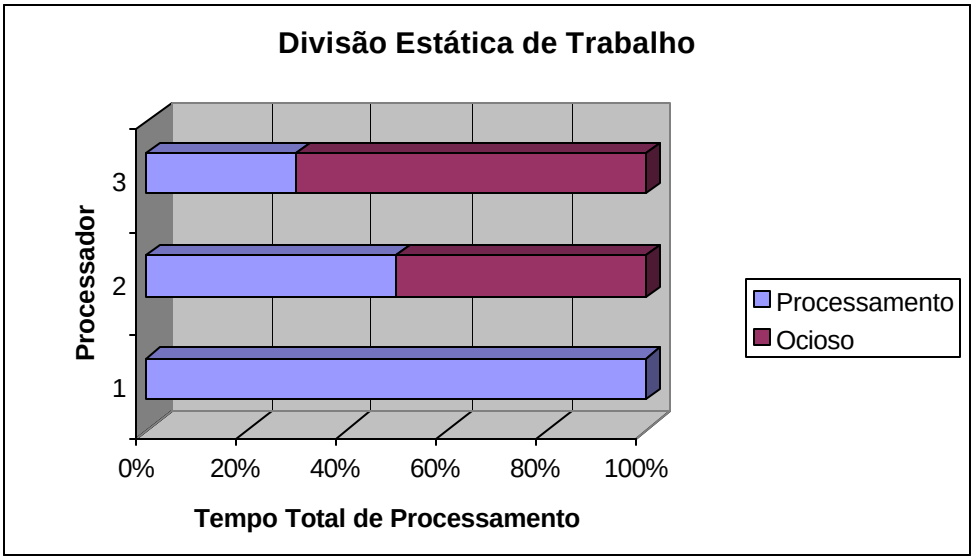


Fig. 3.1 - Divisão estática de trabalho entre processadores.

Esta técnica, entretanto, facilita o desenvolvimento dos programas, mas somente pode ter sua eficiência garantida em ambientes onde há certeza do mesmo tempo de processamento em todos os processadores envolvidos.

3.1.2 Divisão Dinâmica de Trabalho

Este tipo de divisão consiste em particionar os dados (no caso a imagem) em pequenas porções, as quais não precisam ser distribuídas de uma só vez. Ao contrário, um subconjunto delas é distribuído inicialmente pelos processadores; e assim que algum processador termina a sua parte, ele está automaticamente disponível para receber outra porção e continuar o trabalho. Para isto, é

necessária uma estrutura de controle central para ser consultada e permitir ao processador identificar a próxima porção da imagem a ser trabalhada. Programando desta maneira, todos os processadores são aproveitados sem sobrecargas ou ociosidade. O gráfico da figura 3.2 permite a visualização de um exemplo desta situação.

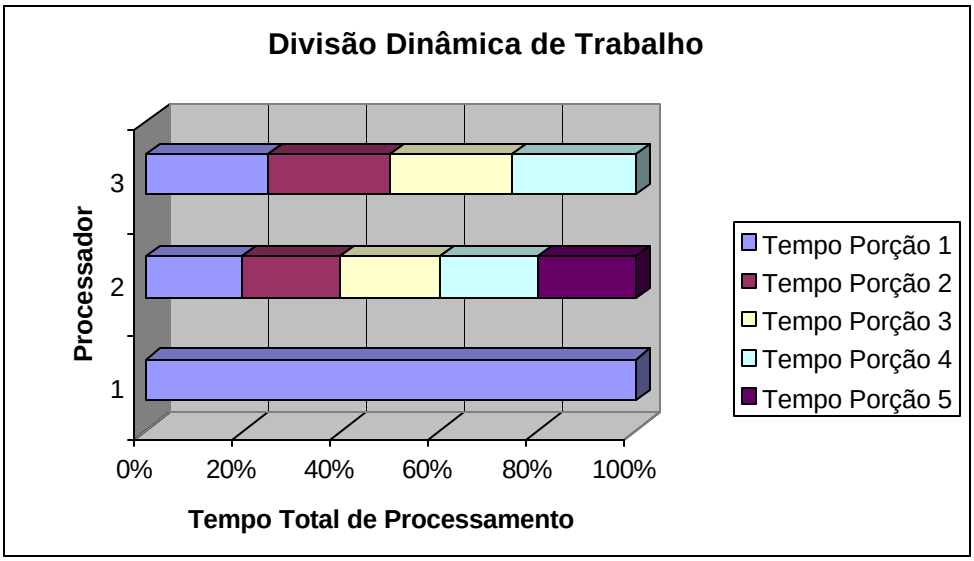


Fig. 3.2 - Divisão dinâmica de trabalho entre processadores.

Pelo gráfico mostrado, podemos observar que, enquanto um processador leva mais tempo para processar a sua porção da imagem, os outros vão adiantando o trabalho de acordo com as respectivas oportunidades de processamento. Em geral, nenhum processador fica ocioso até o final da execução do programa.

Neste trabalho, iniciamos o desenvolvimento dos algoritmos paralelos utilizando divisão estática de trabalho. Entretanto, é importante ressaltar que para cada caso, foi medido o grau de ociosidade dos processadores, ou seja, o desequilíbrio de carga entre eles. Estas informações contribuíram para a análise do desenvolvimento com divisão dinâmica de trabalho.

3.2 ESQUEMA DE PARALELIZAÇÃO PARA O PROGRAMA DE CLASSIFICAÇÃO MAXVER

As estruturas de dados definidas no programa MAXVER seqüencial permanecem as mesmas na versão paralela. Apenas foram adicionadas variáveis de controle para o processamento com vários processadores, além das rotinas de comunicação entre eles.

Como a metodologia de classificação MAXVER atua pixel a pixel, sem levar em consideração a informação contextual, houve uma maior flexibilidade na escolha das técnicas de paralelização empregadas. Isto ocorreu porque os pixels podem ser processados independentemente, fazendo com que cada processador possa trabalhar sobre um bloco bem definido da imagem, independentemente dos demais.

Assumimos o processador de índice 0, como o responsável pelos procedimentos de entrada e saída de dados. Este processador lê os parâmetros de classificação definidos pelo usuário, além dos dados de treinamento e as bandas da imagem em questão e distribui tais valores para os outros processadores.

Em uma abordagem inicial da paralelização do algoritmo de classificação MAXVER, as imagens lidas seriam divididas em *P* faixas retangulares, onde *P* corresponderia ao número de processadores envolvidos na classificação. Desta forma cada processador classificaria a sua faixa de imagem, como pode ser observado na figura 3.3.

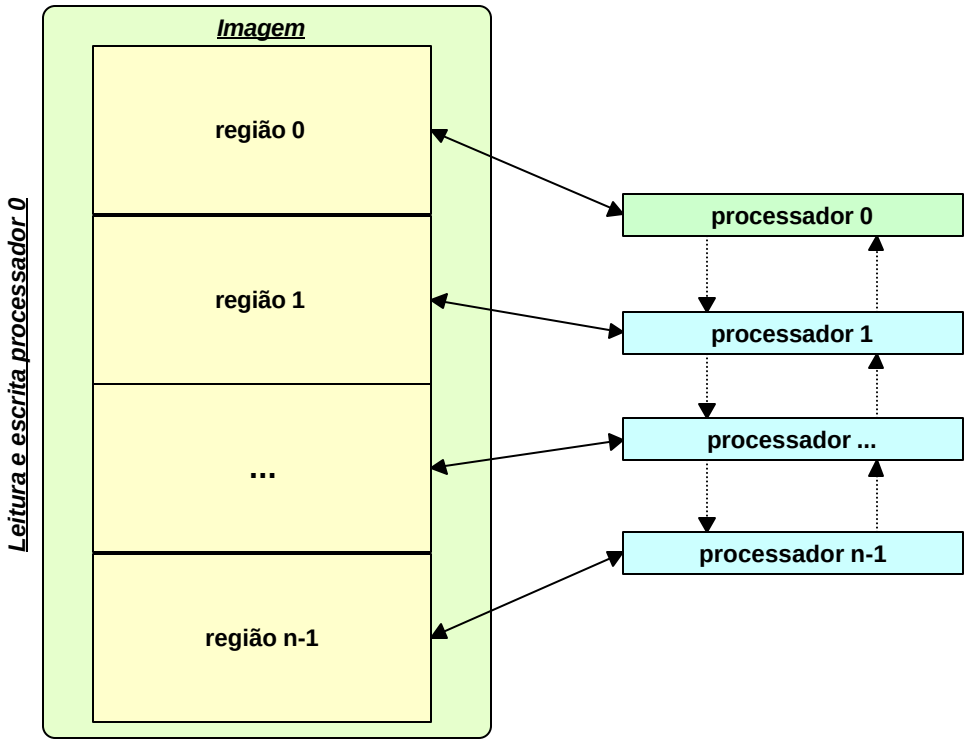


Fig. 3.3 - Divisão trivial das imagens entre os processadores.

Esta abordagem seria satisfatória para um volume de informação pequeno. Porém, se tomarmos como exemplo 3 bandas de uma cena inteira (6400 linhas x 7000 colunas por banda) de uma imagem Landsat real, podemos verificar que esta abordagem não serviria. Isto é fácil de se observar ao assumirmos que, para a imagem dada como exemplo, nós temos um volume total de informação correspondente a 179.200KB (considerando a imagem temática resultante e cada pixel ocupando 1 byte). Se considerarmos 4 processadores, cada um deveria trabalhar com um total de 44.800KB. Entretanto, um dos equipamentos utilizados neste trabalho (com memória compartilhada), possuía apenas 128MB de memória. Isto nos leva a crer que, em termos práticos, nem sempre é economicamente viável adicionar grande quantidade de memória aos sistemas, dado que o custo de memória é significativo frente ao custo total dos equipamentos.

A utilização dos dados em disco resolveria esta questão, porém as operações de *"swap"* entre memória/disco poderiam comprometer seriamente o desempenho desta implementação. Foi o que observamos com alguns testes iniciais. Os tempos obtidos utilizando esta abordagem foram extremamente altos, o que inviabilizou o processo neste sentido.

A solução encontrada para tal problema foi dividir as imagens em faixas retangulares, menores. Desta forma, todos os processadores trabalham juntos sobre uma mesma faixa, por vez, com um volume de informação reduzido. Nesta abordagem, cada processador tem a capacidade de classificar os dados somente em memória, reduzindo o acesso a discos e consequentemente reduzindo o tempo de execução.

Nesta nova abordagem, ao ler os dados, o processador 0 considera a imagem dividida em *n* faixas retangulares. Cada faixa lida pelo processador 0 é sub-dividida de acordo com o número de processadores envolvidos (consideramos este número sempre par) e as sub-faixas resultantes desta subdivisão são distribuídas para os processadores, sendo uma sub-faixa para cada um. A figura 3.4 exemplifica o esquema de divisão considerando a utilização de quatro processadores.

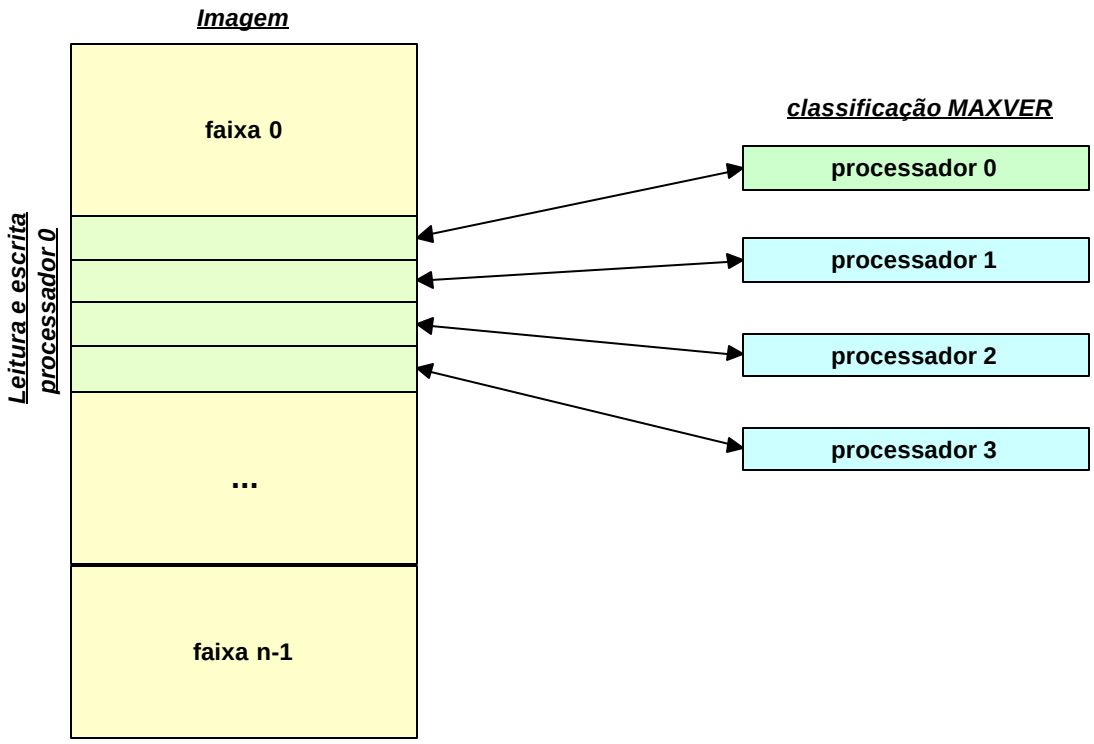


Fig. 3.4 - Divisão da imagem em faixas retangulares para 4 processadores (classificação MAXVER).

Cada processador, após receber a sua porção de imagem, realiza a classificação MAXVER (para esta porção) seguindo o algoritmo original utilizado no sistema SPRING. Feito isto, a sub-faixa da imagem é enviada novamente para o processador 0, que é o responsável pelo ordenamento e salvamento dos dados classificados em disco.

Observamos que os tempos gastos pelos vários processadores no processo de classificação eram praticamente constantes entre eles (ver Capítulo 4 deste trabalho). Por este motivo, consideramos apenas a divisão estática de trabalho, pois a divisão dinâmica não traria nenhum benefício extra.

Os modelos simplificados de objetos, dinâmico e funcional para os elementos envolvidos no processo de classificação MAXVER em paralelo, são apresentados no apêndice B deste trabalho.

3.3 ESQUEMA DE PARALELIZAÇÃO PARA O PROGRAMA DE CLASSIFICAÇÃO ICM

O programa paralelo de classificação ICM segue, basicamente, os mesmos princípios utilizados no MAXVER. Neste ponto nos referimos ao processador responsável por I/O e às operações realizadas por ele, além das estruturas de dados utilizadas e a forma como a imagem é particionada para classificação. Entretanto, outras características devem ser observadas.

Em relação à divisão de trabalho, concluímos que, da mesma forma que o classificador MAXVER, o ICM não necessita ser implementado com uma divisão dinâmica de trabalho. Isto ocorre porque o volume de processamento é o mesmo para todos os pixels e, portanto, para todas as regiões da imagem. Adotamos, então, uma divisão estática de trabalho.

Na versão paralela do classificador ICM, a troca de mensagens é mais intensa que no caso do MAXVER, pois a cada iteração, os processadores devem atualizar suas informações sobre os pixels de borda dos respectivos blocos de imagem. Além disso, o término do processo depende dos resultados gerados por todos os processadores. Por este motivo, há um controle sobre o número de pixels modificados por cada processador. Se a soma destes pixels ultrapassar o limite mínimo definido pelo usuário, o programa deve ser encerrado. Se isto não ocorrer, é uma indicação de que alguma porção da imagem ainda precisa ser processada. Neste caso todos os processadores devem executar mais iterações, até que toda a imagem atenda aos limites estabelecidos para a classificação ICM em questão.

Não é necessário nenhum mecanismo de sincronização entre as iterações do ICM, pois a própria dependência na troca de mensagens, natural da versão paralela do programa, garante esta situação. Cada processador só prossegue o seu trabalho depois de enviar seus resultados e receber os dados

classificados provenientes dos outros processadores vizinhos, confirmando que o término do processo iterativo ainda não foi atingido.

Mesmo com a imagem sendo processada de forma particionada, todos os fatores que são critérios de parada do programa são calculados considerando-se a imagem inteira.

Cada processador classifica sua porção da imagem, seguindo o algoritmo de classificação original utilizado no programa SPRING, inclusive calculando o fator β localmente, a cada iteração. É importante ressaltar que, a cada iteração, as linhas de borda das faixas classificadas são trocadas entre os processadores vizinhos, garantindo a informação contextual. A figura 3.5 exemplifica esta troca de informação entre processadores vizinhos, considerando-se a utilização de quatro processadores.

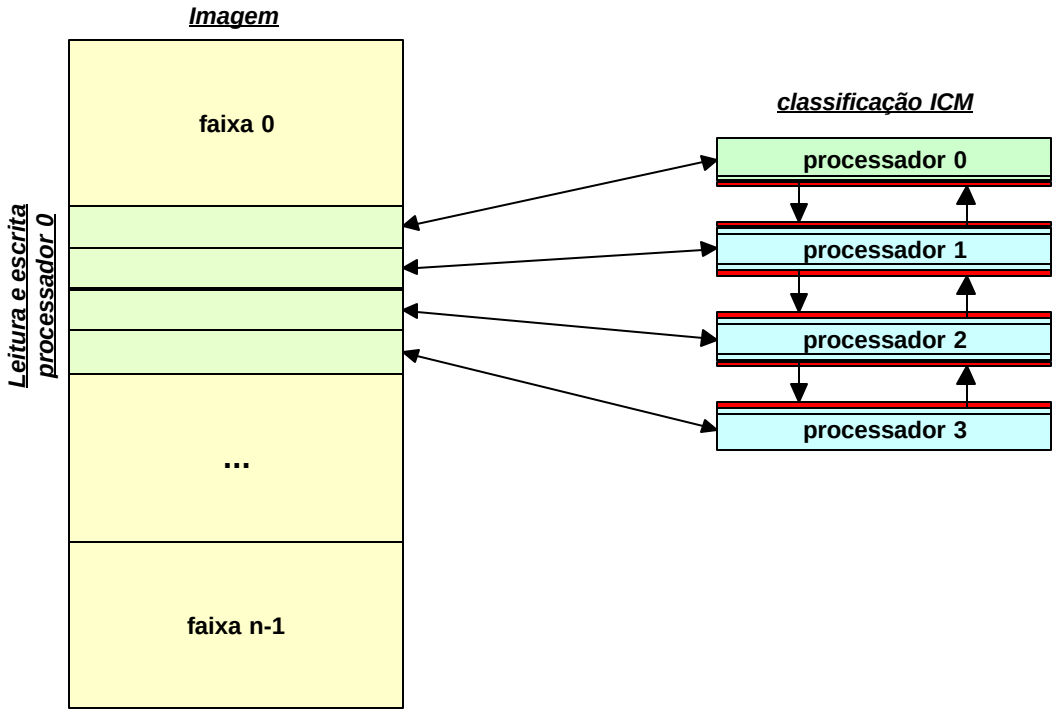


Fig. 3.5 - Divisão da imagem em faixas retangulares e comunicação entre 4 processadores (classificação ICM).

Outro ponto importante a ser observado diz respeito a como a imagem é percorrida no processo de classificação ICM (ver item 2.3 deste trabalho). Com a divisão das imagens em faixas retangulares, este tipo de percorrimento pode gerar pequenas diferenças na classificação de alguns pixels, quando comparados aos resultados gerados pelos programas originais. Entretanto, estas variações são mínimas, não chegando a 1% do total de pixels da imagem (conforme veremos mais adiante no Capítulo 5). É importante ressaltar que os pixels correspondentes a este 1%, não estão concentrados numa mesma região, evitando o efeito de formação de linhas indesejáveis na imagem.

No apêndice B deste trabalho, apresentamos os modelos simplificados de objetos, dinâmico e funcional para os elementos envolvidos no processo de classificação ICM em paralelo.

3.4 AVALIAÇÃO DA IMAGEM CLASSIFICADA

Com os programas paralelos de classificação MAXVER e ICM desenvolvidos e executados, conduzimos a validação e a avaliação dos resultados obtidos para as diferentes imagens testadas. Para que as imagens geradas fossem consideradas válidas, analisamos alguns aspectos, conforme descrito a seguir

Assumimos que um programa estava correto quando os resultados produzidos (imagens classificadas) fossem idênticos aos gerados pelo correspondente código original do SPRING. Esta comparação foi feita através de pequenos utilitários criados para comparar as imagens pixel a pixel. Entretanto, é importante ressaltar que houve uma pequena diferença nas imagens geradas pelo programa paralelo de classificação ICM, em relação aos resultados do programa sequencial. Este fato ocorreu devido ao particionamento da imagem em faixas retangulares. Porém, a variação foi mínima (menos de 1% do total de

pixels da imagem), e os resultados foram considerados aceitáveis conforme podemos observar no Capítulo 5 deste trabalho.

3.5 AVALIAÇÃO DOS PROGRAMAS PARALELOS

Programas paralelos são desenvolvidos para resolver grandes problemas em menos tempo. Portanto, o principal aspecto avaliado é quanto aos tempos de execução nos diferentes equipamentos e a real vantagem em relação ao processamento puramente seqüencial. Como o processo de I/O é um dos tópicos para os quais ainda não existem soluções paralelas universalmente aceitas, os tempos de processamento foram avaliados sem levar em conta este aspecto.

Para analisar o desempenho de um sistema qualquer, devemos levar em conta três fatores:

- O hardware sendo utilizado;
- A linguagem de programação e o compilador;
- O volume dos dados a serem processados.

Quando se trata de um sistema paralelo, devemos acrescentar o número de processadores envolvidos. Considerando todos estes aspectos, estaremos interessados em medir o *speedup (S)* do processo, que é a razão entre os tempos de execução de uma solução seqüencial e de uma solução paralela. Esta razão é dada por:

$$S(n, p) = \frac{T_s(n)}{T_p(n, p)} \tag{3.1}$$

onde:

- *n = dimensão do volume de dados processados;*
- $T_s(n)$ = *tempo de execução de solução sequencial;*
- $T_p(n, p)$ = *tempo de execução de solução em paralelo para "p" processadores.*

Outro fator considerado é a *eficiência (E)*, uma grandeza que indica a utilização média dos processadores num programa paralelo. A eficiência é dada por:

$$E(n, p) = \frac{S(n, p)}{p} = \frac{T_s(n)}{pT_p(n, p)} \tag{3.2}$$

Avaliamos o desempenho obtido em cada versão paralela do processo de classificação através dos valores resultantes de *speedup* e *eficiência.*

3.6 RESUMO

Este capítulo apresentou aspectos característicos da utilização da biblioteca de troca de mensagens MPI. Observamos que um mesmo programa desenvolvido utilizando MPI pode ser executado em ambientes de memória distribuída, assim como ambientes de memória compartilhada.

Também foi apresentado que o principal fator de retardo na execução dos programas são os procedimentos de I/O. Neste trabalho, para garantir a portabilidade, optamos por deixar os procedimentos de leitura e armazenamento em disco para somente um processador.

Neste capítulo pudemos entender a diferença entre divisão de trabalho *estática* e divisão de trabalho *dinâmica*.

Foram apresentados os procedimentos empregados para a classificação de imagens em paralelo, além dos modelos de objetos, dinâmico e funcional para os dois programas (MAXVER e ICM).

Concluindo, as imagens geradas pelos programas de classificação em paralelo foram comparadas às geradas pelos programas originais seqüenciais, pixel a pixel. Resultados mais detalhados serão apresentados nos próximos capítulos.

Também compreendemos o que é *speedup* e *eficiência*, quando avaliamos um programa paralelo, e quais fatores contribuem para suas variações.

No próximo capítulo apresentaremos os dados utilizados para os testes dos programas de classificação em paralelo e os respectivos resultados. Para tal, consideramos os diferentes ambientes operacionais e equipamentos onde os programas foram executados.

CAPÍTULO 4

TESTES E RESULTADOS OBTIDOS

Apresentamos, neste capítulo, as configurações básicas dos sistemas computacionais envolvidos no desenvolvimento, testes e avaliação dos resultados da classificação em paralelo.

Descrevemos, ainda, as imagens que serviram de fonte de informação para os processos de classificação MAXVER e ICM.

Após a descrição dos sistemas e imagens utilizadas, apresentamos, na forma tabular, os tempos obtidos com a execução dos programas de classificação seqüencial e paralela, para cada algoritmo (MAXVER ou ICM), sistema e imagem avaliados.

4.1 SISTEMAS COMPUTACIONAIS UTILIZADOS

Inicialmente, conduzimos os testes de classificação num *cluster* de estações SPARC interligadas com rede padrão Ethernet. Contudo, este equipamento foi utilizado apenas para depuração dos programas e para testes iniciais com um volume de informação menor, pois neste tipo de implementação a comunicação entre as estações é muito mais lenta do que num sistema verdadeiramente paralelo.

As novas versões dos programas de classificação foram avaliadas considerando-se ambientes genuinamente paralelos de memória compartilhada e de memória distribuída. No primeiro caso, utilizamos um sistema do tipo PC, baseado em processadores Intel-Pentium/Pro 200 MHz que compartilham um mesmo barramento, ao qual está conectado um módulo comum de memória com 128 MB.

Este equipamento possui duas limitações importantes, que devem ser observadas durante a avaliação dos programas. A primeira, se refere ao fato da memória, sendo compartilhada, ter o seu espaço dividido (em termos de ocupação) entre os processadores, limitando a capacidade de armazenamento de cada um. A outra limitação, diz respeito ao acesso, pois um único barramento, só pode ser utilizado por um processador de cada vez, gerando concorrência e retardando o processamento.

O sistema Intel-Pentium/Pro mencionado, possui quatro processadores e opera sob sistema operacional Linux RedHat 6.0, onde está disponível a biblioteca de comunicação MPICH.

Nos testes com sistemas de memória distribuída, utilizamos um sistema IBM-SP/2, no qual já existe uma implementação nativa da biblioteca MPI. Conduzimos os testes no sistema SP/2 da Universidade do Vale do Paraíba (Univap). Este equipamento possui quatro processadores distribuídos como quatro estações (nós) distintas e interligadas por uma chave de interconexão de alta velocidade. Esta chave é do tipo "crossbar", a qual permite comunicação direta, de um nó a qualquer outro, a uma velocidade compatível, de tal forma que os nós possam ser mantidos ocupados, trabalhando em um mesmo programa, na maior parte do tempo. Em sistemas SP/2 que não possuem esta chave, a comunicação entre os nós é mais lenta, não atendendo aos requisitos de desempenho do conjunto.

Cada nó, do sistema IBM-SP2 da Univap, é equivalente a uma estação IBM-POWER2, modelo 590, absolutamente separados. Cada um destes nós opera com o sistema operacional AIX 4.1.4, que é o UNIX da IBM. Cada processador do SP2 da Univap possui 256 MB de memória principal. Dois desses processadores, são do tipo *wide*, cujo desempenho é de 156 milhões operações de ponto flutuante, em dupla precisão, por segundo (MFLOPS DP). Os outros dois processadores são do tipo *thin*, cujo desempenho é um pouco

menor: 132 MFLOPS DP. Esta diferença de velocidade entre os processadores do IBM-SP2 da Univap, influencia nos resultados de processamento dos programas paralelos (execução com 2 ou 4 processadores).

A tabela 4.1 apresenta a comparação entre os valores dos índices SPECint95 (desempenho em inteiros) e SPECfp95 (desempenho em ponto flutuante) para algumas máquinas representativas.

TABELA 4.1 - COMPARAÇÃO DE DESEMPENHO ENTRE PROCESSADORES

Processador	**SPECint95**	**SPECfp95**
Processador wide do SP2	3,8	12,4
Pentium 100 Mhz	3,6	2,8
HP PA8000 160 Mhz	10,4	16,3
Pentium Pro 200 Mhz	8,2	6,75

A grande vantagem em se utilizar um padrão de comunicação para os programas paralelos é a portabilidade dos programas desenvolvidos. Isto pode ser comprovado com os testes realizados nos dois tipos de sistemas mencionados. Em nenhum dos casos foi necessário alterar o código-fonte dos programas. Bastou apenas recompilá-los adequadamente, *linkando-os* com as bibliotecas MPI correspondentes em cada sistema, e configurando as variáveis de ambiente convenientemente.

No equipamento paralelo do tipo PC, não foi necessário alterar as variáveis de ambiente, deixando-as com valores padrão. Entretanto, no ambiente SP2, as seguintes variáveis de ambiente foram definidas, com os respectivos valores:

- *MP_PROCS = 4* (ou 2, correspondendo ao número de processadores utilizados);
- *MP_EUILIB = us* (define utilização da interconexão de alta velocidade);
- *MP_EAGER_LIMIT = 8192* (o padrão é 4096, foi aumentado para permitir transferência de um volume maior de informações nas mensagens), Franke (1994).

Embora os sistemas utilizados não estivessem alocados exclusivamente para os testes deste trabalho, tomamos o cuidado de executar os programas em horários onde os sistemas não estivessem sendo sobrecarregados pela execução de programas de outros usuários.

4.2 IMAGENS UTILIZADAS

Utilizamos imagens orbitais adquiridas através do sensor multi-espectral *Thematic-Mapper* do satélite **Landsat**.

Para testes e avaliação dos programas, basicamente, foram utilizadas duas imagens. A primeira faz parte dos exemplos que acompanham o sistema SPRING. Ela cobre uma área da região de Brasília. Esta imagem possui um tamanho relativamente pequeno, de 748 linhas por 695 colunas (520 KB por banda espectral).

A outra imagem utilizada foi uma cena (TM-Landsat) inteira. Ela cobre uma região do Vale do Paraíba e possui 6400 linhas por 7000 colunas (44,8 MB por banda espectral).

Nos dois casos, utilizamos três bandas espectrais (bandas 3, 4 e 5), por apresentarem uma melhor resposta na representação das áreas de interesse,

facilitando o processo de amostragem e consequentemente a classificação. Cada uma destas bandas possui resolução espacial de 30x30m.

Mais adiante neste capítulo, apresentamos representações das bandas espectrais de cada imagem, assim como tabelas com os respectivos resultados obtidos pelo processo de classificação em paralelo nos diferentes sistemas utilizados. Tratamos as duas imagens analisadas simplesmente como "Imagem Pequena" e "Imagem Grande".

4.3 MÉTODOS UTILIZADOS PARA AQUISIÇÃO DOS TEMPOS DE PROCESSAMENTO

Basicamente, utilizamos dois procedimentos para obter os tempos de processamento dos programas paralelos. O primeiro consiste em utilizar uma função disponibilizada pela biblioteca MPI, chamada *MPI_Wtime*. Esta função retorna um valor de precisão dupla representando o número de segundos passados desde um instante anterior qualquer. Com esta função é possível calcular intervalos de tempo em um programa paralelo.

No segundo procedimento, utilizamos a *profile interface* do MPI. A idéia *da profile interface* é permitir aos usuários da biblioteca MPI redefinir suas funções. Para entendermos este procedimento, é necessário saber que o padrão MPI estabelece que cada implementação da biblioteca deve permitir que suas funções sejam chamadas pelo nome usual ou pelo nome iniciado pela letra P. Por exemplo, um programa pode utilizar a função MPI_Send ou PMPI_Send, que terá a mesma funcionalidade. Desta maneira, cada usuário pode personalizar sua implementação das funções, independentemente do código MPI original.

Seguindo este conceito, foi criada uma biblioteca com redefinições de algumas funções MPI. Cada redefinição chama a função correspondente que inicia com a letra P, além de calcular o tempo de execução da função chamada. Ao gerar os programas paralelos, primeiramente *linkamos* com a biblioteca com as redefinições das funções. Em seguida, *linkamos* com as bibliotecas MPI. Desta forma, quando os programas chamam as funções MPI, na verdade eles estão chamando pelas funções que foram redefinidas e estas, por sua vez, se encarregam de chamar as funções verdadeiras, calculando o tempo gasto e armazenando-o em arquivos para posterior avaliação.
Com este procedimento, podemos monitorar o tempo gasto com cada chamada a funções MPI pelos programas paralelos, sem necessidade de alteração no código-fonte original dos programas.

4.4 AMOSTRAS E PARÂMETROS UTILIZADOS PARA CLASSIFICAÇÃO

Como os programas de classificação paralelos utilizam os arquivos de amostras gerados pelo SPRING, utilizamos o próprio SPRING para coleta de amostras nas imagens utilizadas.

Para a imagem pequena (região de Brasília), definimos quatro classes e adquirimos trinta e cinco amostras. Para a imagem grande (região do Vale do Paraíba), definimos seis classes e coletamos sessenta e cinco amostras.

O processo de coleta das amostras não foi muito rigoroso e nem está em questão a qualidade das amostras. O objetivo foi apenas fornecer informações suficientes para avaliar o desempenho dos programas paralelos.

Os algoritmos de classificação MAXVER e ICM necessitam de alguns parâmetros, fornecidos pelo usuário, para as suas execuções. A seguir,

apresentamos os parâmetros utilizados nos testes realizados neste trabalho, de acordo com cada algoritmo:

- MAXVER
  - ❖ Limiar de classificação = 100%
- ICM
  - ❖ Limiar de classificação = 100%
  - ❖ Percentual de pixels modificados (critério de parada) = 1%

Com estes parâmetros obtivemos resultados satisfatórios no que diz respeito aos dois algoritmos. Apenas como informação adicional, com estes parâmetros o programa de classificação ICM em paralelo realizou 4 iterações para a imagem menor e 3 iterações para a imagem maior.

4.5 TESTES COM IMAGEM PEQUENA

4.5.1 Bandas

A primeira imagem utilizada está representada pelas figuras 4.1, 4.2 e 4.3, a seguir.

Fig. 4.1 - Banda espectral 3 de sensor TM-Landsat da região de Brasília/DF.

Fig. 4.2 - Banda espectral 4 de sensor TM-Landsat da região de Brasília/DF.

Fig. 4.3 - Banda espectral 5 de sensor TM-Landsat da região de Brasília/DF.

As figuras 4.4 e 4.5 representam as bandas temáticas obtidas após a classificação em paralelo. É importante lembrar que a imagem classificada pelo algoritmo MAXVER também é fonte de informação para a classificação ICM, além das três bandas originais. Nestas figuras, pode-se notar que a imagem resultante da classificação ICM tem um aspecto mais “homogêneo”, isto é, não há tantos pontos isolados como na classificação pelo MAXVER. Este efeito é resultado do aspecto contextual presente no método ICM.

Fig. 4.4 - Banda temática da região de Brasília gerada por classificador MAXVER em paralelo.

Fig. 4.5 - Banda temática da região de Brasília gerada por classificador ICM em paralelo.

4.5.2 Tempos de Processamento

Para esta imagem, consideramos somente os tempos totais (por processador), incluindo o tempo gasto com os procedimentos de I/O, o que tende a aumentar significativamente o tempo de execução dos programas.

Obs: Os tempos são dados em *segundos*.

- Tempos em Equipamento Pentium Pro/Intel com 4 Processadores

TABELA 4.2 - TEMPOS DE CLASSIFICAÇÃO MAXVER COM MEMÓRIA COMPARTILHADA

MAXVER							
	NP=1	NP=2		NP=4			
Processador	**0**	**0**	**1**	**0**	**1**	**2**	**3**
Tempo Total	4,28	2,59 (60%)	2,53 (59%)	1,90 (44%)	1,86 (43%)	1,86 (43%)	1,86 (43%)

TABELA 4.3 - TEMPOS DE CLASSIFICAÇÃO ICM COM MEMÓRIA COMPARTILHADA

ICM							
	NP=1	NP=2		NP=4			
Processador	**0**	**0**	**1**	**0**	**1**	**2**	**3**
Tempo Total	27,42	14,14 (51%)	14,06 (51%)	8,05 (29%)	8,00 (29%)	8,00 (29%)	8,00 (29%)

- Tempos em Equipamento SP2 com 4 Processadores

TABELA 4.4 - TEMPOS DE CLASSIFICAÇÃO MAXVER COM MEMÓRIA DISTRIBUÍDA

MAXVER							
	NP=1	NP=2		NP=4			
Processador	0	0	1	0	1	2	3
Tempo Total	10,07	5,31 (52%)	5,25 (52%)	2,79 (27%)	2,71 (26%)	2,68 (26%)	2,69 (26%)

TABELA 4.5 - TEMPOS DE CLASSIFICAÇÃO ICM COM MEMÓRIA DISTRIBUÍDA

ICM							
	NP=1	NP=2		NP=4			
TEMPO	P0	P0	P1	P0	P1	P2	P3
Total	74,31	37,76 (50%)	37,70 (50%)	19,09 (25%)	18,98 (25%)	19,00 (25%)	18,99 (25%)

4.6 TESTES COM IMAGEM GRANDE

4.6.1 Bandas

As três bandas da cena inteira utilizada para os testes estão representadas pelas figuras 4.6, 4.7 e 4.8, a seguir.

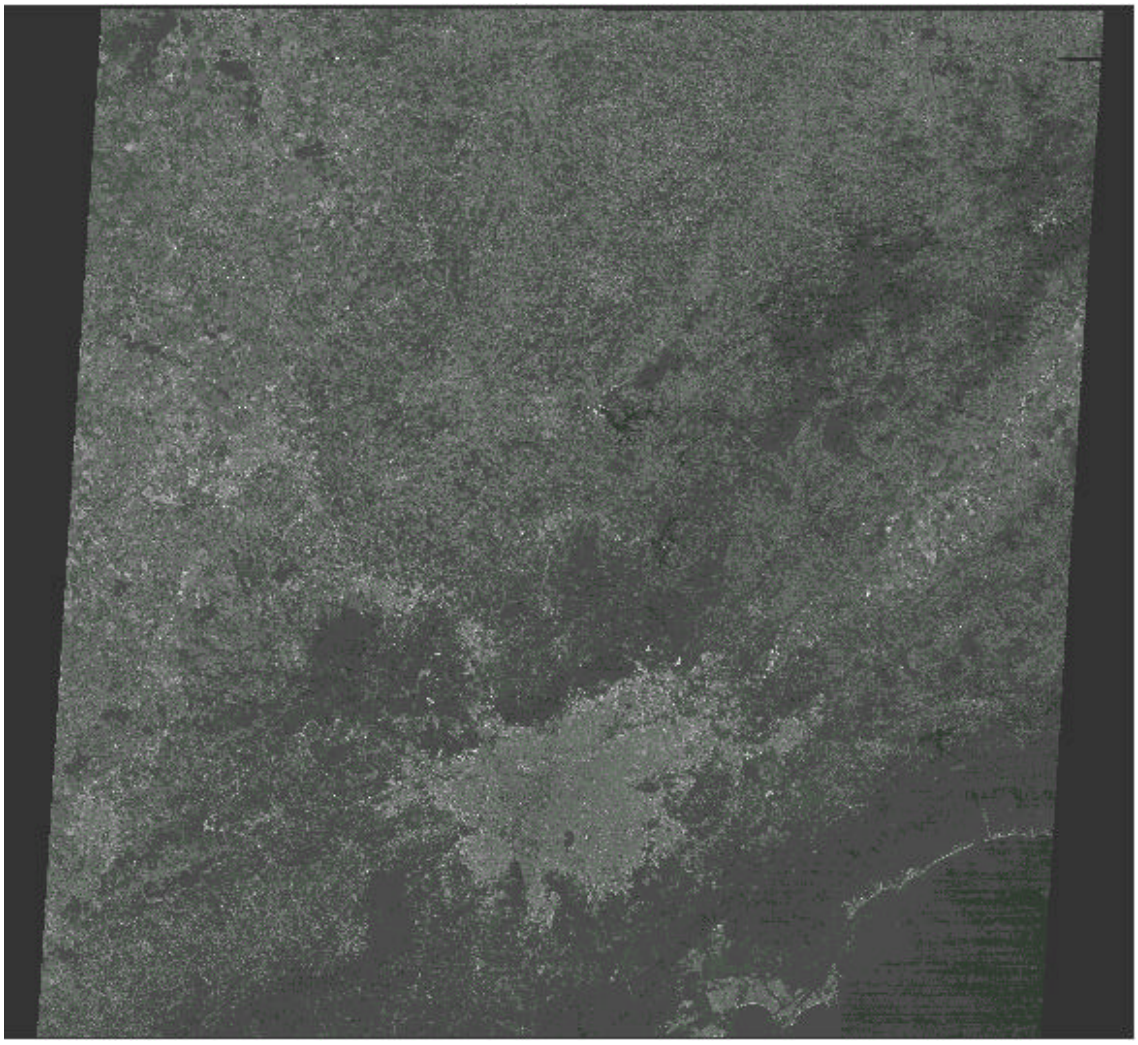


Fig. 4.6 - Banda espectral 3 de sensor TM-Landsat de região do Vale do Paraíba.

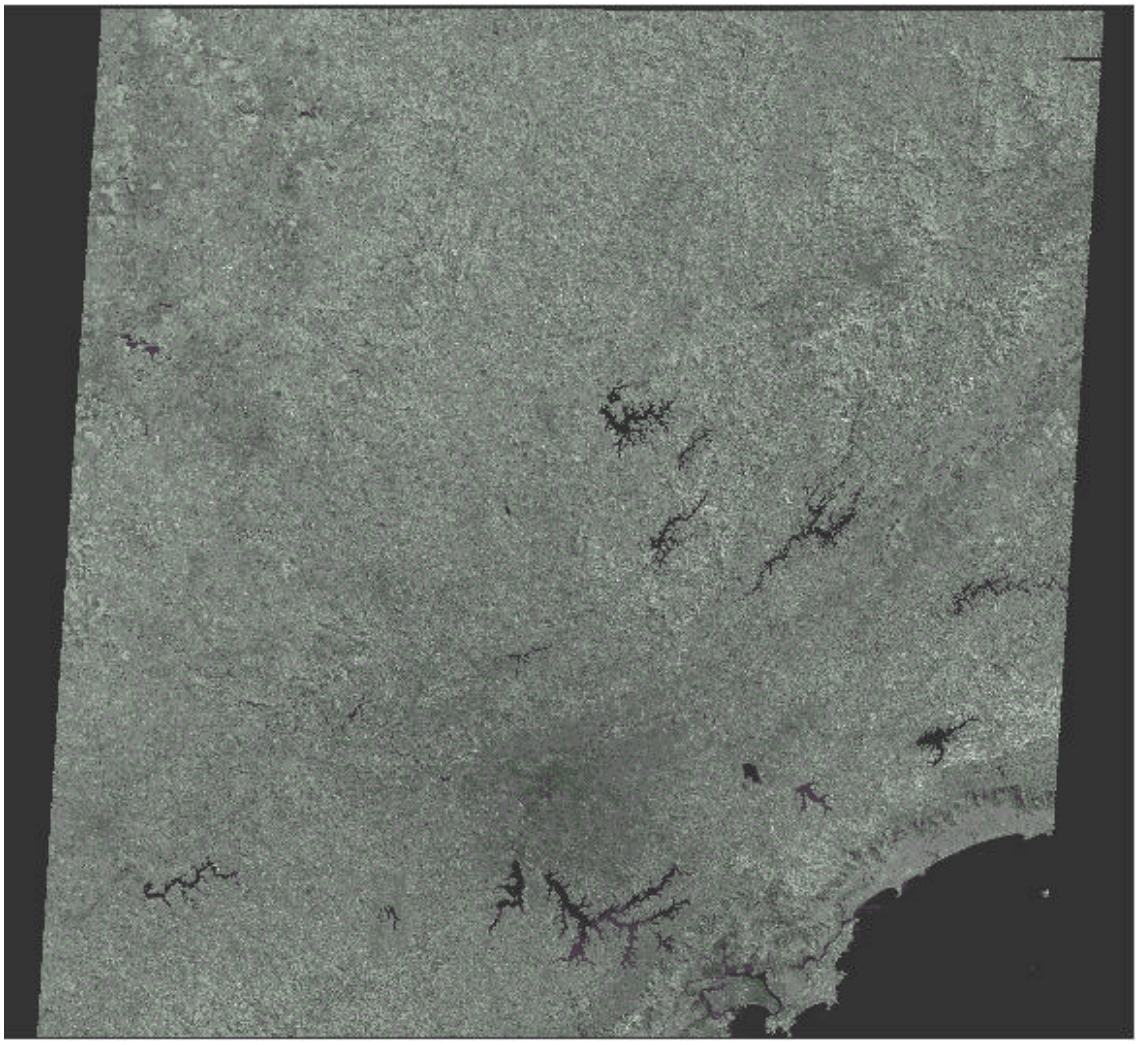


Fig. 4.7 - Banda espectral 4 de sensor TM-Landsat de região do Vale do Paraíba.

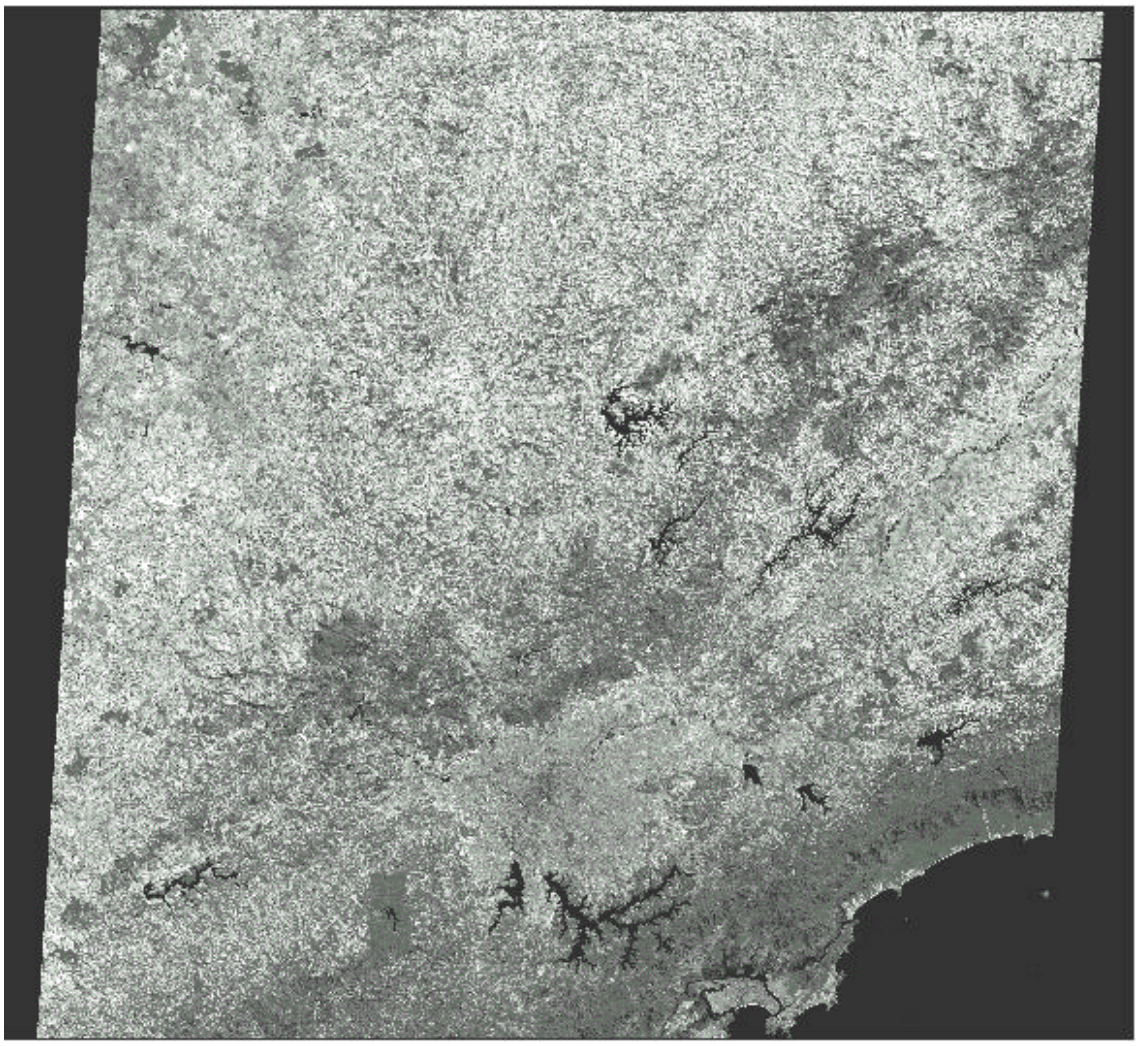


Fig. 4.8 - Banda espectral 5 de sensor TM-Landsat de região do Vale do Paraíba.

As figuras 4.9 e 4.10 representam as bandas temáticas obtidas após a classificação em paralelo.

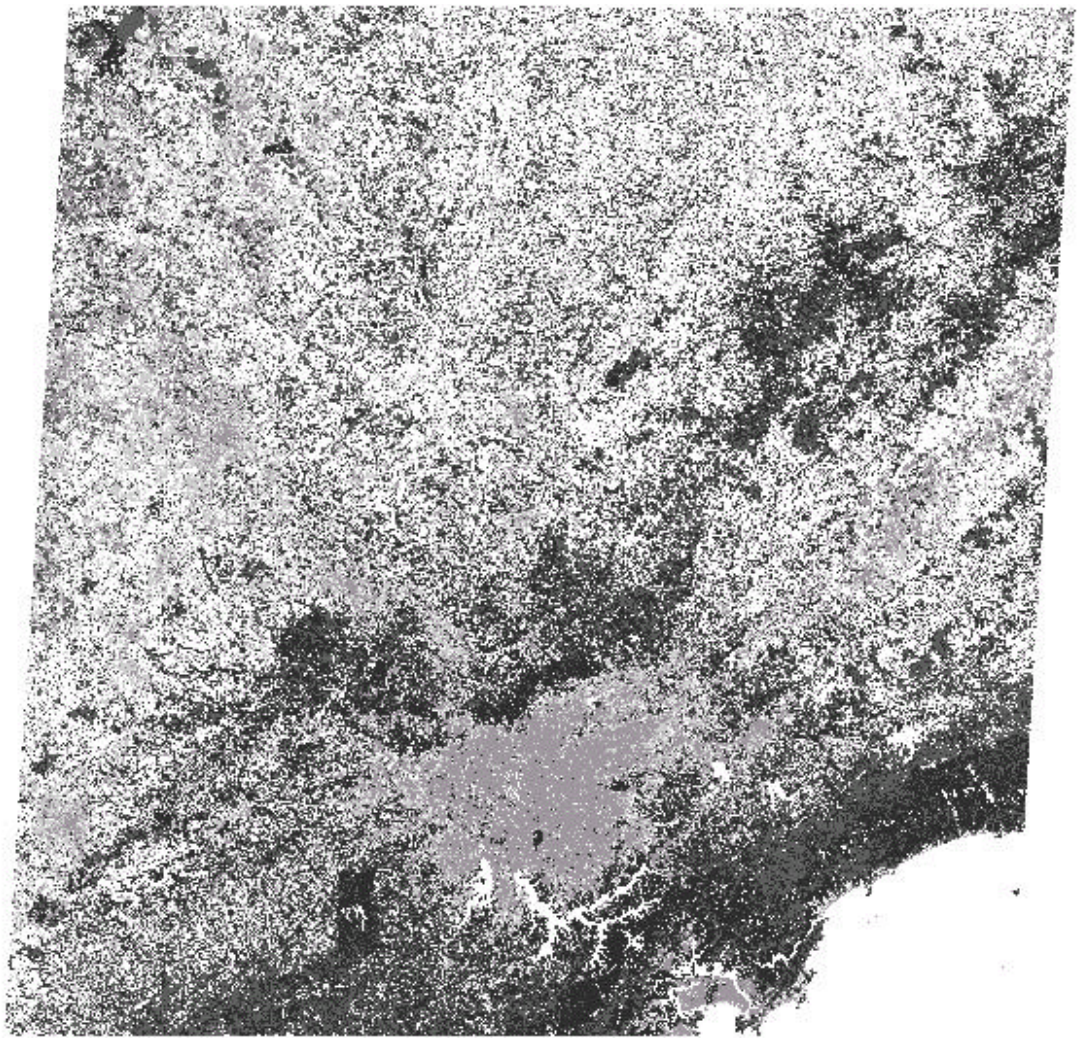


Fig. 4.9 - Banda temática de região do Vale do Paraíba gerada por classificador MAXVER em paralelo.

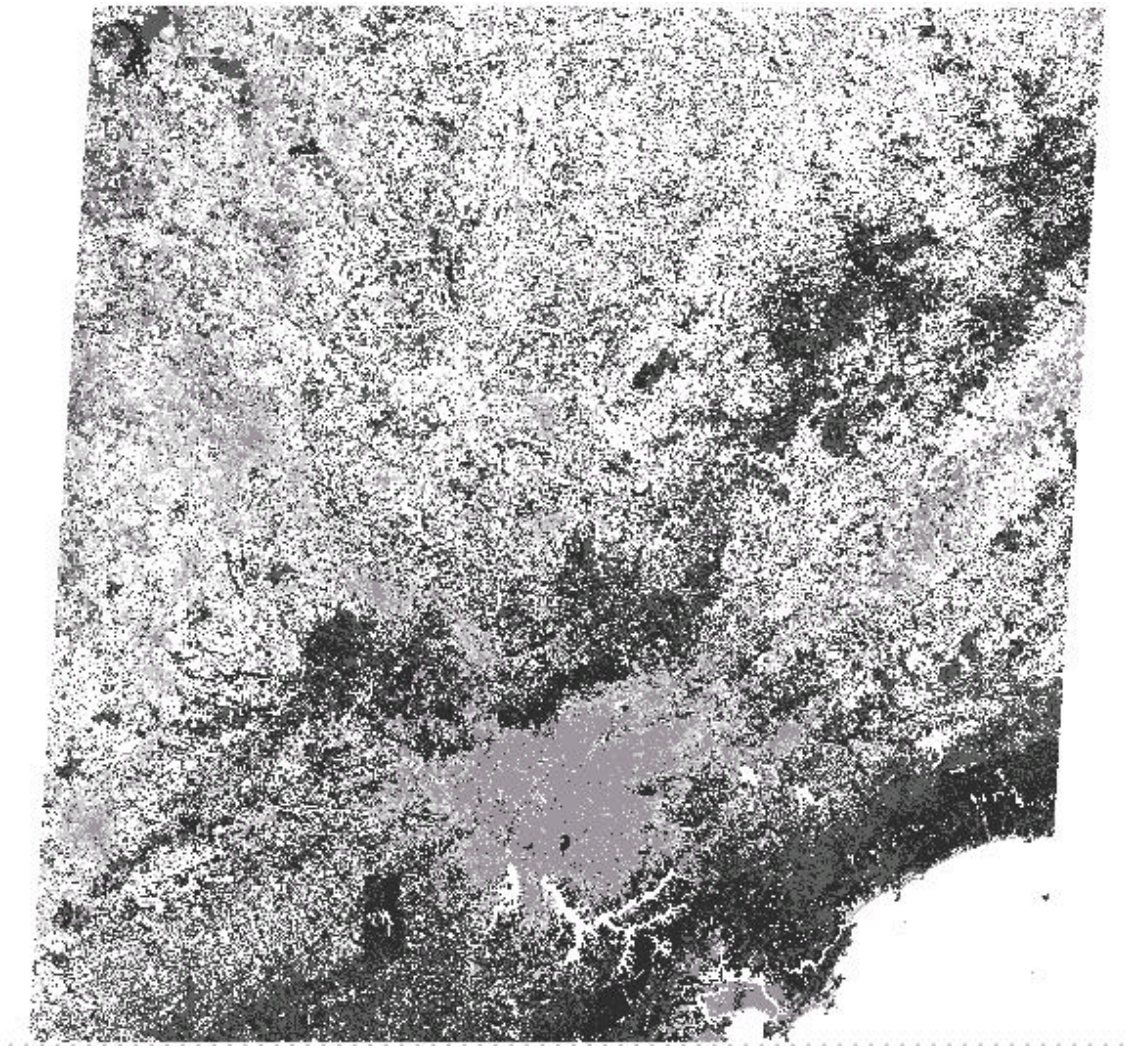


Fig. 4.10 - Banda temática de região do Vale do Paraíba gerada por classificador ICM em paralelo.

4.6.2 Tempos de Processamento

Conforme foi apresentado no capítulo anterior, para otimizar a classificação e reduzir o acesso a disco ao processar uma imagem grande (6400 linhas x 7000 colunas), consideramos uma subdivisão em faixas retangulares (100 linhas x 7000 colunas). A cada iteração, cada uma das faixas da imagem é classificada individualmente com todos os processadores (cada um sobre sua porção)

trabalhando ao mesmo tempo sobre ela. Ao medir os tempos de classificação, consideramos os tempos médios por faixa da imagem, além do tempo total.
Os tempos, para esta imagem, foram medidos de duas maneiras. Primeiramente, consideramos somente o trecho dos programas em que era realizada a classificação propriamente dita, desprezando os procedimentos de I/O e o restante da comunicação entre os processadores durante distribuição e agrupamento das sub-faixas das bandas da imagem. Desta forma, medindo somente a rotina de classificação, pudemos avaliar o ganho obtido somente com o processamento, em paralelo, do algoritmo. As tabelas 4.6 a 4.9 apresentam os resultados desta medição.

Obs: Todos os tempos são dados em *segundos*.

- Tempos em Equipamento Pentium Pro/Intel com 4 Processadores (somente rotina de classificação).

TABELA 4.6 - TEMPOS DE ROTINA DE CLASSIFICAÇÃO MAXVER, COM MEMÓRIA COMPARTILHADA

MAXVER							
	NP=1	NP=2		NP=4			
Processador	**0**	**0**	**1**	**0**	**1**	**2**	**3**
Tempo Médio por Faixa	7,84	4,58	4,38	3,31	3,05	3,18	3,18
Tempo Total	539,83	283,14 (52%)		173,44 (32%)			

TABELA 4.7 - TEMPOS DE ROTINA DE CLASSIFICAÇÃO ICM, COM MEMÓRIA COMPARTILHADA

ICM							
	NP=1	NP=2		NP=4			
Processador	**0**	**0**	**1**	**0**	**1**	**2**	**3**
Tempo Médio por Faixa	15,94	8,02	7,99	4,51	4,53	4,56	4,55
Tempo Total	3061,67	1541,51 (50%)		835,01 (27%)			

- Tempos em Equipamento SP2 com 4 Processadores (somente rotina de classificação).

TABELA 4.8 - TEMPOS DE ROTINA DE CLASSIFICAÇÃO MAXVER, COM MEMÓRIA DISTRIBUÍDA

MAXVER							
	NP=1	NP=2		NP=4			
Processador	0	0	1	0	1	2	3
Tempo Médio por Faixa	10,05	5,00	5,01	2,54	2,42	2,77	2,77
Tempo Total	640,24	321,25 (50%)		184,69 (28%)			

TABELA 4.9 - TEMPOS DE ROTINA DE CLASSIFICAÇÃO ICM, COM MEMÓRIA DISTRIBUÍDA

ICM							
	NP=1	NP=2		NP=4			
Processador	0	0	1	0	1	2	3
Tempo Médio por Faixa	24,08	12,07	12,07	7,11	7,11	7,11	7,11
Tempo Total	4666,74	2330,33 (49%)		1386,40 (29%)			

Na segunda forma de medição, consideramos os tempos totais de execução dos programas, com toda a comunicação e os procedimentos de I/O, sempre avaliando cada processador isoladamente. Nesta etapa, também foram medidos os tempos gastos com as funções de comunicação do MPI. Os resultados desta medição estão nas tabelas 4.10 a 4.13. Conforme se pode ver nestas tabelas, os tempos gastos na função MPI_Barrier deixam explícito o fato de que os demais processadores levam um tempo considerável aguardando que o processador 0 realize as operações de I/O (leitura e escrita das imagens em disco).

Obs: Todos os tempos são dados em *segundos*.

- Tempos em Equipamento Pentium Pro/Intel com 4 Processadores

TABELA 4.10 - TEMPOS DE CLASSIFICAÇÃO MAXVER, COM MEMÓRIA COMPARTILHADA (COM TEMPOS DE FUNÇÕES MPI)

MAXVER							
	NP=1	NP=2		NP=4			
Processador	**0**	**0**	**1**	**0**	**1**	**2**	**3**
Função							
MPI_Send	0,00	9,84	8,87	33,26	1,57	1,81	2,14
MPI_Recv	0,00	12,11	17,01	10,52	21,47	31,16	32,29
MPI_Barrier	0,00	19,25	112,94	21,59	127,67	117,46	118,97
MPI_Bcast	0,00	0,00	0,00	0,00	0,67	0,22	0,03
Tempo Total	607,312185	414,62 (68%)	414,64 (68%)	318,88 (52%)	318,88 (52%)	318,91 (52%)	318,88 (52%)

TABELA 4.11 - TEMPOS DE CLASSIFICAÇÃO ICM, COM MEMÓRIA COMPARTILHADA (COM TEMPOS DE FUNÇÕES MPI)

ICM							
	NP=1	NP=2		NP=4			
Processador	**0**	**0**	**1**	**0**	**1**	**2**	**3**
Função							
MPI_Send	0,00	106,59	15,90	86,80	7,54	7,00	7,02
MPI_Recv	0,00	21,40	118,57	24,22	58,28	85,59	93,95
MPI_Barrier	0,00	85,80	563,48	73,06	617,15	589,96	583,99
MPI_Bcast	0,00	1,78	5,52	0,84	34,39	1,12	3,17
MPI_Reduce	0,01	6,40	2,04	66,56	1,68	36,64	5,67
Tempo Total	3530,397303	2239,29 (63%)	2239,28 (63%)	1511,99 (42%)	1512,00 (42%)	1512,00 (42%)	1512,00 (42%)

- Tempos em Equipamento SP2 com 4 Processadores

TABELA 4.12 - TEMPOS DE CLASSIFICAÇÃO MAXVER, COM MEMÓRIA DISTRIBUÍDA (COM TEMPOS DE FUNÇÕES MPI)

MAXVER							
	NP=1	NP=2		NP=4			
Processador	**0**	**0**	**1**	**0**	**1**	**2**	**3**
Função							
MPI_Send	0,00	2,04	0,56	3,41	0,29	0,34	0,34
MPI_Recv	0,00	2,45	3,10	26,03	1,69	3,15	4,65
MPI_Barrier	0,00	1,70	11,67	1,18	35,29	9,65	7,54
MPI_Bcast	0,00	0,00	0,00	0,00	0,00	0,00	0,00
Tempo Total	644,55	335,37 (52%)	335,35 (52%)	197,56 (30%)	197,52 (30%)	197,50 (30%)	197,51 (30%)

TABELA 4.13 - TEMPOS DE CLASSIFICAÇÃO ICM, COM MEMÓRIA DISTRIBUÍDA (COM TEMPOS DE FUNÇÕES MPI)

ICM							
	NP=1	NP=2		NP=4			
Processador	**0**	**0**	**1**	**0**	**1**	**2**	**3**
Função							
MPI_Send	0,00	7,97	1,88	12,91	1,20	1,56	1,30
MPI_Recv	0,00	4,00	10,60	5,18	5,70	11,27	17,07
MPI_Barrier	0,01	5,22	30,48	3,54	48,46	42,28	36,64
MPI_Bcast	0,01	0,02	14,67	0,03	225,51	0,08	0,82
MPI_Reduce	0,04	2,35	0,03	266,86	0,03	51,41	0,04
Tempo Total	4699,71	2377,85 (50%)	2377,82 (50%)	1443,53 (30%)	1443,45 (30%)	1443,47 (30%)	1443,49 (30%)

4.7 RESUMO

Neste capítulo, observamos que os programas desenvolvidos puderam ser utilizados em diferentes equipamentos de processamento paralelo, sem necessitar de alterações no seu código-fonte. Isto graças ao padrão de comunicação utilizando bibliotecas MPI. Utilizamos equipamento de menor porte, com memória compartilhada, bem como um equipamento maior, o qual possui uma arquitetura de memória distribuída.

Identificamos a utilização de imagens obtidas por sensor óptico no padrão TM-Landsat e utilização de três bandas espectrais para cada uma delas. Também apresentamos o número de classes e amostras envolvidas no processo de classificação para os respectivos testes e avaliações. Mostramos que os sistemas são viáveis tanto para um volume de informação menor (imagem 748 linhas x 695 colunas) como para um volume maior (imagem com 6400 linhas x 7000 colunas). Exibimos as bandas temáticas obtidas com a classificação, através de figuras representativas.

Este capítulo também informou sobre as metodologias utilizadas na medição dos tempos de processamento e como a biblioteca MPI facilita este processo através de sua *profile interface*.

Finalmente, obtivemos os tempos de cada execução dos programas de classificação, utilizando, para isto, os diferentes critérios de medição.

O próximo capítulo avalia os resultados obtidos, calculando os valores de *speedup* e *eficiência*, considerando os diferentes equipamentos, assim como o número de processadores utilizados na classificação. Também são analisados os erros obtidos com os programas paralelos.

CAPÍTULO 5

ANÁLISE DOS RESULTADOS

Como foi mencionado no Capítulo 3 deste trabalho, os programas paralelos precisam ser avaliados para podermos comprovar as vantagens da classificação de imagens utilizando este tipo de processamento.

Neste capítulo, calculamos os valores obtidos para as variáveis *speedup* e *eficiência*. Baseado nos dados obtidos, apresentamos conclusões sobre o desempenho dos programas e as possíveis vantagens, ou não, de se utilizar divisão dinâmica de trabalho entre os processadores.

Os programas também são avaliados quanto às diferenças que eles podem produzir. Para isso, comparamos as imagens temáticas resultantes do processo com as originais geradas pelos programas seqüenciais.

5.1 DESEMPENHO E EFICIÊNCIA

Para o cálculo do ganho de desempenho (*speedup*) e da *eficiência* dos programas paralelos, consideramos o número de processadores envolvidos, assim como o equipamento em que cada programa estava sendo executado.

Os valores para *speedup* e *eficiência* podem ser obtidos utilizando-se as fórmulas 3.1 e 3.2, respectivamente.

Nos testes realizados, avaliamos apenas o processamento com a imagem grande, pois uma vez que os programas foram analisados para um volume de informações significativo (uma cena inteira, por exemplo), acreditamos ser desnecessário avaliar os resultados com uma imagem menor.

A tabela 5.1 apresenta os resultados obtidos para os testes no equipamento Pentium Pro/Intel. Estes resultados foram baseados nos tempos totais dos processamentos com 2 e 4 processadores, apresentados nas tabelas 4.6 e 4.7 do capítulo anterior.

TABELA 5.1 - SPEEDUP E EFICIÊNCIA EM EQUIPAMENTO PENTIUM PRO/INTEL

Algoritmo	Número de Processadores	Tempo Seqüencial	Tempo Paralelo	Speedup	Eficiência
MAXVER	2	539,83	283,14	1,90	0,95
	4		173,44	3,11	0,77
ICM	2	3061,67	1541,51	1,98	0,99
	4		835,01	3,66	0,91

Os resultados obtidos com os testes no equipamento SP2 podem ser observados na tabela 5.2, cujos valores são baseados nos tempos totais das tabelas 4.8 e 4.9 (2 e 4 processadores) do capítulo anterior.

TABELA 5.2 - SPEEDUP E EFICIÊNCIA EM EQUIPAMENTO SP2/IBM

Algoritmo	Número de Processadores	Tempo Seqüencial	Tempo Paralelo	Speedup	Eficiência
MAXVER	2	640,24	321,25	1,99	0,99
	4		184,69	3,46	0,86
ICM	2	4666,74	2330,33	2,00	1,00
	4		1386,40	3,36	0,84

Analisamos também qual o ganho total de desempenho obtido para os programas completos, ou seja, incluindo os tempos de I/O e de comunicação nos programas paralelos. Estes tempos podem ser observados nas tabelas 4.10, 4.11, 4.12 e 4.13 do capítulo anterior. Os valores de speedup nesta caso refletem melhor os ganhos observados por um usuário real destes programas. As tabelas 5.3 e 5.4 apresentam estes novos valores de speedup total.

TABELA 5.3 - SPEEDUP TOTAL EM EQUIPAMENTO PENTIUM PRO/INTEL

Algoritmo	Número de Processadores	Tempo Seqüencial	Tempo Paralelo	Speedup
MAXVER	2	607,31	414,64	1,46
	4		318,91	1,90
ICM	2	3530,39	2239,29	1,57
	4		1512,00	2,33

TABELA 5.4 - SPEEDUP TOTAL EM EQUIPAMENTO SP2/IBM

Algoritmo	Número de Processadores	Tempo Seqüencial	Tempo Paralelo	Speedup
MAXVER	2	644,55	335,37	1,92
	4		197,56	3,26
ICM	2	4699,71	2377,85	1,97
	4		1443,53	3,25

Um valor para *speedup* pode ser considerado bom na proporção em que ele se aproxima do valor correspondente ao número de processadores "*p*" envolvidos na execução do programa. Podemos dizer que o valor do *speedup* não é bom se ele é muito menor que *p*. As figuras 5.1 e 5.2 representam graficamente os valores obtidos para *speedup*, em comparação aos valores ideais.

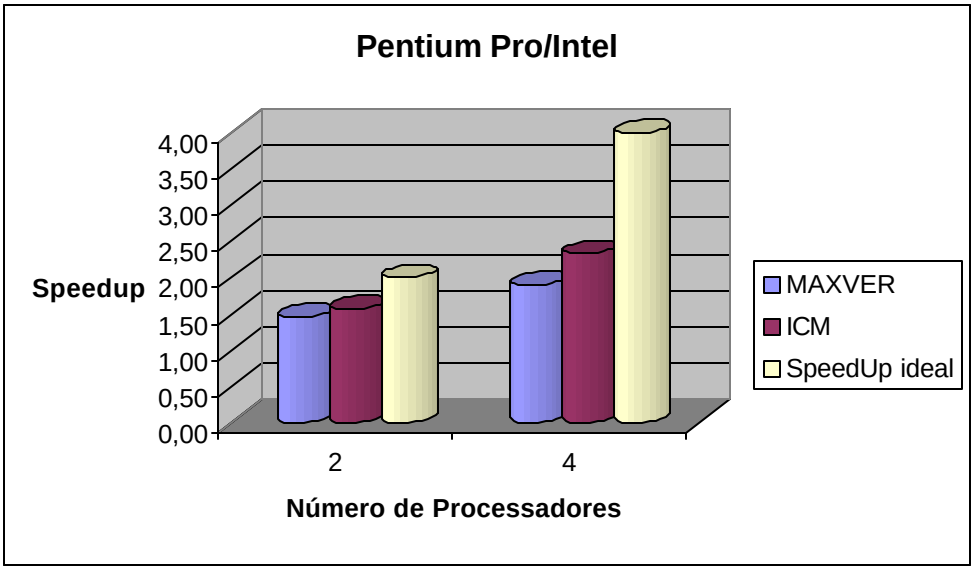


Fig. 5.1 – Valores para *speedup* total em equipamento Pentium Pro/Intel.

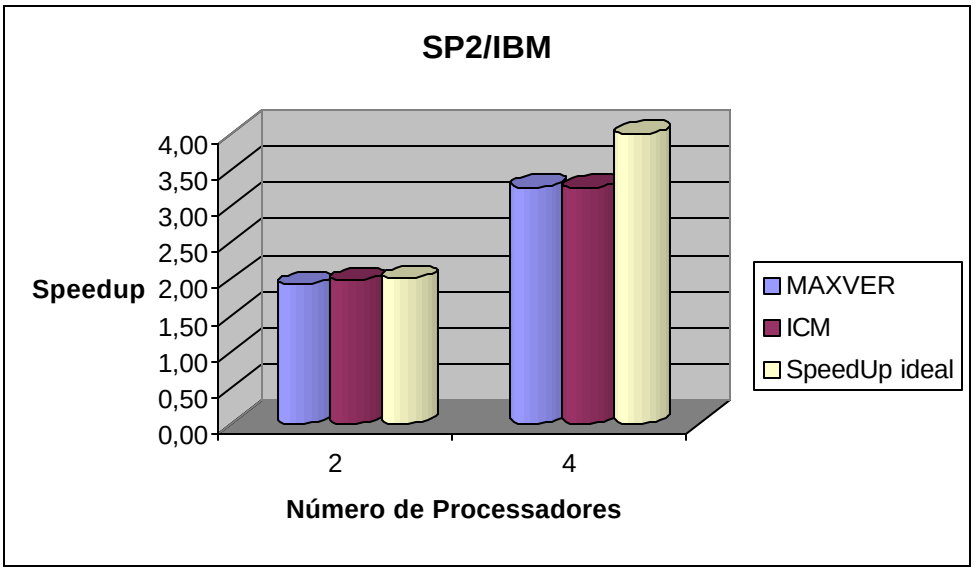


Fig. 5.2 – Valores para *speedup* total em equipamento SP2/IBM.

Observando os resultados calculados para *speedup*, podemos concluir que, em geral, os programas para classificação MAXVER e ICM em paralelo possuem valores de *speedup* muito bons, principalmente no sistema SP2/IBM. Isto quer dizer que o tempo de execução diminui quase que proporcionalmente ao número de processadores utilizados.

Torna-se desnecessário um aprofundamento na análise da *eficiência*, pois podemos observar facilmente, que a ociosidade dos processadores se deve ao processo de I/O, e não ao desbalanceamento de carga entre eles.

5.2 COMPARAÇÃO ENTRE OS EQUIPAMENTOS UTILIZADOS

Os gráficos a seguir, têm como objetivo estabelecer uma comparação entre os equipamentos utilizados nos testes deste trabalho. Como já foi mencionado, o primeiro equipamento é um Pentium-Pro com processadores Intel e possui memória compartilhada. O segundo é um equipamento SP2 da IBM e possui uma arquitetura de memória distribuída. Para geração dos gráficos,

consideramos somente os resultados obtidos com o processamento da imagem grande.

A figura 5.3 representa os tempos de execução dos programas para o algoritmo MAXVER. Consideramos somente o trecho em que é realizada a classificação propriamente dita. Desta forma, são desprezados os tempos gastos com rotinas de I/O e comunicação entre os processadores. Os dados utilizados foram retirados das tabelas 4.6 e 4.8 do capítulo anterior.

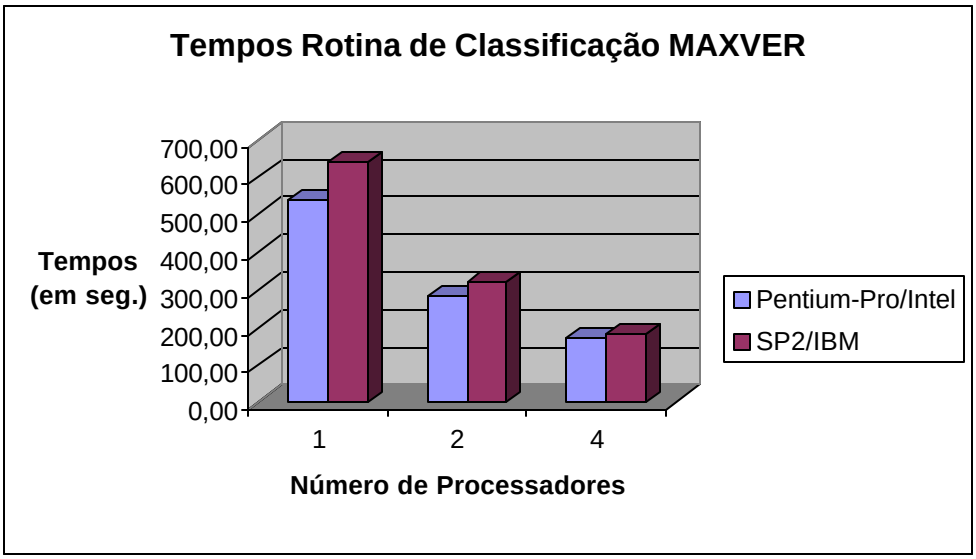


Fig. 5.3 - Tempos de execução considerando somente rotina de classificação MAXVER

Da mesma forma, os tempos para o algoritmo ICM são representados na figura 5.4. Este dados estão contidos nas tabelas 4.7 e 4.9 do Capítulo 4 deste trabalho.

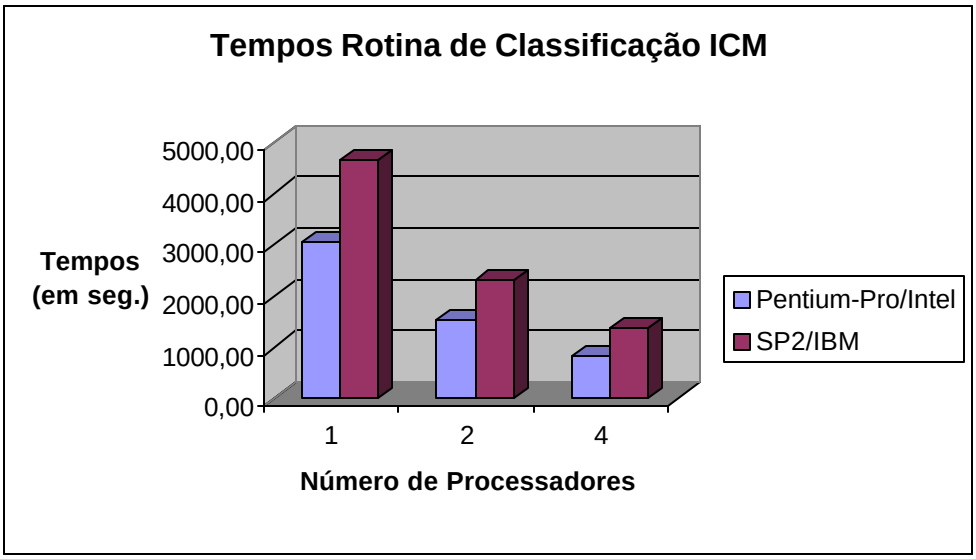


Fig. 5.4 - Tempos de execução considerando somente rotina de classificação ICM

A figura 5.5, representa os tempos totais dos programas de classificação, para o algoritmo MAXVER. Neste caso, também foram considerados os tempos de I/O e comunicação, que podem ser obtidos nas tabelas 4.10 e 4.12 do capítulo anterior.

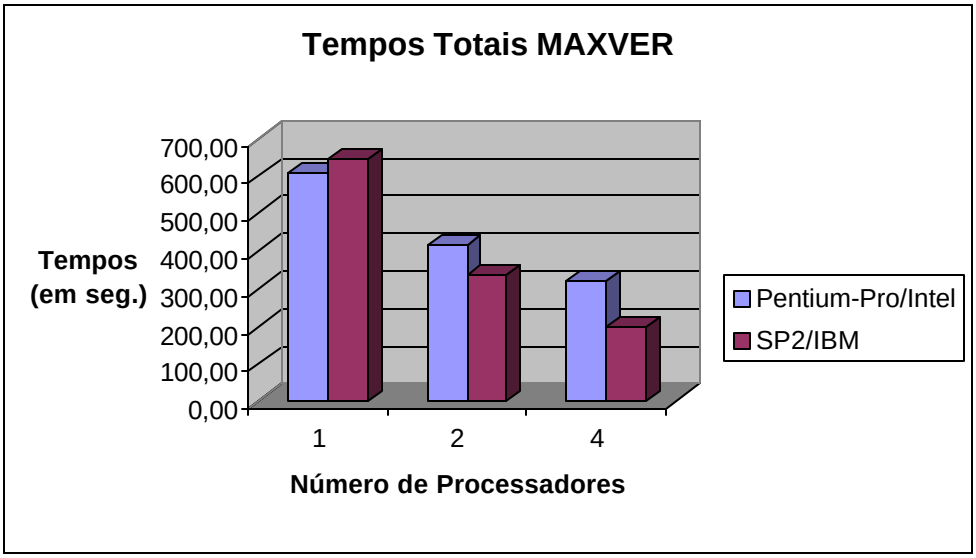


Fig. 5.5 - Tempos totais de classificação MAXVER

Os tempos totais para os programas de classificação ICM, podem ser observados nas tabelas 4.11 e 4.13 do capítulo anterior. A figura 5.6 representa estes tempos.

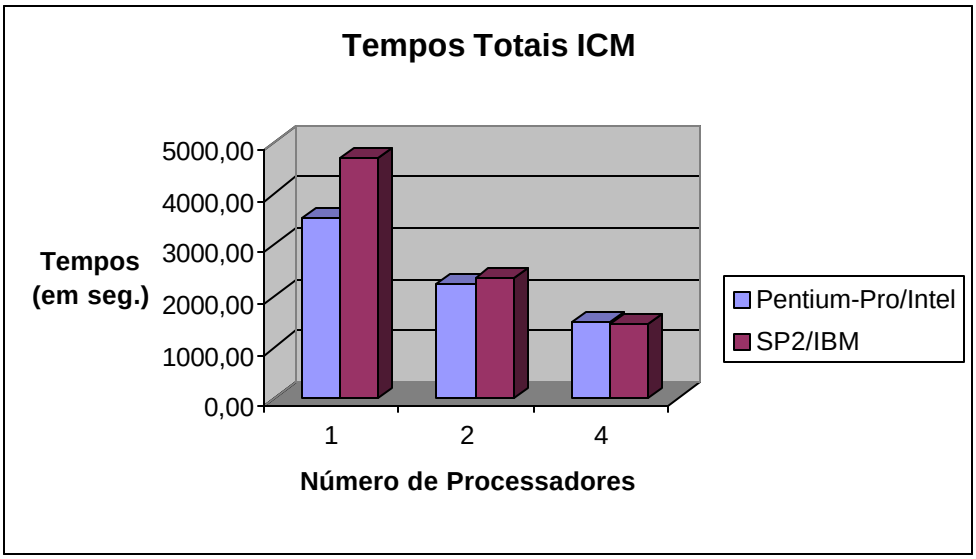


Fig. 5.6 - Tempos totais de classificação ICM

Pelos gráficos apresentados, podemos observar que, apesar do equipamento com memória compartilhada realizar algumas classificações em menos tempo, o equipamento de memória distribuída mantém um desempenho proporcional à medida que se aumenta o número de processadores. Isto confirma a tendência de melhor escalabilidade dos sistemas de memória distribuída, em geral, ao se aumentar o número de processadores (Hwang, 1998). Esta tendência pode ser confirmada pelos melhores valores de speedup obtidos no sistema SP2/IBM, conforme já mostrado nas figuras 5.1 e 5.2.

5.3 POTENCIAL GANHO COM DIVISÃO DINÂMICA

Quando particionamos as imagens em faixas, e estas em sub-faixas, as quais são distribuídas para cada processador individualmente, seria possível que um processador terminasse a classificação de uma sub-faixa antes do que os

outros e ficasse ocioso até receber uma nova sub-faixa. Este fato poderia ocorrer devido aos diferentes dados das regiões da imagem e de como os algoritmos os processariam.

Por este motivo, consideramos a possibilidade da divisão dinâmica de trabalho. Entretanto, ao observar os resultados obtidos para eficiência dos programas, podemos concluir que a divisão dinâmica de trabalho não contribuiria para um ganho significativo de velocidade no processo de classificação. Isto porque a medição de eficiência nos indica que a utilização de cada processador chegou a uma média de quase 90% durante toda a classificação, ou seja, nenhum processador teve um tempo significativo de ociosidade. O ganho máximo com a divisão dinâmica seria de aproximadamente 10%.

Considerando que a divisão dinâmica de trabalho não traria um potencial ganho de velocidade significativo, optamos por desenvolver os algoritmos utilizando apenas a divisão estática.

5.4 ANÁLISE DE ERROS NA CLASSIFICAÇÃO

Consideramos como erros os pixels das imagens temáticas classificados com valores diferentes dos obtidos com a classificação (MAXVER e/ou ICM) utilizando somente um processador (seqüencial). Considerando que a imagem grande possui 6400 linhas por 7000 colunas, ela possui um total de 44,8 milhões de pixels. A tabela 5.5 apresenta os valores de erro medidos para esta imagem.

TABELA 5.5 – PERCENTUAL DE ERROS DA IMAGEM CLASSIFICADA PARALELAMENTE

	Algoritmo de classificação			
Classificação Paralela	**MAXVER**		**ICM**	
Número de processadores	2	4	2	4
Número de pixels diferentes em relação a classificação seqüencial	0	0	26158	365523
Percentual de erro (aprox.)	0%	0%	0,06%	0,8%

Os resultados obtidos para os programas de classificação utilizando o algoritmo MAXVER são exatamente iguais, o que garante a sua perfeição. Entretanto, para o algoritmo ICM, as imagens são geradas com pequenas diferenças conforme observamos na tabela 5.5. Estas diferenças tendem a ocorrer próximas às divisões das sub-faixas da imagem (ver mais detalhes nos Capítulos 2 e 3 deste trabalho). Isto porque o referido algoritmo é contextual e não pixel a pixel como o MAXVER. Uma diferença básica entre as versões seqüencial e paralela do algoritmo, é que o parâmetro "*beta*", no caso seqüencial, é calculado a partir da imagem completa, enquanto que, no caso paralelo, cada processador calcula tal parâmetro localmente, a partir dos dados em sua sub-faixa; esta aproximação é válida, pois não há consenso sobre a melhor forma de estimação prática daquele parâmetro, podendo mesmo, em alguns casos, ser utilizado um valor constante (Orgambide, 1993). Contudo, o percentual de pixels diferentes não chega a atingir nem 1% do tamanho total da imagem (considerando-se 2 e 4 processadores). Esta diferença pode ser considerada desprezível, uma vez que o objetivo é otimizar a execução da classificação sem comprometer a interpretação das informações das regiões analisadas.

Vale lembrar que na forma como os programas paralelos foram desenvolvidos, quanto maior for o número de processadores utilizados, maior será a porcentagem de erro, pois para cada faixa processada, haverá uma maior subdivisão em sub-faixas.

5.5 RESUMO

Neste capítulo, pudemos comprovar que, mesmo com o equipamento de memória distribuída apresentando um melhor desempenho geral em relação ao de memória compartilhada para a maioria dos casos testados, a utilização dos programas paralelos com o padrão de comunicação MPI nos dois ambientes trouxe uma redução significativa nos tempos de execução dos algoritmos (MAXVER e ICM). Isto foi observado através dos valores de *speedup* e *eficiência*, obtidos utilizando-se 2 e 4 processadores. Neste sentido, os usuários poderão classificar um maior volume de informações em menos tempo.

Observamos também que da forma como os programas foram implementados, a utilização de divisão dinâmica de trabalho não traria benefícios significativos ao processamento, sendo portanto pouco atraente.

Ao comparar as imagens temáticas geradas pelos programas em paralelo com aquelas geradas pelos programas seqüenciais, concluímos que as diferenças entre elas eram desprezíveis. Deste modo, os resultados da classificação paralela podem ser considerados aceitáveis, tendo em vista a economia de tempo alcançada.

No próximo capítulo, apresentamos as conclusões a que chegamos com o desenvolvimento deste trabalho e a análise dos dados obtidos, bem como apontamos possíveis direções para estudos futuros.

CAPÍTULO 6

CONCLUSÕES

Este trabalho abordou as questões relacionadas à paralelização de programas classificadores de imagens, sempre focalizando uma padronização nas metodologias de desenvolvimento, a fim de permitir uma maior portabilidade entre as diferentes arquiteturas dos sistemas computacionais paralelos atuais.

Inicialmente, foram discutidas as possibilidades e métodos de análise empregados sobre imagens obtidas a partir de sensores multi-espectrais. Neste sentido, foi destacado o processo de classificação, entre os métodos de análise quantitativa.

Ao destacar a classificação de imagens, foram apresentados os conceitos básicos deste procedimento, como a forma em que as áreas de interesse são mapeadas e identificadas, além do formato dos dados resultantes da classificação. O trabalho também apresentou as características de uma imagem multi-espectral e como as informações das regiões sensoriadas são armazenadas nas diferentes bandas destas imagens.

Foi descrito o funcionamento básico dos algoritmos de classificação e a definição quanto à metodologia de caracterização das classes (métodos *supervisionados* e *não-supervisionados*). O trabalho se baseou, exclusivamente, em algoritmos de classificação supervisionados.

Entre os algoritmos de classificação supervisionados, foram estudados os conceitos relacionados ao da Máxima Verossimilhança (MAXVER) e ao *Iterated Conditional Modes* (ICM). O primeiro é um exemplo de classificação do tipo pixel a pixel, e o segundo é um algoritmo contextual.

Estes algoritmos já estão implementados de forma seqüencial no Sistema para Processamento de Informações Georeferenciadas (SPRING), do INPE. Para que fosse possível estabelecer os objetivos e definir as metodologias de desenvolvimento em relação aos programas paralelos, foi necessário um estudo da situação atual dos dois algoritmos no referido sistema. Para otimização do estudo, os códigos referentes a estes classificadores foram desmembrados do sistema SPRING e criadas versões independentes destes programas de classificação. Desta forma, simplificou-se o processo de análise dos algoritmos, além de facilitar a execução dos programas durante os testes.

Depois de analisar os programas seqüenciais, partiu-se para o estudo das questões relacionadas ao processamento em paralelo. O trabalho apresentou, de forma simplificada, as características das arquiteturas de computadores paralelos (tais como distribuição de memória, formas de armazenamento de dados, comunicação entre processadores, etc), além das metodologias de programação relacionadas. Foi descrito o padrão para troca de mensagens entre processadores *"Message Passing Interface"* (MPI), pois este serviu de base para o desenvolvimento dos programas paralelos deste trabalho.

O padrão MPI é um método poderoso e abrangente para se conseguir paralelismo e portabilidade. A utilização das funções das bibliotecas MPI trouxe alguns pequenos problemas, como uma maior dificuldade no desenvolvimento dos programas. Entretanto, isto garantiu simplicidade e facilidade ao migrar um programa de uma arquitetura paralela para outra.

A apresentação dos programas paralelos, desenvolvidos durante este trabalho, iniciou-se com a descrição das características comuns aos dois programas. Neste ponto, foi analisada a forma de distribuição de trabalho entre os processadores, a forma de leitura e escrita dos dados, além dos aspectos relacionados à utilização do MPI, que permitiriam portabilidade entre os diferentes sistemas.

Após a apresentação das características comuns aos dois programas paralelos, foram apresentadas suas características específicas. Com o auxílio de representação gráfica, utilizando diagramas no padrão *Object Modeling Technique* (OMT), foram descritas as metodologias empregadas no desenvolvimento de cada um dos programas, incluindo a forma de manipulação e comunicação dos dados entre os processadores.

Foram descritos os critérios para avaliação das imagens classificadas pelos programas paralelos. Além disso, também foram definidos os conceitos de *speedup* e *eficiência*, os quais são de extrema importância para avaliação do desempenho dos programas de classificação em paralelo.

Em relação aos testes realizados durante este trabalho, foram descritos os sistemas utilizados, destacando suas características principais, e as imagens que foram utilizadas no processo de classificação. O trabalho também apresentou as técnicas utilizadas para a medição dos tempos de processamento seqüencial e paralelo.

Este trabalho apresentou, ainda, um relatório detalhado com avaliação das imagens classificadas, dos tempos de processamento (considerando ou não fatores de retardo, como por exemplo I/O e memória dos sistemas), do percentual de erro ao se utilizar uma abordagem de classificação de imagens em paralelo, e do desempenho (*speedup* e *eficiência*) dos programas desenvolvidos.

6.1 RESULTADOS ALCANÇADOS

Muitas tarefas, especialmente aquelas ligadas ao processamento digital de imagens, ainda demoram muito tempo para serem realizadas, dependentes de equipamentos com capacidade de processamento insuficiente. Isto acaba retardando, e, muitas vezes, inviabilizando projetos científicos e/ou comerciais, em diferentes setores. Neste trabalho exemplificamos o processo de classificação de imagens, que ainda requer um alto custo computacional e leva tempos razoáveis para processar certos volumes de informações em muitos dos casos de interesse prático.

Este trabalho propôs uma alternativa para reduzir o tempo de classificação de imagens a taxas significativamente menores (para o mesmo volume de informação, comparado ao atual), sem que para isto fosse necessário um grande investimento em equipamentos, nem no desenvolvimento de novos sistemas. Isto foi possível utilizando técnicas e equipamentos de processamento paralelo.

Ao verificar os relatórios apresentados neste trabalho, observamos que o tempo de processamento gasto com a execução dos programas paralelos caiu quase que proporcionalmente ao número de processadores utilizados. Por exemplo, considerando-se os resultados obtidos com o programa ICM no sistema SP2/IBM (Tabela 5.4), notamos que, para execução com dois processadores, o tempo de processamento caiu para aproximadamente 51% em relação à execução com um processador, e, para quatro processadores, chegamos a quase 30% do tempo que é gasto no programa seqüencial. Tais valores estão bem próximos dos limites máximos de ganho de desempenho teoricamente possíveis.

Ao comparar as imagens classificadas pelos programas (MAXVER e ICM) seqüenciais com aquelas geradas pelos programas paralelos, verificamos que

foi introduzido um percentual de erro nas imagens classificadas com o algoritmo ICM. Como este algoritmo é contextual, isto ocorreu devido à necessidade de se subdividir as imagens em faixas retangulares para distribuição entre os processadores. Entretanto, este percentual não chega a 1% do volume total de pixels processados, o que é uma margem de erro desprezível, considerando-se o ganho obtido no tempo de processamento.

Para garantir que os sistemas atuais não necessitassem de grandes alterações e pudessem ser reutilizados nos diferentes equipamentos paralelos, utilizamos o padrão de comunicação MPI, o que se mostrou bastante eficiente. Os programas desenvolvidos não necessitaram de alterações para rodar em equipamentos com arquiteturas diferentes: Sun - Sparc Station (*cluster* de estações), Intel - Pentium Pro e IBM - SP2. Para serem executados nestes diferentes ambientes, bastou serem compilados com as bibliotecas MPI adequadas.

Em resumo, ficou mostrado ao longo deste trabalho que o processo de classificação pode ter seu tempo de processamento consideravelmente reduzido, através do uso de processamento paralelo. Também ficou claro que, se programados com MPI, os algoritmos podem ser portados entre diferentes sistemas, sem necessidade de mudanças no código-fonte. Estes dois fatores, combinados, podem trazer um enorme ganho de desempenho para as diversas aplicações que necessitem classificar imagens obtidas por sensores orbitais.

Acreditamos que este trabalho tenha se tornado uma valiosa fonte de informação e referência para usuários de processamento de imagens que utilizam ou pretendem utilizar técnicas de processamento paralelo, seja com imagens obtidas através de sensores ópticos ou não.

6.2 TRABALHOS FUTUROS

O fato deste trabalho abordar o tópico "Classificação de Imagens" não restringe as metodologias apresentadas a somente este assunto. Atualmente, existem diferentes tópicos relacionados a processamento de imagens que envolvem um alto processamento computacional e necessitam de um menor tempo de execução. As técnicas empregadas neste trabalho devem servir como referência para o desenvolvimento de aplicações paralelas destinadas a outros algoritmos de processamento de imagens.

Outra aplicação das metodologias estudadas neste trabalho está em sistemas paralelos distribuídos, ou seja, o processamento dos algoritmos em ambientes compostos por *clusters* distantes uns dos outros. Como os programas utilizam o padrão MPI, é natural que seja possível executá-los, sem modificações, utilizando recursos de metacomputação atualmente disponíveis (exemplo: ambiente Globus), (Foster e Kesselman, 1998).

Conforme verificado nos vários testes realizados, o tempo gasto em I/O para leitura e escrita das imagens em disco foi considerável. Isto se deve, fundamentalmente, a dois fatos: primeiro, os arquivos com as imagens originais estavam armazenados sob forma seqüencial; em segundo lugar, não havia, nos sistemas utilizados, nenhum suporte para a realização de I/O paralelo. No futuro, à medida em que sistemas com estruturas de suporte a I/O paralelo se tornem disponíveis, será interessante estudar modificações nos algoritmos apresentados, de modo a minimizar a influência do tempo de I/O no desempenho total dos programas.

O trabalho atual também permite uma extensão das metodologias empregadas, utilizando desenvolvimento orientado a objetos. Um possível exemplo seria a integração destes algoritmos paralelos ao ambiente geral de classes e objetos do sistema SPRING.

REFERÊNCIAS BIBLIOGRÁFICAS

Bader, D. A.; Jájá, J.; Harwood, D.; Davis, L. S. **Parallel algorithms for image enhancement and segmentation by region growing with an experimental study**. Institute for Advanced Computer Studies University of Maryland. 1995.

Besag, J. On the statistical analysis of dirty pictures (with discussion). **Journal of the Royal Statistical Society B**, v. 48, n. 3, p. 259-302, 1986.

Besag, J. Towards Bayesian image analysis. **Journal of Applied Statistics**, v. 16, n. 3, p. 395-407, 1989.

Dowd, K.; Severance, C. **High performance computing**. London: 2ª Ed., O'Reilly & Associates. 1998.

Fernandes, S. F. Paralelismo e Imagens: Um experimento de paralelização para a segmentação de imagens com aplicações para a classificação automática de cenas geradas por plataformas orbitais. INPE, 1999.

Foster, I.; Kesselman, C. Globus: A metacomputing infrastructure toolkit. **International Journal of Supercomputer Applications**. v.11, n. 2, p.115-128, 1997.

Foster, I.; Kesselman, C. **The Grid:** blue print for a new computing infrastructure. San Francisco, CA: Morgan Kaufman Publishers, Nov. 1998.

Franke, H.; Hochschild, P.; Pattnaik, P.; Prost, J. P.; Snir, M. **MPI on IBM SP1/SP2**: current status and future directions. IBM T. J. Watson Research Center, 1994.

Gerogii, H. **Gibbs measures and phase transitions**. Berlim: Walter De Gruyter, 1988 (De Gruyter Studies in Mathematics, 9).

Gropp, W.; Lusk, E.; Skjellum, A. **Using MPI.** Cambridge, MA: The MIT Press, 1994.

Hwang, K.; Xu, Z. **Scaleable parallel computing:** technology, architecture, programing. Hong Kong: McGraw-Hill, 1998.

Instituto Nacional de Pesquisas Espaciais. **Documentação de ajuda do SPRING 3.0**. [Programa de Computador]. 1998.

Jensen, J. L.; Moller, J. Pseudolikelihood for exponential family models of spatial processes. Denmark, Department of Theoretical Statistics, Institute of Mathematics, University of Arhus, 1989.

Loveman, D. B. High Performance Fortran**. IEEE Parallel & Distributed Technology**. v. 1, n. 1, p. 25-42, Feb. 1993.

Mendes, C. L. Classificação por máxima verossimilhança num ambiente maciçamente paralelo. INPE, Projeto “Computação: 87/1786 –1.

Moik, J. G. **Digital processing of remotely sensed images**. NASA Scientific and Technical Information Branch. 1980.

Niblack, W. **An introduction to digital image processing**. Denmark: Prentice/Hall International. 1986.

Orgambide, A. C. F. Algumas ferramentas estatísticas na síntese, processamento e analise de imagens de radar de abertura sintética. INPE-5548-TDI/534, 1993.

Pacheco, P. S.; **Parallel programing with MPI**. San Francisco, CA: Morgan Kaufmann Publishers, 1997.

Richards, J. A. **Remote sensing digital image analysis** . Berlim: 2ª Ed. Springer-Verlag.1993.

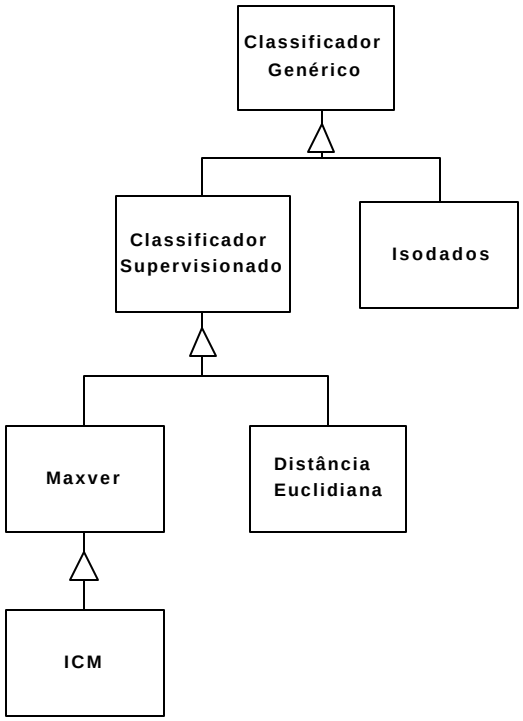
Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorensen, W. **Object-oriented modeling and design.** New Jersey: Prentice Hall, 1991.

Schowengerdt, R. A. **Techniques for image processing and classification in remote sensing**. Tucson, AZ: Academic Press, 1983.

APÊNDICE A

DIAGRAMAS E DICIONÁRIOS DE DADOS DOS PROGRAMAS SEQUENCIAIS

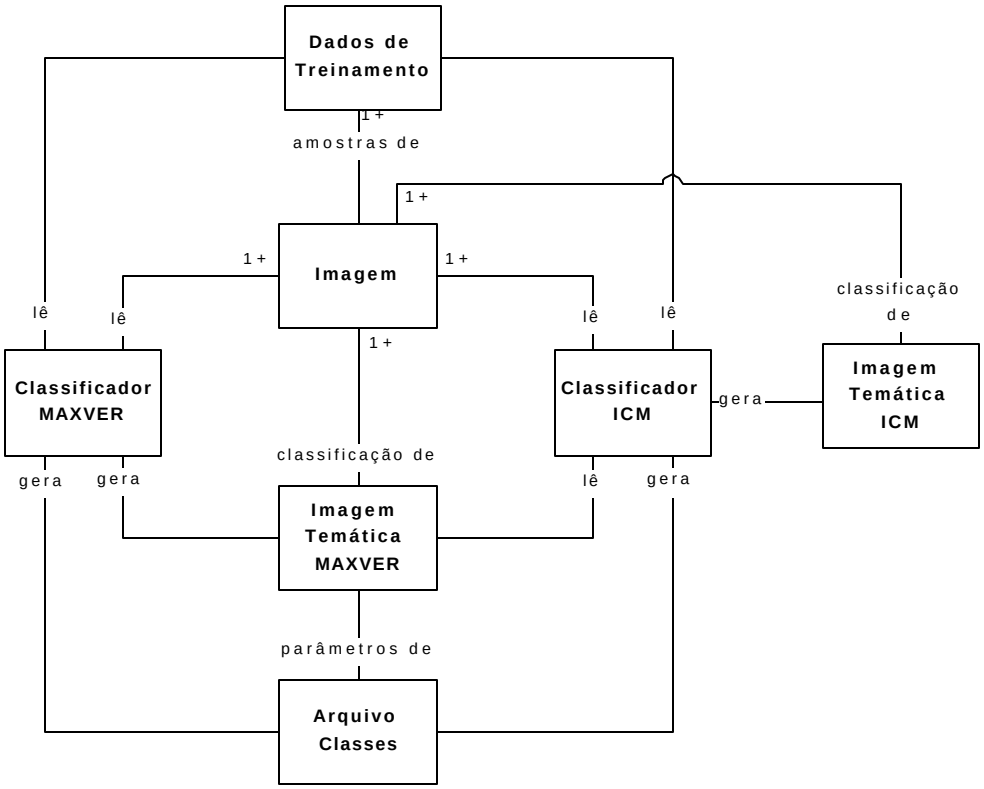
A.1 DIAGRAMA DE OBJETOS PARA MÓDULO DE CLASSIFICAÇÃO DO SPRING



A.2 DICIONÁRIO DE DADOS DO MODELO DE OBJETOS DO MÓDULO DE CLASSIFICAÇÃO DO SPRING

Objeto	Descrição
Classificador Genérico	Classe genérica para classificação de imagens.
Classificador Supervisionado	Classe genérica para classificações supervisionadas.
Distância Euclidiana	Especialização para classificação Distância Euclidiana
ICM	Especialização para classificação ICM.
Isodados	Classe genérica para manipulação de Isodados.
Maxver	Especialização para classificação MAXVER.

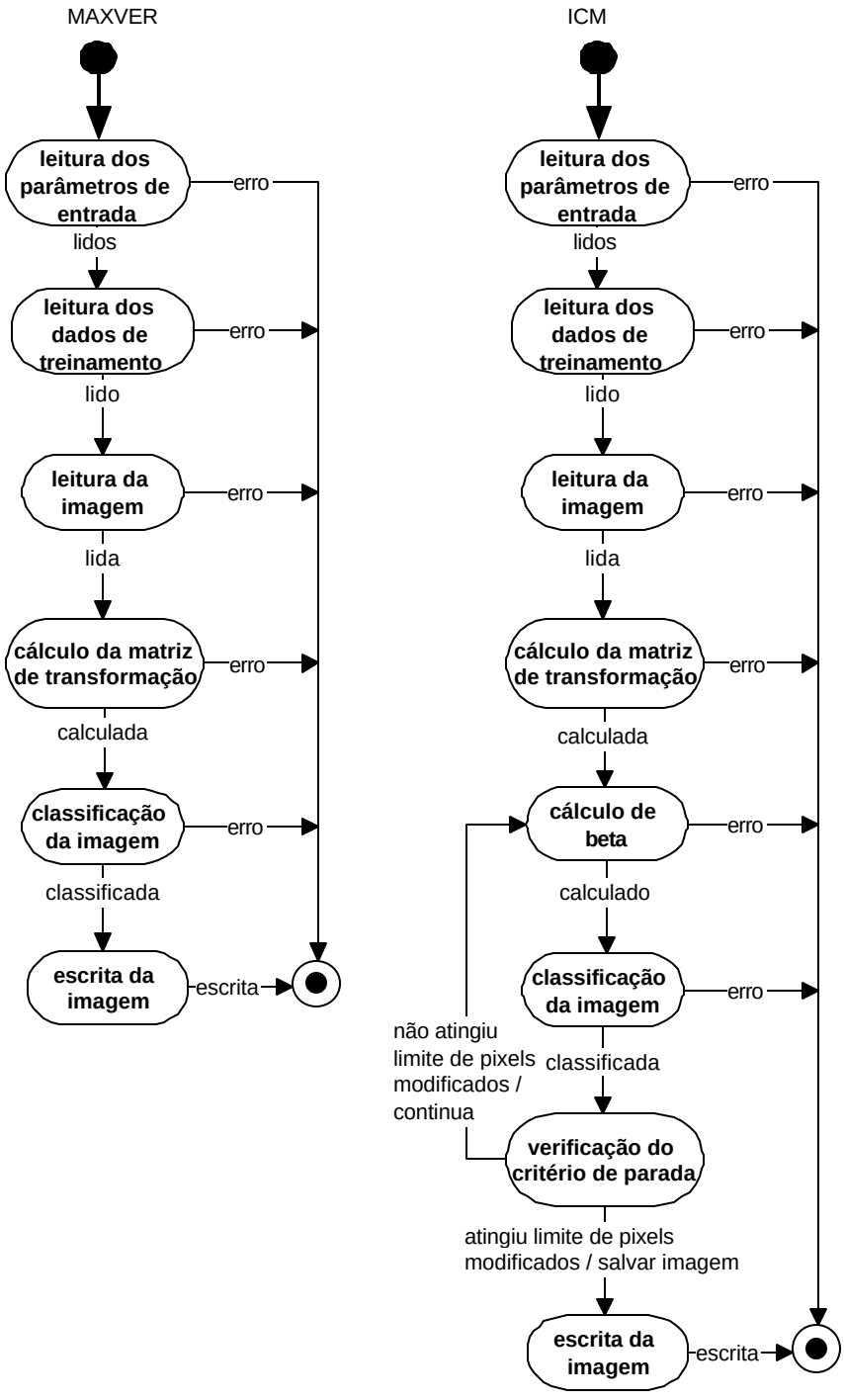
A.3 DIAGRAMA DE OBJETOS DA CLASSIFICAÇÃO SEQUENCIAL



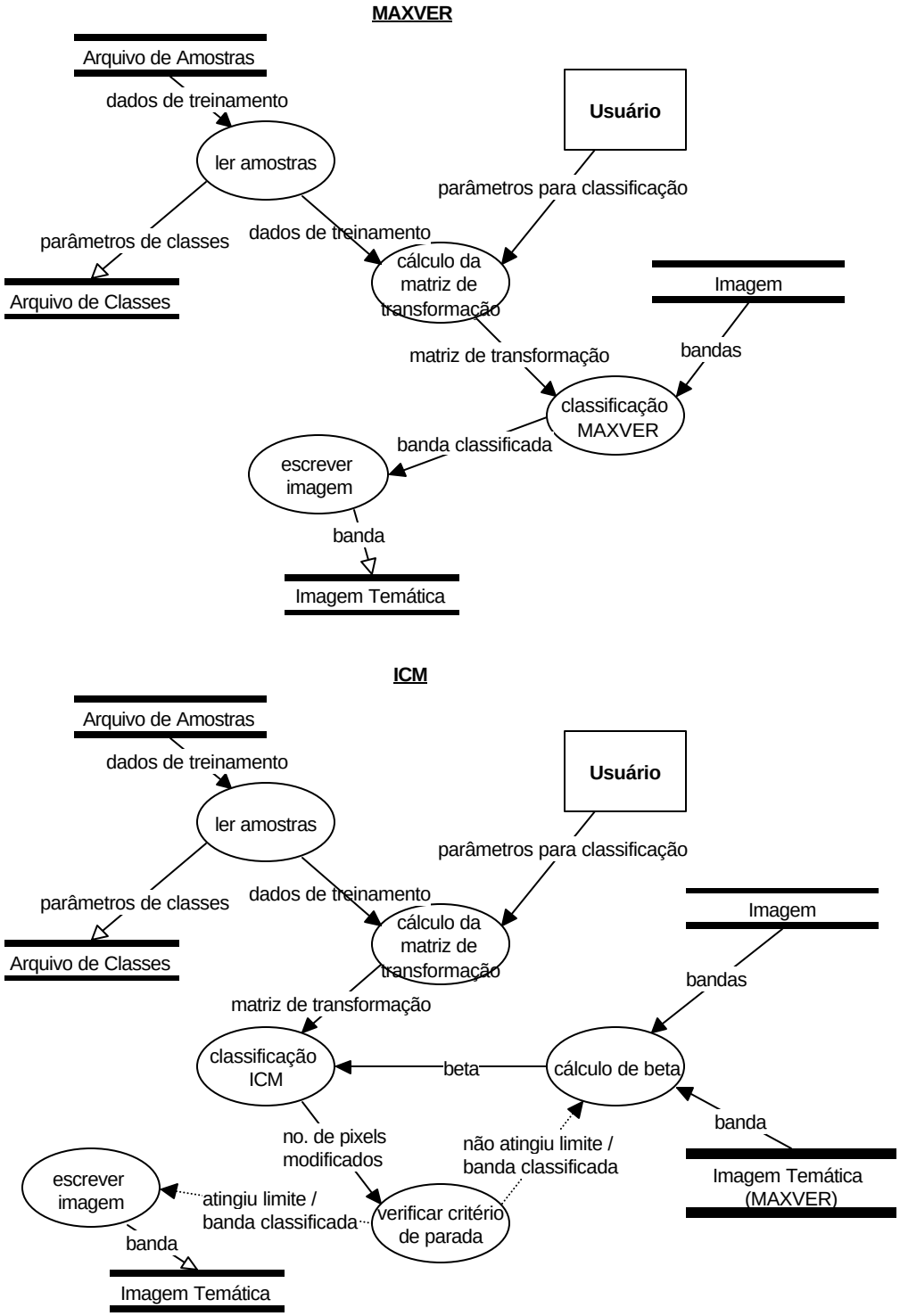
A.4 DICIONÁRIO DE DADOS DO MODELO DE OBJETOS DA CLASSIFICAÇÃO SEQUENCIAL

Objeto	Descrição
Arquivo de Classes	Arquivo gerado por aplicativos de classificação com parâmetros das classes analisadas.
Classificador ICM	Aplicativo para classificação com algoritmo ICM.
Classificador MAXVER	Aplicativo para classificação com algoritmo MAXVER.
Dados de Treinamento	Arquivo gerado no SPRING com amostras de treinamento.
Imagem	Bandas da imagem original a ser classificada.
Imagem Temática ICM	Banda temática gerada por aplicativo de classificação ICM.
Imagem Temática MAXVER	Banda temática gerada por aplicativo de classificação MAXVER

A.5 DIAGRAMAS DE ESTADOS PARA CLASSIFICAÇÃO SEQUENCIAL



A.6 DIAGRAMAS DE FLUXO DE DADOS PARA CLASSIFICAÇÃO SEQUENCIAL



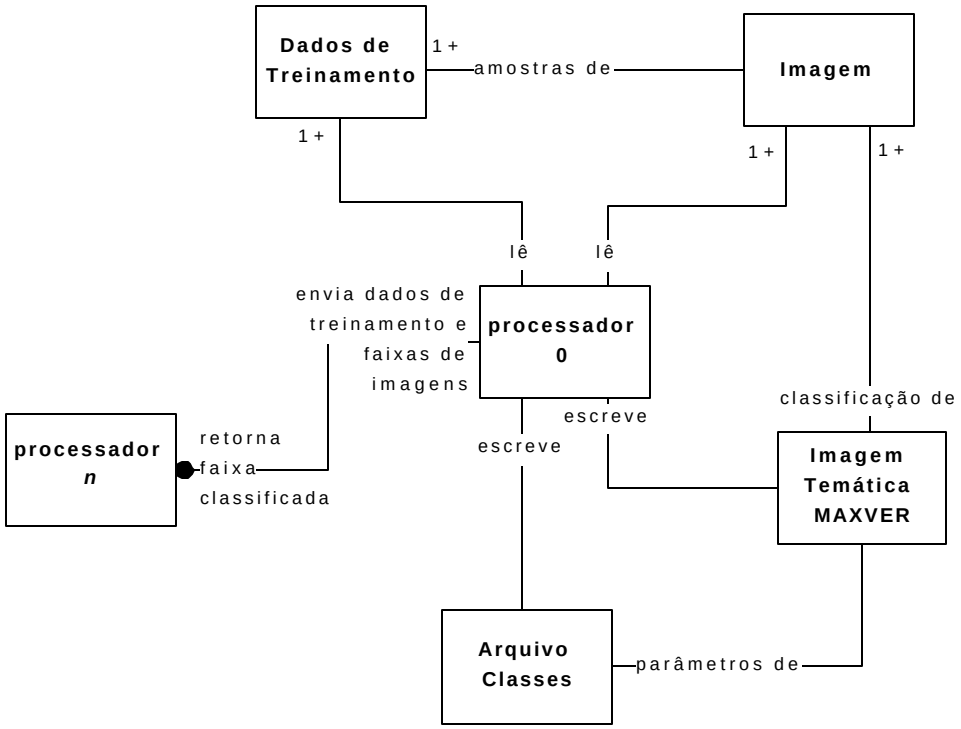
A.7 DICIONÁRIO DE DADOS DO MODELO FUNCIONAL DA CLASSIFICAÇÃO SEQUENCIAL

Processo	Descrição
Ler Amostras	Lê arquivo gerado pelo SPRING com amostras de treinamento para classificação.
Cálculo da Matriz de Transformação	Calcula matriz de transformação a ser utilizada no cálculo da Distância de Mahalanobis.
Classificação MAXVER	Executa classificação da imagem utilizando algoritmo MAXVER.
Classificação ICM	Executa classificação da imagem utilizando algoritmo ICM.
Escrever Imagem	Grava imagem classificada.
Cálculo de Beta	Calcula parâmetro de atratividade para classificação ICM.
Verificar Critério de Parada	Verifica se o algoritmo ICM deve proceder em mais uma iteração ou não, de acordo com critérios de parada definidos pelo usuário (percentual de modificações de pixels e/ou número máximo de iterações).

APÊNDICE B

DIAGRAMAS E DICIONÁRIOS DE DADOS DOS PROGRAMAS PARALELOS

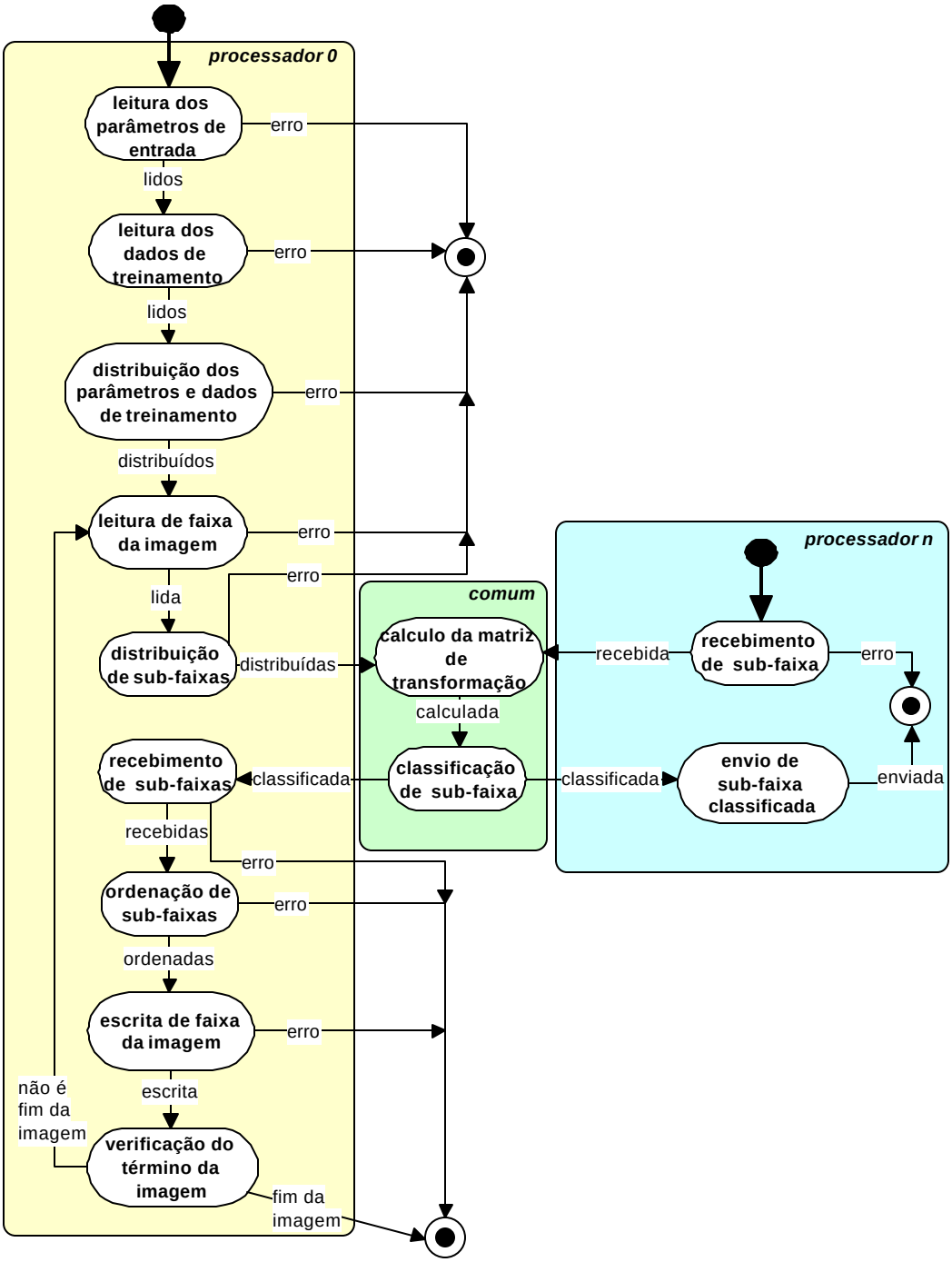
B.1 DIAGRAMA DE OBJETOS PARA CLASSIFICAÇÃO MAXVER EM PARALELO



B.2 DICIONÁRIO DE DADOS DO MODELO DE OBJETOS DA CLASSIFICAÇÃO MAXVER EM PARALELO

Objeto	Descrição
Arquivo de Classes	Arquivo gerado pelo aplicativo de classificação com parâmetros das classes analisadas.
Dados de Treinamento	Arquivo gerado no SPRING com amostras de treinamento.
Imagem	Bandas da imagem original a ser classificada.
Imagem Temática MAXVER	Banda temática gerada por aplicativo de classificação MAXVER
Processador 0	Processador responsável por interface com usuário, além de leitura e distribuição das faixas das bandas da imagem entre os demais processadores (se existirem). Também executa classificação MAXVER sobre sub-faixa da imagem. Finalmente, recebe sub-faixas classificadas dos demais processadores, organizando-as e salvando-as em disco.
Processador *n*	Cada um dos processadores envolvidos na classificação da imagem, com exceção do processador 0. Cada um recebe a sua sub-faixa e executa classificação MAXVER sobre ela, retornando os resultados para o processador 0.

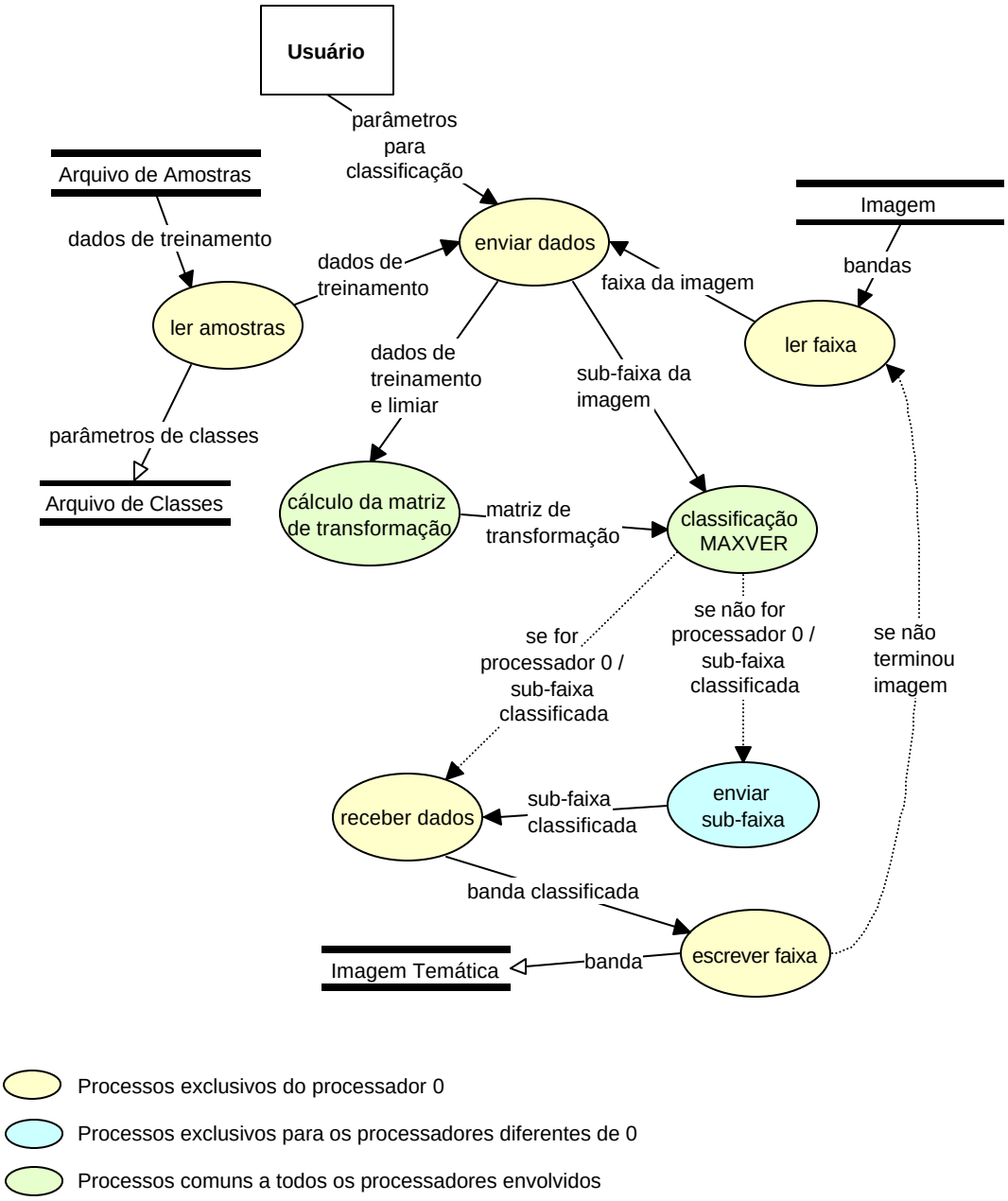
B.3 DIAGRAMA DE ESTADOS PARA CLASSIFICAÇÃO MAXVER EM PARALELO



B.4 DICIONÁRIO DE DADOS DO MODELO DINÂMICO DA CLASSIFICAÇÃO MAXVER EM PARALELO

Estado	Descrição
Cálculo da matriz de transformação	Cada processador executa o cálculo da matriz de transformação (necessária para o processo de classificação MAXVER) para sua sub-faixa de imagem.
Classificação de sub-faixa	Cada processador executa classificação MAXVER sobre sua sub-faixa de imagem.
Distribuição de sub-faixas	Divide a faixa retangular, lida da imagem, em *n* sub-faixas (correspondente ao número de processadores envolvidos na classificação) e as envia para os respectivos processadores.
Distribuição dos parâmetros e dados de treinamento	Processador 0 distribui parâmetros e dados de treinamento lidos para os demais processadores (se existirem) envolvidos no processo.
Envio de sub-faixa classificada	Após classificação MAXVER para sub-faixa de imagem, cada processador a envia para o processador 0.
Escrita de faixa da imagem	Escreve faixa retangular da imagem classificada em disco.
Leitura de faixa da imagem	Lê faixa retangular de dados das bandas da imagem.
Leitura dos dados de treinamento	Lê arquivo contendo amostras de treinamento.
Leitura dos parâmetros de entrada	Lê nome do arquivo de amostras e limiar de amostragem para classificação.
Ordenação de sub-faixas	As sub-faixas são ordenadas a fim de formar a faixa retangular da imagem classificada a ser armazenada em disco.
Recebimento de sub-faixa	Processador recebe sub-faixa enviada pelo processador 0.
Recebimento de sub-faixas	Processador 0 recebe as sub-faixas classificadas, enviadas pelos outros processadores.
Verificação do término da imagem	Verifica se todas as faixas da imagem foram lidas e classificadas.

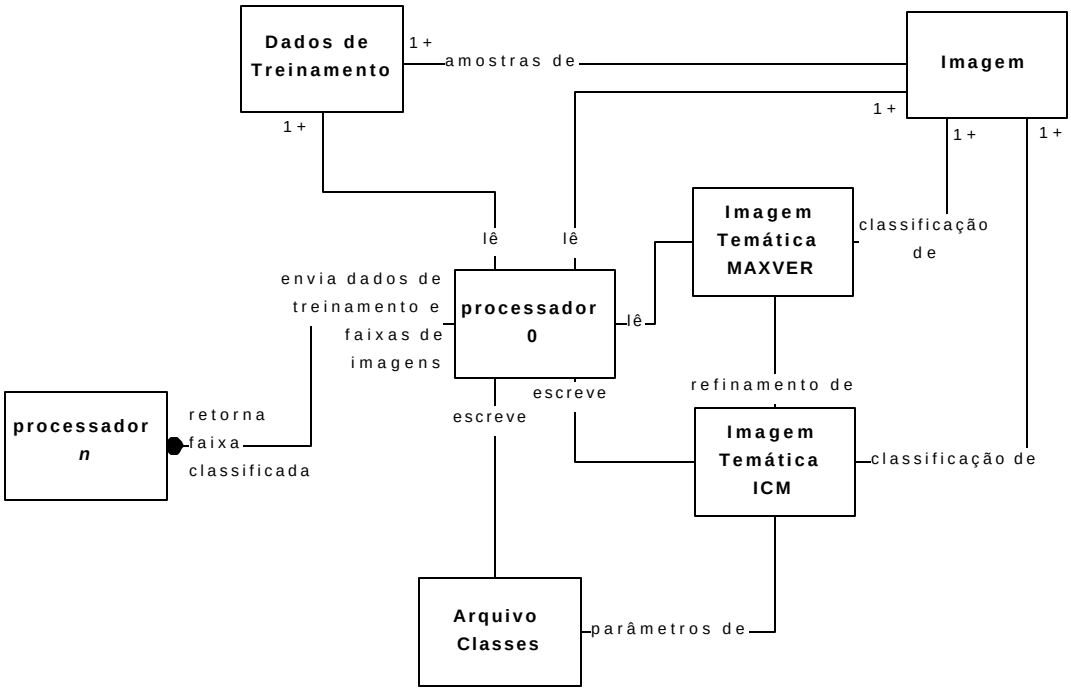
B.5 DIAGRAMA DE FLUXO DE DADOS PARA CLASSIFICAÇÃO MAXVER EM PARALELO



B.6 DICIONÁRIO DE DADOS DO MODELO FUNCIONAL DA CLASSIFICAÇÃO MAXVER EM PARALELO

Processo	Descrição
Cálculo da Matriz de Transformação	Calcula matriz de transformação a ser utilizada no cálculo da Distância de Mahalanobis.
Classificação MAXVER	Executa classificação em sub-faixa da imagem utilizando algoritmo MAXVER.
Enviar dados	Processador 0 envia dados de treinamento, obtidos a partir de arquivo gerado no SPRING, para os outros processadores envolvidos na classificação. Também envia limiar fornecido pelo usuário.
Enviar sub-faixa	Envia sub-faixa classificada da imagem para processador 0, para ser gravada em disco.
Escrever faixa	Grava em disco faixa classificada da imagem.
Ler Amostras	Lê arquivo gerado pelo SPRING com amostras de treinamento para classificação.
Ler faixa	Lê faixa de imagem, para sub-divisão e envio aos respectivos processadores. Este processo se repete após gravação de faixa classificada, enquanto não chegar ao fim da imagem.
Receber dados	Recebe e ordena todas as sub-faixas enviadas pelos processadores para gravar faixa de imagem.

B.7 DIAGRAMA DE OBJETOS PARA CLASSIFICAÇÃO ICM EM PARALELO

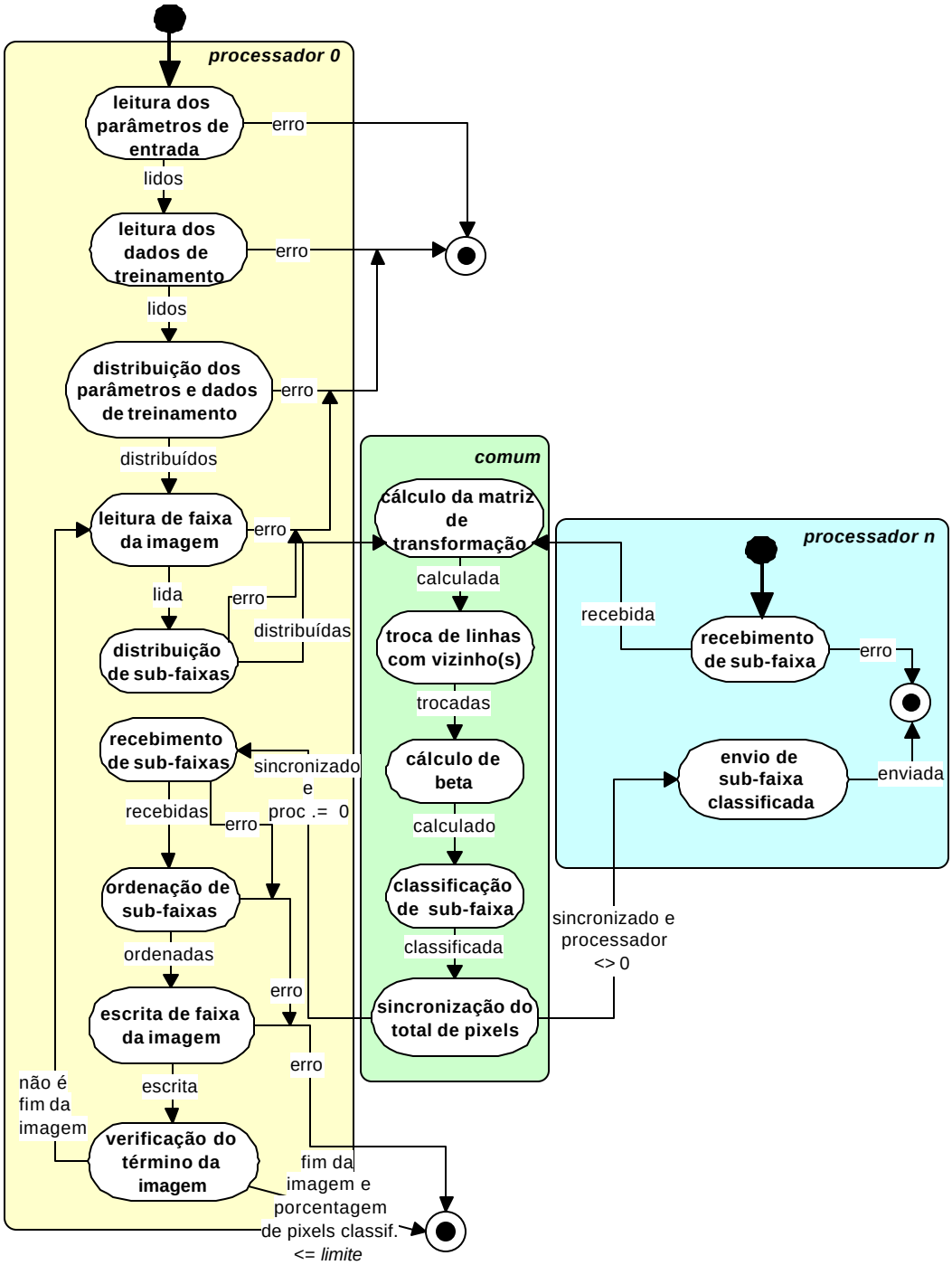


B.8 DICIONÁRIO DE DADOS DO MODELO DE OBJETOS DA CLASSIFICAÇÃO ICM EM PARALELO

Objeto	Descrição
Arquivo de Classes	Arquivo gerado pelo aplicativo de classificação com parâmetros das classes analisadas.
Dados de Treinamento	Arquivo gerado no SPRING com amostras de treinamento.
Imagem	Bandas da imagem original a ser classificada.
Imagem Temática ICM	Banda temática gerada por aplicativo de classificação ICM.
Imagem Temática MAXVER	Banda temática gerada por aplicativo de classificação MAXVER.
Processador 0	Processador responsável por interface com usuário, além de leitura e distribuição das sub-faixas das bandas da imagem entre os demais processadores (se existirem). Também executa classificação ICM sobre sub-faixa da imagem. Finalmente recebe sub-faixas classificadas dos demais processadores, organizando-as e salvando-as em disco.
Processador *n*	Cada um dos processadores envolvidos na classificação da imagem, com exceção do processador 0. Cada um recebe a sua sub-faixa e executa classificação ICM sobre ela, retornando-a para o processador 0.

OBS: Devido ao algoritmo de classificação ICM ser contextual, todos os processadores relacionam-se entre si trocando linhas classificadas das bordas das sub-faixas da imagem em questão.

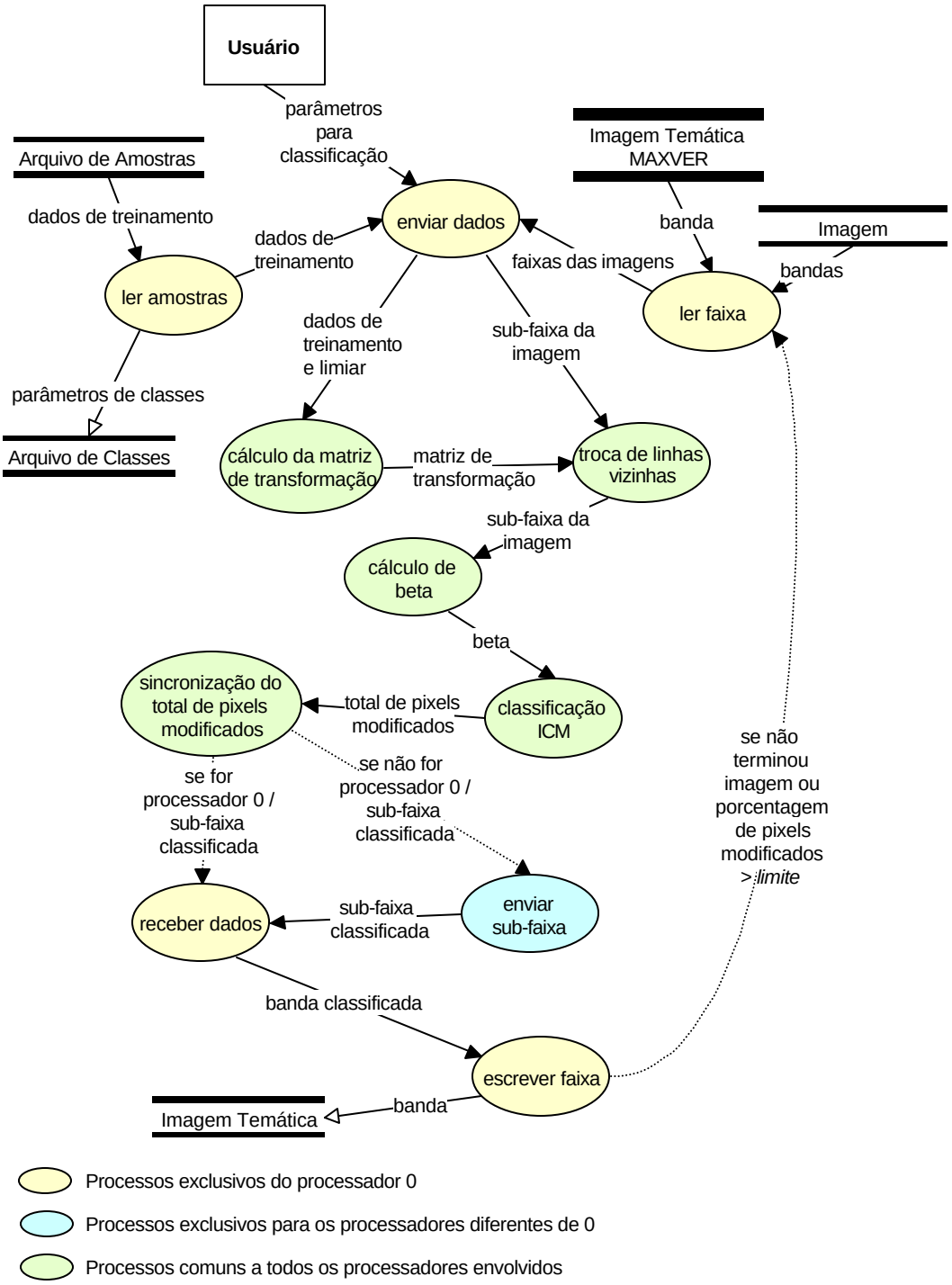
B.9 DIAGRAMA DE ESTADOS PARA CLASSIFICAÇÃO ICM EM PARALELO



B.10 DICIONÁRIO DE DADOS DO MODELO DINÂMICO DA CLASSIFICAÇÃO ICM EM PARALELO

Estado	Descrição
Cálculo da matriz de transformação	Cada processador executa o cálculo da matriz de transformação (necessária p/ o proc. de classif. ICM) para sua sub-faixa de imagem.
Cálculo de beta	Calcula parâmetro de atratividade β, para a sub-faixa em questão.
Classificação de sub-faixa	Cada processador executa classificação ICM sobre sua sub-faixa de imagem.
Distribuição de sub-faixas	Divide a faixa retangular, lida da imagem, em *n* sub-faixas (corresp. ao núm. de processadores envolvidos na classificação) e as envia para os respectivos processadores. Se a divisão não for exata, as linhas restantes são sempre enviadas para o último processador.
Distrib. dos parâ-metros e dados de treinamento	Processador 0 distribui parâmetros e dados de treinamento lidos para os demais processadores (se existirem) envolvidos no processo.
Envio de sub-faixa classificada	Após classificação ICM para sub-faixa de imagem, cada processador a envia para o processador 0.
Escrita de faixa da imagem	Escreve faixa retangular da imagem classificada em disco.
Leitura de faixa da imagem	Lê faixa retangular de dados das bandas da imagem, considerando as linhas vizinhas da faixa em questão.
Leitura dos dados de treinamento	Lê arquivo contendo amostras de treinamento.
Leitura dos parâmetros de entrada	Lê nome do arquivo de amostras, limiar de amostragem e porcentagem mínima de pixels classificados (critério de parada).
Ordenação de sub-faixas	As sub-faixas são ordenadas a fim de formar a faixa retangular da imagem classificada a ser armazenada em disco.
Recebimento de sub-faixa	Processador recebe sub-faixa enviada pelo processador 0.
Recebimento de sub-faixas	Processador 0 recebe as sub-faixas classificadas, enviadas pelos outros processadores.
Sincronização do total de pixels	Soma do número de pixels modificados durante classificação para cada sub-faixa de cada processador. Este procedimento garante que todos os processadores (na mesma iteração) interrompam o processo quando o número total de pixels modificados seja menor ou igual ao definido pelo usuário.
Troca de linhas com vizinho(s)	Recebe linhas da borda das sub-faixas dos processadores vizinhos e envia linhas da borda da sub-faixa a ser classificada. Este processo garante a classificação contextual.
Verificação do término da imagem	Verifica se todas as faixas da imagem foram lidas e classificadas. Também verifica porcentagem de pixels classificados. Se for menor ou igual ao especificado pelo usuário, termina classificação.

B.11 DIAGRAMA DE FLUXO DE DADOS PARA CLASSIFICAÇÃO ICM EM PARALELO



B.12 DICIONÁRIO DE DADOS DO MODELO FUNCIONAL DA CLASSIFICAÇÃO ICM EM PARALELO

Processo	Descrição
Cálculo da Matriz de Transformação	Calcula matriz de transformação a ser utilizada no cálculo da Distância de Mahalanobis.
Cálculo de beta	Calculo do parâmetro de atratividade β, considerando a sub-faixa em questão.
Classificação ICM	Executa classificação em sub-faixa da imagem utilizando algoritmo ICM.
Enviar dados	Processador 0 envia dados de treinamento, obtidos a partir de arquivo gerado no SPRING, para os outros processadores envolvidos na classificação. Também envia limiar fornecido pelo usuário.
Enviar sub-faixa	Envia sub-faixa classificada da imagem para processador 0, para ser gravada em disco.
Escrever faixa	Grava em disco, faixa classificada da imagem. Após gravação verifica se limite de porcentagem de pixels modificados foi atingido. Se não foi, a classificação continuará em mais uma iteração.
Ler Amostras	Lê arquivo gerado pelo SPRING com amostras de treinamento para classificação.
Ler faixa	Lê faixa de imagem, para sub-divisão e envio aos respectivos processadores. Este processo se repete após gravação de faixa classificada, enquanto não chegar ao fim da imagem e não for atingido o limite mínimo de porcentagem de pixels modificados na classificação.
Receber dados	Recebe e ordena todas as sub-faixas enviadas pelos processadores para gravar faixa de imagem.
Sincronização do total de pixels modificados	O número de pixels modificados por cada processador em cada sub-faixa é somado e enviado a todos os processadores. Este procedimento garante que todos os processadores (na mesma iteração) interrompam o processo quando o número total de pixels modificados seja menor ou igual ao definido pelo usuário.
Troca de linhas vizinhas	Cada processador recebe linhas da borda das sub-faixas dos processadores vizinhos e envia linhas da borda da sub-faixa a ser classificada. Este processo garante a classificação contextual.