UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Classificação de Imagens Digitais por Textura usando Redes Neurais

por

FELIPE LIBERMAN

Dissertação submetida à avaliação, como requisito
parcial para a obtenção do grau de Mestre
em Ciência da Computação

Prof. Dr. Paulo Martins Engel
Orientador

Porto Alegre, 27 de novembro de 1997

Agradecimentos

Agradeço a todos aqueles que contribuíram para a realização deste trabalho, em especial:

Aos meus pais, Bernardo e Ligia, pelo despertar da curiosidade desde a infância, pelo apoio durante todo trabalho e pela incansável dedicação durante todas etapas da minha vida.

À minha noiva Rita, pela influência e visão otimista as quais me motivaram a realização do curso.

Ao meu orientador, professor Paulo Engel, pelo fundamental apoio técnico e dicas, responsável pela qualidade desse trabalho.

Ao CPGCC pela oportunidade oferecida e a todos seus funcionários pela dedicação e competência.

À todos professores do Instituto, em especial à professora Mara Abel pela ajuda na classificação das imagens de minerais e ao professor Vítor Haertel do CEPSRM pela indicação de referencias bibliográficas.

Aos colegas da pós-graduação, em especial aos amigos Newton C. K. Borges e Joelson Coelho pelo companheirismo, pelas inúmeras discussões, sugestões e experiências compartilhadas.

Sumário

Lista de abreviaturas e símbolos

RN	Rede Neural.
BPN	BackPropagation Network.
MLP	Multi-Layer Percepton.
IMASEG	Sistema de classificação de Imagens.
INPE	Instituto Nacional de Pesquisas Espaciais.
RGB	Red-Green-Blue.
OCR	Optical Caracter Recognition.
XOR	Operação booleana do OU-Exclusivo.
Mat.Cooc.	Matriz de Coocorrência dos níveis de cinza.
tanh(x)	Tangente hiperbólica de x.
D.Padrão	Desvio-padrão.
s	Desvio-padrão.
s^2	Variância.
h	Uniformidade.
e	Entropia.
k	Contraste.

Lista de figuras

Lista de tabelas

Resumo

Este trabalho apresenta um estudo sobre a classificação de imagens digitais através da textura com o auxílio de redes neurais. São utilizadas técnicas e conceitos de duas áreas da Informática: O Processamento de Imagens Digitais e a Inteligência Artificial. São apresentados os principais tópicos de processamento de imagens, as principais aplicações em tarefas industriais, reconhecimento de padrões e manipulação de imagens, os tipos de imagem e os formatos de armazenamento. São destacados os atributos da imagem, a textura e sua quantificação através da matriz de coocorrência dos níveis de cinza. Também apresenta-se alguns sistemas computacionais disponíveis para processamento de imagens. Na área de Inteligência Artificial, o enfoque é para técnicas computacionais inteligentes, mais especificamente as Redes Neurais. É feita uma breve apresentação da área, incluindo seu histórico e suas principais aplicações. As redes neurais são classificadas quanto ao tipo de treinamento, à regra de aprendizado, à topologia da rede e quanto ao tipo de interconexão dos neurônios. O modelo BPN (BackPropagation Network) é visto com maior detalhe, visto ser utilizado na implementação do sistema IMASEG (Sistema para Classificação de Imagens) que faz parte desse trabalho. O BPN é descrito quanto ao seu funcionamento, a forma de aprendizado e as respectivas equações utilizadas. O sistema IMASEG foi desenvolvido com o objetivo de implementar as técnicas propostas para a classificação de imagens utilizando textura e redes neurais. Seu funcionamento e algoritmos utilizados são detalhados e ao final, apresenta-se os resultados obtidos com a respectiva análise.

A classificação de imagens é uma das principais etapas no processamento de imagens digitais. Dado um conjunto de classes e um padrão apresentado como entrada para o sistema, o problema consiste em decidir a que classe o padrão pertence. Deve haver a alternativa de rejeição do padrão. Podemos extrair da imagem atributos espectrais, espaciais e de contexto. Por serem mais facilmente quantificáveis, a maioria dos sistemas tradicionais utiliza apenas atributos espectrais para caracterizar uma imagem. Essa abordagem é muito utilizada em imagens multiespectrais. Entretanto, utilizando apenas atributos espectrais, não se obtém uma informação completa sobre a imagem, pois não são levados em consideração as relações espaciais entre seus pixels, bem como a forma de objetos. A textura, atributo espacial, é ainda pouco utilizada, visto que tem origem na sensação visual causada pelas variações tonais existentes em uma determinada região da imagem, tornando difícil sua quantificação. Neste trabalho, é feito um estudo sobre a utilização dos atributos espaciais da imagem no seu processamento. É feita uma análise do comportamento de cinco deles: média, desvio-padrão, uniformidade, entropia e contraste, todos extraídos de janelas pertencentes à uma classe. A uniformidade, entropia e contraste provém da matriz de coocorrência dos níveis de cinza. Através do cálculo do valor desses atributos em diversas imagens, constata-se que existem algumas importantes relações entre eles.

A partir da análise dos diferentes modelos de redes neurais e das diversas formas de quantificar a textura de uma imagem, é proposto um sistema computacional com o objetivo de classificar imagens. Esse sistema faz o processamento das imagens através de uma janela móvel. O usuário deve escolher o tamanho para a janela: 3×3, 5×5 ou 7×7 *pixels*. Essa escolha irá depender do tipo e da granularidade da textura que a imagem contém. Em seguida, utilizando a janela, deve selecionar amostras representativas de

cada textura (classe) presente na imagem que se deseja classificar. O sistema então, encarrega-se de treinar a rede neural utilizando as amostras selecionadas pelo usuário. Para realizar o treinamento, é necessário encontrar uma forma de mapear os dados da realidade para a rede neural. Essa tarefa nem sempre é trivial. Nesse sistema, são propostas duas abordagens para realizar essa tarefa. Na primeira, o mapeamento é feito através do cálculo das feições da média, desvio-padrão e uniformidade, sendo esse último obtido da matriz de coocorrência. Essas feições, após um escalonamento para a mesma faixa de valores, serão os parâmetros de entrada para a rede neural. Na segunda abordagem, o mapeamento é direto, ou seja, o valor de cada pixel, após o escalonamento, corresponde a uma entrada da rede neural. Após a etapa de treinamento, a imagem é processada por inteiro, fazendo-se uma varredura com a janela, gerando como saída uma imagem temática na qual cada tema representa uma das texturas existentes na imagem original.

Para testar o sistema IMASEG, foram geradas várias imagens sintéticas com 256 níveis de cinza. Deste total, foram selecionadas 6 imagens para serem apresentadas nesse trabalho. Elas são representativas das diversas situações que podem ocorrer em relação aos valores da média, desvio-padrão e uniformidade. Cada imagem original é processada pelas duas abordagens, gerando duas imagens de saída. É feita uma análise quantitativa e qualitativa dos resultados obtidos, apontando-se as prováveis causas de sucessos e problemas encontrados. Conclui-se que a classificação por textura atinge o objetivo proposto e é muito útil no processamento de imagens, levando-se em consideração os bons resultados obtidos.

Palavras-chave: Processamento de imagens, Textura, Matriz de Coocorrência, Redes Neurais, Rede Neural Backpropagation.

TITLE: "Classification of digital images through texture with the aid of neural networks"

Abstract

This paper is a study about the classification of digital images through texture with the aid of neural networks. The techniques and concepts from the field of Computer Science employed are: Digital Images Processing and Artificial Intelligence. The focus in Image Processing is on its main application in industrial tasks, pattern recognition and image manipulation, the types of images and the storing formats. The specific aspects analyzed are image attributes, texture and its quantification through the Coocurrence Matrix. Several available computing systems for image classification are presented. In Artificial Intelligence, the attention is concentrated on intelligent computational systems, more specifically on the neural networks which are briefly introduced. The subject´s historical data and its main application are also addressed. The neural networks are classified according to the type of training, the learning rules, the network topology and the interconnection of neurones. The BPN model (Back Propagation Network) is examined more closely since it is employed in the implementation of the IMASEG system (classifying images system) which is part of this study. The BPN system is described in according to its functioning capacities, the learning method and the respective equations utilized. The IMASEG system was developed with the specific aim of implementing the techniques of image classification. Throughout the paper, the system´s operation and related algorithms are presented to the reader, as well as the results obtained and the analysis performed provided in the end of the paper

The image classification is one of the principal steps for the processing of digital images. It consists to decide of which class the pattern belong. It can refuse the pattern. We can extract spectral, spatial and contextual image's attributes. Because they are easily quantified, a major part of the traditional systems of image processing employ only the spectral attributes to work the images and are, therefore, extensively used in the processing of multispectral images. However, the exploration of images through spectral attributes is not enough to provide a complete recognition of the image since information such as spatial relations among its pixels as well as the form of objects are not taken into consideration. The use of image processing with spatial attributes is also considered in this paper. Texture is still not a commonly employed attribute. This is due to the fact that its based on visual sensation which is produced by the existing tonal variations of a specific image region, making its quantification a difficult task to perform. A behavior analysis of the spatial attributes under consideration in this paper are the following: mean, standard deviation, uniformity, entropy and contrast. These five attributes were all taken from windows belonging to a single class. Uniformity, entropy and contrast are issued from the gray level coocurrence matrix. Via a calculation of the value of these attributes is observed that there is an important relationship among them.

This paper proposes a system of image classification based on the analysis of different models of neural networks and also through the analysis of the diverse ways of quantifying the texture of an image. This system performs the image processing through a shifting window. Then, the user must choose the window's size from among the following dimensions: 3×3, 5×5 or 7×7 pixels. The choice will vary depending on the

type and on the image's texture granularity. The selection of meaningful samples of each texture (class) present in the image one wishes to classify is the next step in the process. The system, then, is in charge of training the neural networks by applying the user's selected samples. In order to perform the training, it is necessary to first establish a way of mapping the data reality to the neural network, oftentimes a difficult task. In this system two approaches are proposed for the execution of this task. In the first, the mapping is done through the calculation of the mean, standard deviation and uniformity features. The last item is obtained from the coocurrence matrix. After these features have been scaled to the same value band, they will become the input to the neural networks. In the second approach, it is expected that the neural network will be able to extract textures attributes without executing an explicit calculation exercise. After the training phase, the image is completely processed through a window scanning generating a thematic image as the output onto which each theme will represent one of the texture's original image.

In order to verify the adequacy of the IMASEG system, several synthetical graylevel images were created. Of these, 7 images were chosen as objects for this analysis, representing the various possible situations that might occur in relation to the average, standard deviation and uniformity. Each original image is processed in according with these two chosen approaches, thus generating two images as outputs, as well as a quantitative and qualitative analysis of the obtained results, pointing to the probable successes and failures generated. The final conclusion is that the classification through texture partially attains the proposed objectives and can be very useful in the processing of images, serving as an aid in the traditional classification process.

Keywords: Image processing, Texture, Coocurrence Matrix, Neural Networks, Backpropagation Network.

1. Introdução

A área de Processamento de Imagens Digitais é interdisciplinar, utilizando conceitos da informática, física (ótica) e eletrônica e está adquirindo uma importância cada vez maior pois está se tornando extremamente útil em diversas outras áreas do conhecimento como sensoriamento remoto, astronomia, medicina, artes e outros. Seu objetivo é modificar, analisar e manipular imagens digitais, originalmente analógicas, a partir de um computador. A vontade e a necessidade crescente de automatização de atividades cotidianas evidenciam um envolvimento crescente das ferramentas de processamento de imagens em grande número de domínios. Alguns exemplos de aplicações são: tarefas industriais (pilotagem de robô, inspeção visual), reconhecimento de padrões (imagens, caracteres-OCR, assinaturas) e reconstrução tridimensional (tomografia de ressonância magnética).

A segmentação de uma imagem é um procedimento pelo qual, a partir de uma imagem observada, obtém-se uma nova imagem, onde a cada posição é atribuído um único rótulo, entre K possíveis. Pode-se pensar nesses K rótulos como diferentes cores. A segmentação deve tentar identificar (ou rotular) regiões da imagem onde os *pixels* tenham características similares. Apesar do ponto de partida da segmentação ser as características similares do pixel, o que se quer é identificar regiões ou áreas da imagem que caracterizam um determinado objeto, ou tema. A classificação é uma etapa posterior, onde a cada rótulo associa-se uma classe. Considerável esforço tem sido dedicado à solução do problema de reconhecimento e caracterização de objetos presentes em uma imagem. Até a década de 70, predominou o uso de técnicas óticas de processamento. A partir do início dos anos 80, com os avanços verificados na microeletrônica e no desenvolvimento de arquiteturas paralelas de processamento, as técnicas digitais passaram a ser mais empregadas. Atualmente, o amadurecimento das técnicas computacionais inteligentes, como Sistemas Especialistas, Lógica Nebulosa, Redes Neurais e Algoritmos Genéticos, tem permitido novas abordagens para esse problema. As técnicas inteligentes fazem uso metafórico dos conceitos, princípios e mecanismos fundamentais dos sistemas naturais, procurando capturar, tanto na teoria como na prática, os algoritmos encontrados na natureza. A complexidade do problema de reconhecimento e de classificação de imagens, que dificilmente pode ser abordado em termos algorítmicos, tem tornado o uso dessas técnicas cada vez mais freqüente, especialmente as Redes Neurais Artificiais.

As Redes Neurais têm se apresentado como uma alternativa aos sistemas tradicionais de inteligência artificial. Uma das suas principais aplicações é no reconhecimento de padrões. Em termos gerais, o reconhecimento de padrões é a área que compreende a identificação ou classificação de medidas de informação em categorias. Categorias têm a característica de representar entidades ou padrões de informação que apresentam similaridades. Reconhecimento de padrões é composto de um conjunto de técnicas e abordagens que são usadas de forma integrada na solução de diversos problemas práticos. Nesse contexto, as imagens digitais constituem-se em grande fonte de pesquisa nessa área. Outros problemas práticos incluem: processamento e análise de sinais, reconhecimento de voz, face e caracteres, classificação e identificação de impressões digitais, diagnose médica, monitoramento e análise de sinais biológicos. As Redes Neurais são dispositivos não-lineares, inspirados na funcionalidade dos neurônios biológicos, aplicados no reconhecimento de padrões, na

otimização e na previsão de sistemas complexos. A habilidade em formar mapeamentos não-lineares torna as redes neurais prósperas nessas aplicações [TRE89]. Redes Neurais são compostas por diversas unidades computacionais paralelas, interconectadas parcial ou totalmente. Cada uma dessas unidades (neurônios artificiais) efetua um certo número de operações simples e transmite seus resultados às unidades vizinhas com as quais possui conexão. Através de um processo de treinamento, as redes neurais passam a ser capazes de reconhecer padrões, mesmo que os dados utilizados nesse treinamento sejam não-lineares, incompletos ou até mesmo contraditórios. A habilidade de manipular esses dados imprecisos faz com que as redes neurais sejam extremamente eficazes em tarefas onde especialistas não estão a disposição ou um conjunto de regras não pode ser facilmente formulado.

Como mencionado, as redes neurais são cada vez mais utilizadas em reconhecimento de padrões. Alguns trabalhos nessa área podem ser citados. Perelmuter et all utilizaram a rede neural Backpropagation para classificar peças mecânicas. O sistema realiza um pré-processamento a fim de obter coeficientes de entrada para a rede neural. Tais coeficientes permitem a diferenciação entre diversas imagens e fornecem invariância com relação à rotação, escalonamento e translação. O desempenho do sistema é dependente das condições de captura da imagem e de uma avaliação mais precisa dos coeficientes relevantes [PER95].

O objetivo principal desse trabalho é o desenvolvimento de um sistema de processamento de imagens através de feições de textura da imagem e com auxílio de Redes Neurais. A motivação inicial desse trabalho foi a constatação da necessidade e utilidade de um sistema para processar automaticamente imagens onde a textura tem grande importância. Um exemplo são as imagens médicas, em especial, as do ventrículo esquerdo do coração, no qual a textura do interior (sangue) é diferente da textura do exterior do coração (tecido). Em seu trabalho, Sussner propõe um sistema automático capaz de traçar a borda do coração permitindo também calcular o seu volume. O processo tradicional é feito manualmente por um especialista. Sussner utilizou a matriz de coocorrência e uma rede neural, obtendo bons resultados [SUS95]. A partir desta idéia inicial, foram propostas algumas modificações para construir um sistema de classificação de imagens digitais.

Em processamento de imagens, a textura é um atributo ainda pouco utilizado no reconhecimento de cenas, já que ela advém da sensação visual causada pelas variações tonais existentes em uma determinada região da imagem, tornando difícil a sua quantificação. A maioria dos sistemas utiliza atributos espectrais da imagem. Neste trabalho, faz-se um estudo da potencialidade de classificação de imagens através da textura.

Os capítulos estão divididos da seguinte forma:

O capítulo 1 faz uma introdução do assunto, dando ao leitor uma visão geral do trabalho desenvolvido.

No capítulo 2, apresenta-se mais detalhes da área de processamento de imagens. Esta área é bastante ampla. Foi dado destaque aos tópicos de maior relevância para a compreensão do trabalho, como é o caso dos tipos de imagens digitais, as imagens monocromáticas/multiespectrais e principalmente a textura.

No capítulo 3, é feita uma breve descrição da área de Redes Neurais para que o leitor sem experiência nessa área possa compreender como e por que essa técnica foi utilizada no sistema desenvolvido. Da mesma forma que a área de processamento de imagens, essa área também é muito ampla e também procurou-se direcionar o assunto para os tópicos de maior relevância. A rede BPN (Backpropagation Network) é

abordada com mais detalhes pois esse modelo de rede neural é utilizado no sistema implementado nesse trabalho. São vistos a topologia dessa rede e as equações de aprendizado e propagação utilizadas.

A partir do capítulo 4, são apresentados as propostas originais do autor, descrevendo com detalhes o funcionamento do sistema IMASEG, desenvolvido para avaliar a proposta de classificação de imagens usando textura e redes neurais. Esse sistema foi escrito em linguagem C podendo ser anexado a sistemas mais amplos de processamento de imagens como por exemplo o sistema IRENE desenvolvido no Instituto de Informática.

Os resultados obtidos com os testes do sistema proposto e a sua análise estão descritos no capítulo 5. As conclusões são citadas no capítulo 6.

No anexo-A1, são listados os principais trechos do código fonte a fim de que os resultados possam ser repetidos e no final temos a referência bibliográfica.

2. Processamento de imagens digitais

2.1 Introdução

O processamento de imagens digitais, juntamente com a computação gráfica, pertencem à área de Processamento Gráfico. Em linha gerais, o primeiro tem o objetivo de analisar imagens e padrões enquanto que o segundo busca sintetizar e visualizar as mesmas. A área de processamento de imagens nasceu com a motivação criada pelos programas espaciais da NASA nos EUA na década de 60 e hoje é uma área interdisciplinar com aplicações em diversas áreas.

2.2 Aplicações de processamento de imagens

A importância da visão na vida do ser humano é uma evidência para cada um de nós. A vontade e a necessidade crescente de automatização de atividades cotidianas evidenciam um envolvimento crescente das ferramentas de processamento de imagens em um grande domínio. Os processos, utilizados para tal, podem variar de acordo com o objetivo e a aplicação. Abrangem desde uma simples tarefa de melhoramento de imagens adquiridas, até atingir a dificuldade de interpretação de uma imagem. Aplicações de processamento e análise de imagens podem ser classificadas a partir de critérios como a possibilidade de um conhecimento prévio do contexto ou não, a necessidade de processar em tempo real, etc. A seguir, são apresentadas exemplos de aplicações que envolvem técnicas de processamento de imagens [FAC93] [WAT93].

2.2.1 Tarefas industriais

De forma geral, o desenvolvimento de um projeto industrial que exija a utilização de técnicas de processamento de imagens supõe um importante conhecimento inicial do produto (normas de concepção, regras de produção, etc) e exige técnicas rápidas para atender processos em tempo real.

-Pilotagem de robô

Controlar um robô é uma das tarefas que requer rapidez e precisão. Um robô é usado para manipular peças que podem ser ordenadas, ou montadas com outros tipos de peças. Estas podem já estar acondicionadas em locais previamente conhecidos ou não. O alvo do robô, atingido com auxílio de visão computacional, pode envolver a escolha de uma peça entre várias outras ou a determinação de uma trajetória para evitar obstáculos.

-Inspeção visual

O controle de qualidade de um produto é uma tarefa essencial no domínio industrial. O processo de inspeção implica em medir determinadas propriedades de um produto, como dimensões geométricas, superfícies, posição, orientação, etc. Um dos alvos da visão computacional consiste em analisar, sem contato com o produto, os

defeitos cuja detecção seria impossível ou muito difícil por outros métodos. A inspeção automatizada, mediante técnicas de processamento de imagens, possibilita a quantificação de propriedades e coleta de dados sobre o produto que um inspetor humano não é capaz de realizar, pois além de ser tedioso ou mesmo não realizável, permite uma realimentação constante e imediata do processo de manufatura.

2.2.2 Reconhecimento de padrões

A exigência crescente de informatização e automatização de tarefas humanas repetitivas e/ou cansativas, levou os pesquisadores a desenvolver ferramentas específicas para atender projetos cujo conhecimento inicial é pobre e/ou incompleto e onde um certo grau de inteligência é importante. O reconhecimento de um padrão inscreve-se neste tipo de tarefa. Dentro dos assuntos de pesquisa em reconhecimento de padrões, podemos citar:

-Processamento de imagens de satélite.

As informações obtidas através de imagens de sensoriamento remoto são de grande valia para diversas e importantes aplicações, entre as quais destacamos a avaliação de desflorestamento, a análise de cobertura vegetal, o suporte à previsão de safras, a análise de uso do solo e o monitoramento ambiental em geral. Atualmente, grande parte da superfície terrestre encontra-se imageada por satélites, como o Landsat 5-TM (americano) ou o SPOT (francês). No entanto, o grande volume e complexidade das informações geradas tornam o processo de interpretação das imagens lento e de alto custo, quando executado integralmente por foto-intérpretes pelo processo manual tradicional. Neste cenário, a utilização de ferramentas de auxílio à interpretação de imagens é imprescindível.

-Processamento de imagens médicas.

As imagens médicas também podem ser processadas em grande escala por sistemas automáticos. Um exemplo disso, são as imagens de ecocardiograma do ventrículo esquerdo do coração. Detectando a borda do ventrículo, pode-se calcular o volume do coração, evitando-se a análise manual sujeita a erros de um especialista.

-Reconhecimento de caracteres (OCR- Optical Character Recognition).

Os recentes progressos nas pesquisas permitem o desenvolvimento de sistemas de leitura automática, já disponíveis no mercado. A dificuldade de uma tal pesquisa reside no grande número de tipos de letras usados e na necessidade de incluir o conhecimento sintático, semântico e pragmático do idioma em questão. Estes são problemas abertos não só em visão computacional, mas também em lingüística computacional. Um trabalho completo nessa área pode ser encontrado em [OSO91].

-Reconhecimento de impressões digitais (AFIS-Automatic Finger-Identification System).

Esta aplicação, cujo interesse para o bom funcionamento da sociedade é bastante útil, pode ser em parte automatizado. A dificuldade do confronto de uma impressão digital latente com um banco de dados reside na fragilidade da informação relevante contida na mesma, nas dificuldades de um levantamento cuidadoso do fragmento da impressão, e no tamanho e forma de organização do banco de dados que coloca restrições de desempenho no desenvolvimento de sistemas rápidos.

-Reconhecimento de assinaturas.

Possui grande interesse para entidades bancárias e burocráticas. Mas, a facilidade que o ser humano tem de mudar sua própria assinatura ou falsificar uma assinatura de outra pessoa tem sido o grande desafio para as pesquisas.

2.2.3 Reconstrução tridimensional

A constante evolução do desempenho dos computadores viabiliza a percepção tridimensional do mundo a partir de imagens bidimensionais. A principal aplicação dessa técnica também está na medicina. Com o aparecimento da tomografia de ressonância magnética, com imagens de alta definição e de nitidez nunca antes atingida até agora e com o aumento da capacidade das máquinas de armazenamento de imagens e da resolução dos dispositivos gráficos, houve uma explosão do uso de imagens na medicina. Da oftalmologia, passando pela radiologia à ortodontia, o processamento de imagens auxilia através de ferramentas de visualização tridimensionais de diferentes partes do corpo do ser humano e o desenvolvimento de sistemas de apoio ao diagnóstico médico.

2.2.4 Outros

O processamento de imagens também é utilizado na astronomia em fotometria, aplicações em tempo real, inspeção e calibração de imagens.

Outras aplicações são encontradas nas áreas artísticas e comerciais como produção de animações, efeitos especiais, digitalização de vídeo, acabamento de imagens, etc.

2.3 Imagem

Uma imagem pode ser descrita por uma função $f(x,y)$ da intensidade luminosa, sendo seu valor, em qualquer ponto de coordenadas espaciais (x,y), proporcional ao brilho da imagem naquele ponto. Já no caso de uma imagem que possui informações em intervalos ou bandas distintas de freqüência, é necessário uma função $f(x,y)$ para cada banda. É o caso de imagens coloridas padrão RGB, que são formadas pela informação de cores primárias, como o vermelho "RED", verde "GREEN" e o azul "BLUE". Para o processamento da imagem digitalizada, é fundamental representar sua informação num

formato adequado ao tratamento computacional. Uma imagem pode ser representada por uma matriz, em que os índices de linha e coluna referenciam o brilho médio amostrado no ponto correspondente da cena [FAC93].

2.3.1 Imagem da câmera

Uma imagem monocromática pode ser descrita por uma função matemática bidimensional $f(x,y)$ cujo valor indica a intensidade ao ponto (x,y). A função $f(x,y)$ representa o produto da interação entre a iluminância $i(x,y)$ que ilumina o objeto e as propriedades de reflectância ou de transmitância próprias do objeto, propriedades que podem ser representadas pela função $r(x,y)$ cujo valor exprime a fração de luz incidente que o objeto vai transmitir ou refletir ao ponto (x,y).

$$f(x,y) = i(x,y) \times r(x,y)$$

com:

$$I_1 < i(x,y) < I_2 \quad \text{unidade: } candela/m^2$$

$$0 < r(x,y) < 1$$

Exemplos:

$i(x,y)$ = 9000 tempo claro
= 10^3 tempo nublado
= 10^2 iluminação média de escritório
= 10^{-3} noite clara de lua cheia

$r(x,y)$ = 0,93 neve
= 0,80 parede branco-fosca
= 0,65 aço inoxidável
= 0,01 veludo preto

2.3.2 Imagem digital

De uma forma geral, os sensores fornecem um sinal analógico de vídeo correspondente à amostragem seqüencial da imagem. Dado que a grande maioria das técnicas de processamento de imagens é realizada de forma numérica em um computador (com o possível auxílio de processadores específicos), faz-se necessário a discretização do sinal analógico em uma imagem digital. A amostragem da imagem contida no sinal analógico é obtida por um conversor analógico-digital ou por um digitalizador. Nesse processo, os sinais devem ser amostrados espacialmente e quantizados em amplitude, de forma a obter a imagem digital [PAV82]. Da amostragem espacial, é gerada uma matriz com dimensão M×N, onde cada elemento da matriz é chamado de *pixel* da imagem. Após a quantização, cada *pixel* receberá um valor discreto. Um problema importante nesse processo é a escolha de uma boa amostragem e quantização. Para achar uma qualidade semelhante a de uma imagem de televisão, é preciso amostrar com resolução de 512×512 *pixels* e quantizar com 128 valores discretos (tons). Em geral, 64 tons são considerados como suficientes para o olho humano, mas a maioria dos sistemas de tratamento de imagem usa 256.

2.4 Formato Varredura × Formato Vetorial

O formato de varredura é composto por uma série de elementos de imagens, ou *pixels*, que cobrem uma área apresentada. Imagens de varredura são na maioria das vezes geradas por uma varredura periódica de um feixe de elétrons sobre a superfície de uma imagem que tem um determinado padrão. Um exemplo para isso é uma câmera de vídeo. Os *pixels* não estão necessariamente relacionados uns com os outros. O conceito de forma não é inerente nestas imagens. A apresentação dessas imagens é mais rápida pois os dispositivos de saída tem formato por varredura (exceto *plotter*), não necessitando de conversões feitas para mostrar imagens no formato vetorial.

Já as imagens com descrição vetorial envolvem o uso de segmentos de linha orientados ao invés de *pixels*. Uma imagem em formato vetorial é formada por formas geométricas que por sua vez são formadas por segmentos de linha. A principal característica são a conectividade e hierarquia. É fácil verificar quais segmentos de linha fazem parte de um objeto. Estas imagens podem ser resultado da captura por *scanners* especiais, onde o resultado não é uma imagem matricial, mas o contorno que foi "seguido" pelo sensor. Outro tipo de imagem vetorial é obtida na digitalização com o uso de mesas digitalizadoras, onde a imagem é composta pelo conjunto de coordenadas dos pontos obtidos pela movimentação do cursor sobre a mesa. São usadas principalmente em aplicações CAD e para geração de mapas. São facilmente apresentadas em um plotter pois é um dispositivo do tipo vetorial.

A conversão de uma imagem em formato vetorial para uma imagem em formato de varredura é feita através de um procedimento bem conhecido em computação gráfica (*scan-conversion algorithms*). Já a conversão inversa é mais difícil de ser feita.

2.5 Tipos de imagens digitais

Conforme foi visto, imagens digitais, que doravante chamaremos simplesmente de imagem, são formadas por um conjunto de *pixels*. O *pixel* é o menor elemento da imagem e possui um valor associado (nível). Esse valor está limitado pelo tipo de imagem. O tipo de imagem também indica como esses *pixels* serão mostrados no monitor. É importante salientar que a quantidade de memória requerida para armazenar, mostrar e editar as imagens cresce proporcionalmente ao número de níveis disponíveis.

2.5.1 Imagens Preto & Branco

São imagens onde os *pixels* só podem ter valor 0 (desligado ou preto) ou 1 (ligado ou branco). Este tipo de imagem é usada principalmente em processamento de textos pois neste caso, só estamos interessados nas letras (preto) e no fundo (branco). Pode-se criar a ilusão de vários tons por um arranjo da densidade de *pixels* pretos. Este efeito pode ser obtido pelas técnicas de "Dithering" ou "Halftoning". A FIGURA 2.1 apresenta um exemplo do efeito halftoning.

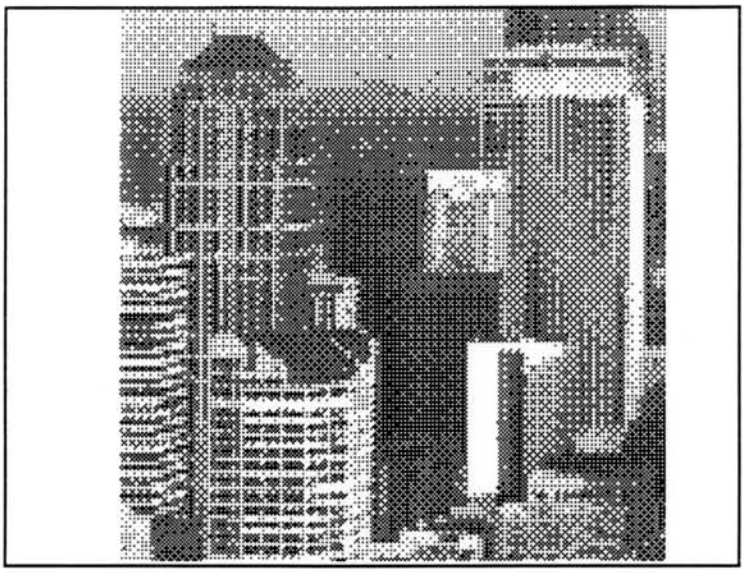

FIGURA 2.1-Imagem preto e branco com efeito halftoning.

2.5.2 Imagens monocromáticas em escala de cinza ("Grayscale")

São imagens onde os *pixels* podem ter valor entre 0 e N, representando a intensidade do cinza (o zero significará intensidade nula, ou seja, preto, e N significará máxima intensidade, ou seja, o branco). Todos os outros valores intermediários serão tons de cinza. Geralmente, N+1 é uma potência de dois, no caso de ser 256, necessitará de 8 bits para cada *pixel* e, conforme foi visto, será mais do que suficiente para representar todas as tonalidades que o olho humano é capaz de distinguir. Podemos ter também estas imagens para outras bandas espectrais (faixa de comprimento de onda), por exemplo, vermelho, azul, verde, etc. Neste caso, cada *pixel* indicará a intensidade do espectro em questão. É interessante ressaltar que estas imagens não precisam ficar restritas às freqüências de luz visível, podendo utilizar outras faixas de comprimento de onda, como por exemplo o infravermelho. Esta banda é muito utilizada em imagens de satélite, que possuem sensores especiais para esta faixa, podendo detectar por exemplo, a emissão de calor e consequentemente detectar queimadas. A FIGURA 2.2 apresenta um exemplo de imagens com 4, 16 e 256 níveis de cinza. As imagens que serão utilizadas para testes neste trabalho, são imagens monocromáticas em 256 níveis de cinza. Algumas delas são imagens capturadas do mundo real e outras serão imagens sintéticas, ou seja, criadas artificialmente para facilitar os testes.

FIGURA 2.2-Imagens com 4, 16 e 256 níveis de cinza.

2.5.3 Imagens multiespectrais

As imagens obtidas por sensoriamento remoto, através de satélites, são captadas por diferentes sensores, que respondem à reflectância dos alvos, e em alguns casos, à sua radiância, em 7 bandas de freqüências distintas, como no caso do LANDSAT-TM. Assim, uma imagem de satélite é chamada de multiespectral, pois representa um conjunto de várias imagens correspondentes a espectros distintos de freqüência, mas com certa correlação entre si. Uma imagem multiespectral pode ser visualizada parcialmente atribuindo-se a um conjunto de 3 das 7 bandas originais, os espectros R, G e B respectivamente. Neste caso, geramos as imagens em "pseudo-cores". Estas imagens são usualmente compostas por um conjunto de 24 *bits*: 8 bits para representar as intensidades de vermelho, 8 bits para o verde e 8 *bits* para o azul. Com a combinação dessas três cores básicas, utilizando-se 24 *bits/pixel*, pode-se chegar a um número de até 16 milhões de cores e tonalidades distintas. Este número é perfeitamente adequado para a representação da realidade sem perda de detalhes e qualidade em relação à cores, pois está acima da capacidade do olho humano em distinguir cores e tonalidades.

2.5.4 Imagens coloridas com uso de tabela auxiliar (Indexed 16-256 Color)

Existe outra forma de se obter imagens coloridas sem a necessidade de especificar a intensidade de cada espectro. Basta associar cada *pixel* a uma posição de uma tabela que contém as cores. Desta forma, economiza-se memória pois não é necessário usar 24 bits/*pixel*. Entretanto, a imagem terá um número máximo de cores distintas de acordo com o tamanho da tabela. A imagem é composta, então, pelo mapa de *pixels*, ou seja, a matriz M×N de pontos com os índices e de uma tabela de acesso indireto às cores reais, a chamada tabela de *palette*. Em geral, a tabela possui 16 ou 256 posições (cores).

2.6 Formato de armazenamento

As imagens digitais só são úteis quando estão armazenadas em uma forma que possam ser lidas por uma ou mais aplicações. É necessário, então, gravar essas imagens em uma forma padronizada para que elas possam ser manipuladas por um grande número de aplicações. Entretanto, existe uma série de especificações de formatos de arquivos utilizados atualmente. Cada fabricante de um equipamento ou software para aquisição ou manipulação de imagens pode criar um novo formato adequado à sua aplicação. Cada formato vai ter as suas características como por exemplo, capacidade de compactação dos arquivos, quantidade de cores, etc).

2.7 Classsificação de imagens

A classificação de imagens é a última etapa no processamento de imagens. Esse processamento pode ser classificado, quanto ao grau de abstração, em três níveis distintos: baixo, médio e alto. Ocorre uma redução progressiva da quantidade de informações manipuladas à medida que se passa por níveis crescentes de abstração. No

processamento de baixo nível, os dados de entrada são *pixels* da imagem original e os dados de saída representam propriedades da imagem, na forma de valores numéricos associados a cada *pixel*. No processamento de nível médio, este conjunto de valores produz como resultado uma lista de características. O processamento de alto nível produz, a partir destas características, uma interpretação do conteúdo da imagem. Uma estrutura funcional completa de um sistema de processamento e análise de imagens pode ser descrito da seguinte forma:

-Aquisição e digitalização: A imagem do sensor é transformada em uma imagem digital sobre a forma de uma tabela de valores discretos inteiros chamados *pixels*.

-Pré-processamento: Essa etapa permite corrigir um certo número de defeitos e imperfeições aparecidos durante a aquisição da imagem, que podem ter com causa características físicas do sistema, as condições deficientes de iluminação, etc. O pré-processamento não é indispensável, mas, na maioria dos casos, necessário.

-Segmentação: O objetivo é dividir uma imagem em partes constitutivas. Em uma imagem natural, a segmentação é efetuada pela detecção de descontinuidades (contornos) e/ou de similaridade (regiões) na imagem. A maioria dos processamentos é baseada na pesquisa dessas entidades que são armazenadas sobre uma forma adequada (segmentos ou primitivas).

-Interpretação ou classificação: É a parte mais "inteligente" do processo de visão por computador. Ela representa o "alto nível" e permite obter a compreensão e a descrição final do fenômeno inicial. Ela faz uso do conhecimento a priori do caso estudado e o conhecimento adquirido durante as fases precedentes. Dado um conjunto de classes e um padrão apresentado como entrada para o sistema, o problema consiste em decidir a que classe o padrão pertence. Deve haver a alternativa de rejeição do padrão.

A classificação pode ser supervisionada ou não-supervisionada. A classificação não-supervisionada consiste em *clusterizar* a imagem. Nessa técnica, a imagem é segmentada em um número indeterminado de classes. É do usuário a tarefa de rotular essas classes. Nesse trabalho será visto apenas a classificação supervisionada.

2.7.1 Classificação supervisionada

A classificação supervisionada é o procedimento mais utilizado para análise quantitativa de imagens, principalmente as de sensoriamento remoto. Baseia-se no uso de algoritmos para rotular os *pixels* de uma imagem pertencentes à determinada classe. Os passos essenciais consistem em [RIC86]:

1- Decidir quais as classes presentes na imagem original que devem ser segmentadas. Pode ser, por exemplo, água, região urbana, vegetação, etc.
2- Escolher *pixels* representativos de cada uma dessas classes. Esses *pixels* formarão um conjunto de treinamento para o algoritmo.
3- Usar os conjuntos de treinamento para estimar os parâmetros do algoritmo particular de classificação que será usado.
4- Usar o classificador treinado, para rotular cada pixel da imagem em uma das classes previamente treinada.

5- Produzir um mapa temático o qual apresenta o resultado da classificação.

Vários algoritmos podem ser usados nos passos 3 e 4. Entre eles o método de classificação pela Máxima-Verossimilhança é o mais comum e mais usado.

Classificação pela Máxima-Verossimilhança

Esse método será apresentado de uma maneira simplificada. A descrição formal e rigorosa pode ser encontrada no Apêndice E de [RIC86]:

Classificação de Bayes

Sejam as classes espectrais de uma imagem representadas por $w_i, i = 1 \text{K } M$ onde M é o número total de classes. Dado um vetor posicional x de componentes cujos valores são os brilhos de um pixel, em cada uma das bandas consideradas, podemos dizer que $p(w_i/x), i = 1 \text{K } M$ representa a probabilidade de x pertencer à classe w_i ou também, a verossimilhança de x em relação à w_i. A classificação é dada por:

$$x \in w_i \quad \text{se} \quad p(w_i/x) > p(w_j/x) \text{ para todo } j \neq i. \qquad (2.1)$$

Essa intuitiva regra de decisão é um caso especial de uma regra mais geral na qual a decisão pode ser influenciada por diferentes graus de significância atribuídos à diferentes classificações incorretas. A regra geral é chamada classificação de Bayes e também pode ser encontrada no apêndice E de [RIC86].

A regra de decisão da máxima verossimilhança

Apesar de sua simplicidade, os valores de $p(w_i/x)$ não são conhecidos. Entretanto, supondo que tenhamos um conjunto grande treinamento para cada classe, pode-se estimar a distribuição de probabilidade de encontrar um pixel pertencente a classe w_i na posição x. Essa probabilidade é representada por $p(x/w_i)$. O valor desejado de $p(w_i/x)$e o valor estimado de $p(x/w_i)$ através do conjunto de treinamento são relacionados pelo teorema de Bayes:

$$p(w_i/x) = p(x/w_i)p(w_i)/p(x),$$

onde $p(w_i)$é a probabilidade da classe $w_i$ocorrer na imagem, também chamada de probabilidade "a priori", pois é obtida pelo conhecimento prévio do especialista da imagem e $p(x)$é a probabilidade de encontrar um pixel de qualquer classe na localização x. Substituindo na expressão 2.1, temos:

$$x \in w_i \quad \text{se} \quad p(x/w_i)p(w_i) > p(x/w_j)p(w_j) \text{ para todo } j \neq i, \qquad (2.2)$$

onde $p(x)$ foi eliminado por ser um fator comum. Essa expressão é mais útil do que a expressão 2.1 pois $p(x/w_i)$ é conhecido e $p(w_i)$ pode ser estimado pelo especialista. Aplicando-se logaritmo natural, a equação 2.2 é rescrita como

$$x \in w_i \text{ se } g_i(x) > g_j(x) \text{ para todo } j \neq i, \qquad (2.3)$$

onde $g_i(x) = \ln\{p(x/w_i)p(w_i)\} = \ln p(x/w_i) + \ln p(w_i)$. A função $g_i(x)$ é conhecida como *função de discriminação.*

Modelo de classes de distribuição normal multivariada

Assumindo que as probabilidades de distribuição de cada classe são distribuições normais, para N bandas, temos que:

$$p(x/w_i) = (2\pi)^{-N/2} |\Sigma_i|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x-m_i)^t \Sigma_i^{-1}(x-m_i)\right\}, \qquad (2.4)$$

onde m_i e Σ_i são o vetor médio e a matriz de covariância dos dados na classe w_i.
Substituindo na equação 2.3 e eliminando fatores comuns como $-\ln(2\pi)$ e $-\frac{1}{2}$, temos a equação final da *função de discriminação*:

$$g_i(x) = -\ln|\Sigma_i| - (x-m_i)^t \Sigma_i^{-1}(x-m_i) \qquad (2.5)$$

Para implementar a máxima verossimilhança, usa-se a equação 2.5 em conjunto com a equação 2.3.

Classificação por Redes Neurais

Redes neurais podem ser vistas como intermediários entre métodos estatísticos e estruturais. A habilidade de aprendizado nas redes neurais fornece uma interessante alternativa aos classificadores Bayesianos. É especialmente interessante o fato de que não precisam ser feitas suposições acerca do modelo probabilístico. Diversos trabalhos já demonstraram que vários classificadores são casos particulares de redes neurais. Yau e Manary mostraram a equivalência entre classificadores Gaussianos e redes neurais Sigma-Pi [YAU90]. Eles mostraram que qualquer classificador Gaussiano pode ser implementado em uma rede neural. Um exemplo de classificação por redes neurais será visto em detalhes no sistema descrito no capítulo 4 desse trabalho. Outro trabalho nessa área pode ser encontrado em [BIS92].

2.8 Atributos da imagem

Os atributos de uma imagem podem ser classificados em dois tipos: atributos espectrais e atributos espaciais.

2.8.1 Atributos espectrais

Os atributos espectrais referem-se diretamente ao brilho de cada pixel. No caso de imagens de satélite, significa a quantidade de energia eletromagnética refletida ou emitida pelo terreno, em várias regiões do espectro. Por serem mais facilmente quantificáveis, os atributos espectrais têm sido tradicionalmente os mais utilizados no processo de classificação de imagens por computador. A não utilização dos demais atributos implica, obviamente, em não utilizar a totalidade da informação contida na imagem [COG94]. O principal descritor espectral é a média da distribuição dos níveis de cinza de uma classe (P), denotado por $\overline{m}(P)$. A expressão 2.6 apresenta a equação da média [CHR78]. Nessa expressão, P_{ij} representa o valor do nível de cinza de um *pixel* da classe P na posição (i, j) da imagem. Os pixels (P_{ij}) são selecionados a partir da identificação de regiões características da classe P presentes na imagem.

$$\overline{m}(P) = \frac{\sum_{i,j} P_{ij}}{|P|} \text{, onde } |P| \text{ é a quantidade de } pixels \text{ na classe } P. \qquad (2.6)$$

Entretanto, a representação de uma distribuição de *pixels* somente através de sua média não permite uma conclusão correta a respeito da dispersão dos seus atributos espectrais em uma determinada classe. Assim, para uma descrição mais precisa de suas características, convém associarmos à média uma medida de dispersão. Essa medida vai expressar com que grau as observações individuais diferem do valor médio representativo da classe P. As medidas mais utilizadas são a variância $s^2(P)$ e o desvio-padrão $s(P)$. A média é uma medida de tendência central, enquanto que a variância e o desvio-padrão são medidas de dispersão. As expressões 2.7 e 2.8 apresentam as equações da variância e do desvio-padrão [BAR94][CHR78].

$$s^2(P) = \sum_{z=0}^{L-1} (z - \overline{m}(P))^2 \frac{p[P,z]}{|P|} \qquad (2.7)$$

onde $p[P,z]$ é a quantidade de *pixels* em P que tem nível de cinza igual à z e L é a quantidade de níveis de cinza da imagem.

$$s(P) = \sqrt{s^2(P)} \qquad (2.8)$$

A variância como medida de dispersão apresenta a desvantagem de possuir dimensão diferente da dos dados observados, sendo esse problema eliminado pela utilização do desvio-padrão.

2.8.2 Atributos espaciais

Os atributos espaciais referem-se à disposição espacial ou à relação de vizinhança entre os *pixels*. Os atributos espaciais tem sido muito bem aproveitados por fotointérpretes, mas há sempre a subjetividade, fazendo com que uma imagem, ao ser interpretada por duas pessoas, apresente discrepâncias entre as duas classificações. Para

evitar a subjetividade humana e acelerar o processo de classificação espacial, alguns métodos computacionais têm sido desenvolvidos para a quantificação de atributos espaciais. A textura, que será vista no item 2.9, é um dos mais importantes atributos espaciais da imagem. A variância também pode ser considerada um atributo espacial se a calcularmos para uma janela e não para a imagem inteira. Nesse caso, a expressão 2.9 é outra forma de calcular a variância, orientada à atributos espaciais [CHR78].

$$s^2(P) = \frac{\sum_{i,j} (P_{ij} - \overline{m}(P))^2}{|P|-1} \qquad (2.9)$$

2.9 Textura

Embora a textura seja um termo popular facilmente compreensível pelo fato de estar presente em tudo que se vê, não existe uma definição universalmente aceita. Pesquisadores a tem caracterizado como sendo as variações tonais repetitivas e organizadas que podem ser distinguidas em uma pequena região de uma imagem.

De acordo com Haralick, a textura é caracterizada pelo número e tipo de suas primitivas e pela organização espacial ou "layout" das mesmas. Primitivas são regiões de contorno e tamanho semelhantes e que têm níveis de cinza dentro da mesma faixa de valores. A organização espacial pode ser aleatória, pode ter uma dependência entre pares de primitivas ou pode ter uma dependência entre *n* primitivas. Dentro dessa abordagem, Haralick propõe várias formas estatísticas para medir e caracterizar a textura de uma imagem, dentre as quais destacamos [HAR79]:

-funções de autocorrelação (autocorrelation functions).
-bordas texturais (textural edgeness).
-probabilidade de coocorrência espacial de níveis de cinza (spatial gray tone cooccurrence probabilities).
-comprimento da série de um nível de cinza (gray tone run lengths).

Funções de autocorrelação: É uma medida do tamanho das primitivas. Texturas finas tem primitivas pequenas e texturas grossas tem primitivas grandes.

Bordas de textura: A textura é vista como a quantidade de bordas por unidade de área. Texturas grossas tem pequeno número de bordas por unidade de área, enquanto que texturas finas tem um grande número de bordas por unidade de área.

Probabilidade de coocorrência espacial de níveis de cinza: É conhecida como matriz de coocorrência dos níveis de cinza. Caracteriza a textura pela coocorrência de seus níveis de cinza. Texturas grossas são aquelas em que os tons mudam lentamente com a distância enquanto que as texturas finas são aquelas em que os tons mudam rapidamente com a distância. Uma das vantagens desta abordagem é ser invariante sob transformações na imagem, como rotação. Sua desvantagem, é não ser capaz de capturar os aspectos das figuras das primitivas tonais, não revelando a forma da textura. Esta abordagem, pelo fato de ser a escolhida para a implementação do sistema IMASEG, será visto com maiores detalhes no item 2.10.

Comprimento da série de um nível de cinza: Uma série de um nível de cinza são *pixels* linearmente adjacentes com o mesmo nível de cinza. As texturas grossas terão um grande número de séries de comprimento longo enquanto que texturas finas terão grande número de séries de comprimento pequeno. As séries são usualmente medidas ao longo das direções horizontal, vertical e diagonais.

Outra técnica bastante conhecida para quantificar a textura é através da morfologia matemática. Esse processo foi proposto por Matheron [MAT67]. A idéia é a de um elemento estruturante, uma forma geométrica qualquer, que se desloca sobre a imagem binária gerando uma nova imagem binária. Propriedades texturais podem ser obtidas através da parametrização do elemento estruturante e o número de elementos encontrados na nova imagem, em função desse parâmetro [CLA95].

Marana et all fazem uma análise de textura utilizando Transformada de Hough e Morfologia Matemática. A partir de linha extraídas pela transformada de Hough são obtidos dados estatísticos. Regiões com estatísticas diferentes são segmentadas usando a operação morfológica de fechamento. Essa operação define as regiões da imagem que possuem segmentos de retas com orientações similares, ou seja, definem as regiões cujas texturas incluem orientações similares [MAR95]. O método da transformada de Hough é aplicável quando se possui informações precisas acerca da forma da curva. Os dados de base da transformada de Hough são geralmente pontos de uma imagem obtida através das transformações de gradiente e limiarização. A idéia é aplicar na imagem uma transformação tal que todos os pontos pertencentes a uma mesma curva sejam mapeados num único ponto de um espaço dos parâmetros da curva procurada. A transformada de Hough é um método de acumulação de requisitos muito geral. Ela permite detectar praticamente qualquer curva, mesmo aquelas pouco visíveis e fortemente ruidosas [HOU62].

2.10 Matrizes de Coocorrência

No item anterior, descreveu-se várias formas para quantificar a textura. O método das matrizes de coocorrência será adotado neste trabalho na implementação do sistema descrito no capítulo 4 e, por isso, será explicado a seguir com mais detalhes.

Este método foi utilizado pela primeira vez por Julesz em experimentos de discriminação de textura [JUL62]. Outros autores também utilizaram essa abordagem em aplicações diversas como identificação de tipos de nuvens baseado na sua textura em imagens de satélite, aplicações médicas, automatização na análise de raios-X do tórax e discriminação de células cervicais [DAR68][BAR69][CHI74][PRE76]. Todos estes estudos obtiveram resultados razoáveis usando coocorrência dos níveis de cinza [HAR79].

2.10.1 Vizinhança entre *pixels* numa imagem

Como já foi mencionado, um dos aspectos da textura está relacionado com a distribuição e dependências espaciais nos níveis de cinza entre *pixels* vizinhos. Suponha uma imagem retangular que tenha N_c células de resolução na direção horizontal e N_r células de resolução na direção vertical e que o nível de cinza em cada célula esteja

quantizado em N_g níveis. Seja $L_c = \{1,2,\mathrm{K}\ ,N_c\}$o domínio espacial na horizontal, $L_r = \{1,2,\mathrm{K}\ ,N_r\}$ o domínio espacial na vertical e $G = \{1,2,\mathrm{K}\ ,N_g\}$o conjunto de N tons de cinza quantificados. O conjunto $L_r \times L_c$ é o conjunto de células da imagem ordenados pelas suas designações linha-coluna. A imagem I pode ser representada por uma função que atribui um tom de cinza de G para cada célula ou par de coordenadas em $L_r \times L_c$; I: $L_r \times L_c \rightarrow G$.

As relações de vizinhança entre dois *pixels* podem ser quanto à orientação e quanto à distância entre eles. Em relação à orientação, eles podem ser vizinhos na horizontal (*H*), na vertical (*V*), na diagonal direita (*DD*) ou na diagonal esquerda (*DE*). Em relação à distância, eles podem estar afastados de 1, 2, 3 ou mais posições. Por exemplo, na imagem da FIGURA 2.3, os *pixels* (1,2) e (1,4) são vizinhos na horizontal, separados pela distância 2. Os *pixels* (2,2) e (3,3) são vizinhos na diagonal esquerda, separados por 1.

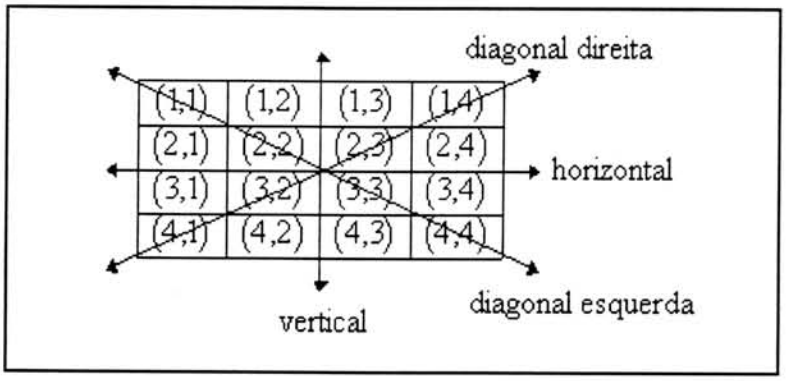

FIGURA 2.3-Posição dos *pixels* numa imagem 4×4.

Os conjuntos R^H, R^V, R^{DD} e R^{DE} da FIGURA 2.4, descrevem formalmente todos os pares de *pixels* vizinhos nas direções *H*, *V*, *DD* e *DE* separados pela distância de uma posição numa imagem 4×4.

$$R^H = \{(k,l),(m,n) \in (L_r \times L_c) \times (L_r \times L_c) \mid\ k-m=0, |l-n|=1\}$$

$$= \{((1,1),(1,2)),((1,2),(1,1)),((1,2),(1,3)),((1,3),(1,2)),((1,3),(1,4)),((1,4),(1,3)),$$
$$((2,1),(2,2)),((2,2),(2,1)),((2,2),(2,3)),((2,3),(2,2)),((2,3),(2,4)),((2,4),(2,3)),$$
$$((3,1),(3,2)),((3,2),(3,1)),((3,2),(3,3)),((3,3),(3,2)),((3,3),(3,4)),((3,4),(3,3)),$$
$$((4,1),(4,2)),((4,2),(4,1)),((4,2),(4,3)),((4,3),(4,2)),((4,3),(4,4)),((4,4),(4,3))\}$$

$$R^V = \{(k,l),(m,n) \in (L_r \times L_c) \times (L_r \times L_c) \mid\ |k-m|=1, l-n=0\}$$

$$R^{DD} = \left\{ (k,l),(m,n) \in (L_r \times L_c) \times (L_r \times L_c) \;\middle|\; \begin{matrix} (k-m=1, l-n=-1) \text{ ou} \\ (k-m=-1, l-n=1) \end{matrix} \right\}$$

$$R^{DE} = \left\{ (k,l),(m,n) \in (L_r \times L_c) \times (L_r \times L_c) \;\middle|\; \begin{matrix} (k-m=1, l-n=1) \text{ ou} \\ (k-m=-1, l-n=-1) \end{matrix} \right\}$$

FIGURA 2.4-Cjto. de todas vizinhanças horizontais com $d = 1$ numa imagem 4×4.

2.10.2 Cálculo da quantidade de pares de *pixels* vizinhos numa imagem

O conjunto de pares de *pixels* vizinhos irá depender do tamanho da imagem $(N_c \times N_r)$, da distância d entre eles e da orientação envolvida (H, V, DD, DE). Conforme visto no item 2.10.1, estes conjuntos são nomeados R , R , R^{DD} e R^{DE}. A quantidade de pares de *pixels* vizinhos, ou seja, o número de elementos de cada conjunto é representada por $\#R^H$, $\#R^V$, $\#R^{DD}$ e $\#R^{DE}$ e é importante pois serve como base para o cálculo dos valores mínimo e máximo dos possíveis parâmetros de entrada da R.N. (uniformidade, entropia e contraste).

No exemplo da FIGURA 2.3, basta contar o número de elementos do conjunto R da FIGURA 2.4. A FIGURA 2.5 apresenta as equações genéricas para cálculo desses valores pressupondo-se a distância d =1 .

$\#R = 2\,N_c\,(N_r - 1)$ para orientação horizontal.
$\#R = 2\,N_r\,(N_c - 1)$ para orientação vertical.
$\#R^{DD} = 2\,(N_r - 1)\,(N_c - 1)$ para diagonal direita.
$\#R^{D} = 2\,(N_c - 1)\,(N_r - 1)$ para diagonal esquerda.
$\#R^S = \#R + \#R^V + \#R^{DD} + \#R^D$,
onde $\#R^S$ é quantidade de pares de *pixels* vizinhos em todas orientações.

FIGURA 2.5-Cálculo da quantidade de pares de *pixels* vizinhos numa imagem.

2.10.3 Construção da matriz de coocorrência

Dada uma orientação de vizinhança entre dois *pixels* de uma janela (*H*-horizontal, *V*-vertical, *DD*-diagonal direita e *DE*-diagonal esquerda) e uma distância d a separá-los, a matriz de coocorrência de uma janela da classe P é uma matriz C_P onde cada elemento C_{ij} representa a freqüência com que dois *pixels* vizinhos ocorrem na janela, um com nível de cinza i e o outro com nível de cinza j, de acordo com a orientação de vizinhança adotada e separados pela distância d. Existirá uma matriz de coocorrência para cada orientação de vizinhança e para cada distância d escolhida. Se fixarmos d=1, teremos quatro matrizes de coocorrência: C_P^H, C_P^V, C_P^{DD}, C_P^{DE}. Por exemplo, no caso da matriz C_P^H, ela representará a probalidade de uma transição horizontal do nível de cinza i para j numa janela da textura P. Formalmente, os elementos C_{ij} de cada uma delas são definidos da seguinte forma:

$$C_P^H(i,j,d) = \#\left\{(k,l),(m,n) \in (L_r \times L_c) \times (L_r \times L_c) \;\middle|\; \begin{matrix} k-m=0, |l-n|=d, \\ I(k,l)=i, I(m,n)=j \end{matrix} \right\}$$

$$C_P^V(i,j,d) = \#\left\{(k,l),(m,n) \in (L_r \times L_c) \times (L_r \times L_c) \;\middle|\; \begin{matrix} |k-m|=d, l-n=0, \\ I(k,l)=i, I(m,n)=j \end{matrix} \right\}$$

$$C_P^{DD}(i,j,d) = \#\left\{ (k,l),(m,n) \in (L_r \times L_c) \times (L_r \times L_c) \;\middle|\; \begin{array}{l} (k-m=d, l-n=-d) \text{ ou} \\ (k-m=-d, l-n=d), \\ I(k,l)=i, I(m,n)=j \end{array} \right\}$$

$$C_P^{DE}(i,j,d) = \#\left\{ (k,l),(m,n) \in (L_r \times L_c) \times (L_r \times L_c) \;\middle|\; \begin{array}{l} (k-m=d, l-n=d) \text{ ou} \\ (k-m=-d, l-n=-d), \\ I(k,l)=i, I(m,n)=j \end{array} \right\}$$

onde # denota o número de elementos do conjunto. A matriz terá dimensão $N_g \times N_g$. Nessa expressões, $I(x,y)$ representa o valor do nível de cinza de um pixel na posição (x,y) de uma janela da classe P. As janelas da classe P são selecionadas a partir da identificação de regiões características dessa classe, presentes na imagem.

A forma geral da matriz de coocorrência está representada na FIGURA 2.6, onde cada elemento $\#(i,j)$ é a quantidade de pares de *pixels* vizinhos que ocorrem na imagem, um com nível de cinza i e outro com nível de cinza j, de acordo com uma distancia d e uma orientação angular de vizinhança. Por exemplo, se d =1 e a orientação for H, então o elemento na posição (2,1) é o número de vezes que encontramos dois *pixels* adjacentes horizontalmente com níveis de cinza 2 e 1 respectivamente. Para determinar este número, contamos o número de pares de células em R^H de tal forma que a primeira célula do par tenha nível de cinza igual à 2 e a segunda célula do par tenha nível de cinza igual à 1.

	0	1	2	3	...	N_g
0	#(0,0)	#(0,1)	#(0,2)	#(0,3)	...	$\#(0,N_g)$
1	#(1,0)	#(1,1)	#(1,2)	#(1,3)	...	$\#(1,N_g)$
2	#(2,0)	#(2,1)	#(2,2)	#(2,3)	...	$\#(2,N_g)$
3	#(3,0)	#(3,1)	#(3,2)	#(3,3)	...	$\#(3,N_g)$
...	...	...	...	...	...	$\#(4,N_g)$
N_g	$\#(N_g,0)$	$\#(N_g,1)$	$\#(N_g,2)$	$\#(N_g,3)$	$\#(N_g,4)$	$\#(N_g,N_g)$

FIGURA 2.6-Forma da matriz de coocorrência.

Na FIGURA 2.7(a) temos uma imagem de dimensão 4×4 com 4 níveis de cinza, variando de 0 à 3. Nas FIGURA 2.7(b) a (e), foram calculadas as quatro matrizes de coocorrência para d=1. Note que estas matrizes de coocorrência sempre serão simétricas: $C_{ij} = C_{ji}$. Além disso, a dimensão dessas matrizes depende unicamente do número de níveis de cinza existentes na imagem, não do tamanho da imagem.

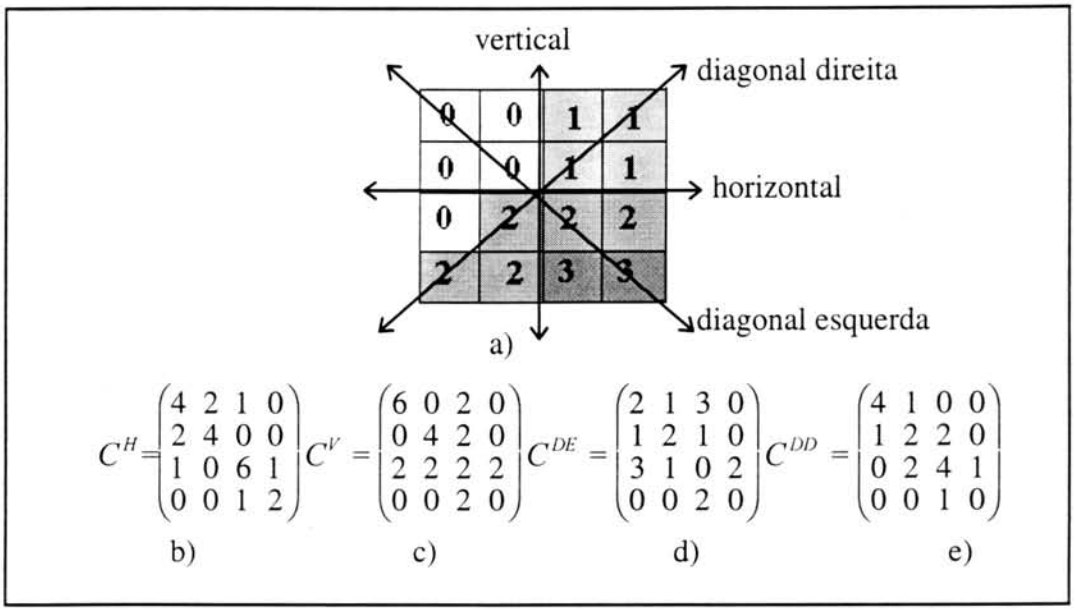

FIGURA 2.7-Imagem e as respectivas matrizes de coocorrência.

2.10.4 Normalização da matriz de coocorrência

Para obtermos a matriz de coocorrência normalizada C^N, deve-se dividir cada elemento da matriz C_{ij} pelo número de vizinhanças da imagem ($\#R$) usada para geração da matriz. Desta forma, temos: $C_{ij}^N = \frac{C_{ij}}{\#R}$. O somatório dos elementos da matriz de coocorrência normalizada será sempre igual à 1.

2.10.5 Feições extraídas da matriz de coocorrência

A matriz de coocorrência por si só não nos dá uma quantificação da textura. Para isso, podem ser extraídas, a partir dessa matriz, várias feições que irão quantificá-la, dependendo do aspecto que nos interessa. Algumas dessas feições estão listadas nas expressões 2.10, 2.11 e 2.12. As demais podem ser encontradas em [HAR73].

$$\text{Uniformidade: } h = \sum_{i,j} C_{ij}^2 \qquad (2.10)$$

$$\text{Entropia: } e = -\sum_{i,j} C_{ij} \times \log C_{ij} \qquad (2.11)$$

$$\text{Contraste: } k = \sum_{i,j} (i-j)^2 \times C_{ij} \qquad (2.12)$$

-Uniformidade ou homogeneidade: Uma imagem uniforme terá poucas transições dos níveis de cinza. Portanto, a matriz de coocorrência associada terá a maioria dos elementos com valores baixos e poucos elementos com valores altos. Isso faz com que a soma dos quadrados dos elementos seja maior numa imagem uniforme e menor numa imagem heterogênea a qual terá a maioria dos elementos da matriz com valores médios.

-Entropia: É uma medida da aleatoriedade ou desorganização presente em uma imagem. Deve-se observar que C_{ij} pode ter valor 0, portanto o logaritmo não estará definido. Neste trabalho, substitui-se $0 \times \log 0$ por 0, visto que $\lim_{Cij \to 0} C_{ij} \times \log C_{ij} = 0$, conforme ilustra FIGURA 2.8. Como C_{ij} da matriz normalizada varia entre $[0;1]$, o valor de $C_{ij} \times \log C_{ij}$ será sempre negativo. Por isso, o sinal negativo na equação da entropia para tornar o valor positivo.

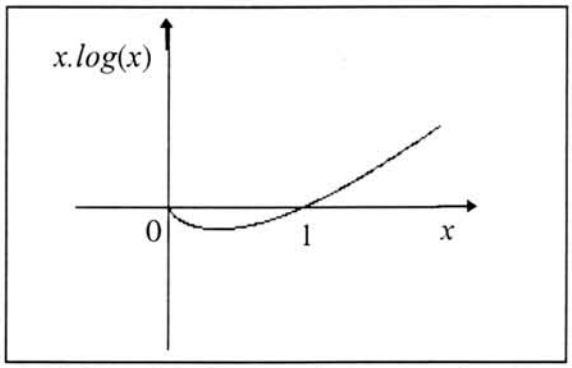

FIGURA 2.8-Gráfico da função $x.\log(x)$.

-Contraste: O contraste será maior quanto mais os elementos da matriz de coocorrência estiverem afastados da diagonal principal. Isto faz com que imagens com alto contraste tenham *pixels* vizinhos com grande diferença no nível de cinza e vice-versa.

Usando as feições das expressões 2.10, 2.11 e 2.12, Haralick realizou vários experimentos para identificação de classes. Num conjunto de imagens aéreas com oito classes de terrenos (residencial velho, residencial novo, lago, pântano, charco, pátio de manobras de trens, cerrado e bosque) foi obtido um percentual de 82% de acerto na identificação. Numa imagem de satélite LANDSAT da baía de Monterrey na Califórnia, foi obtido um percentual de 84% de acerto na classificação, usando sub-imagens de 64×64 *pixels*. Neste experimento, foram usadas feições espectrais e texturais para sete classes de terrenos: floresta costeira, bosque, pasto, áreas urbanas, campos irrigados grandes, campos irrigados pequenos e água. A grande variedade de imagens usadas nestes experimentos e o bom índice de desempenho, é um indicativo do poder e generalidade das matrizes de coocorrência.

Além dessas características, podemos extrair ainda outras importantes informações [PAV82]. Algumas propriedades da matriz de coocorrência são decorrentes da própria maneira como ela é construída. Primeiramente, se fizermos a média das 4 matrizes e analisarmos imagens de dimensões grandes, os elementos da diagonal principal da matriz serão aproximadamente iguais à área da imagem na sua respectiva posição. Em outras palavras, C_{kk} representará aproximadamente o tamanho da área de *pixels* com valor *k*. Segundo, os elementos da diagonal secundária tem valores aproximadamente iguais ao comprimento da borda entre regiões, ou seja, C_{kj} representa aproximadamente o comprimento da borda entre regiões com *pixels* iguais à *k* e *j* respectivamente. Para imagens com baixo contraste, os elementos distantes da diagonal

principal devem ser zero ou ter valores muitos pequenos, enquanto que o oposto é válido para imagens com alto contraste.

A maior dificuldade de usar matrizes de coocorrência reside na sua grande dimensão $\left(N_g \times N_g\right)$. Na implementação do sistema IMASEG será explicado como foi feita uma redução desta dimensão para acelerar o processamento e para que seja possível carregar a matriz na memória. Por fim, a matriz de coocorrência pode ajudar na tentativa de formalização da definição de textura, no sentido de que, se duas áreas de uma imagem tiverem a mesma matriz de coocorrência, então elas terão a aparência de ter a mesma textura.

2.11 Sistemas computacionais para processamento de imagens

Existem diversos sistemas comerciais e científicos disponíveis, desenvolvidos em universidades ou empresas. Podemos destacar, entre eles:

-SPRING

O sistema SPRING (Sistema de Processamento de Informações Georeferenciadas), desenvolvido no INPE com suporte da EMBRAPA e IBM, funciona em estações de trabalho UNIX, sob o "X window system". Esse sistema fornece ao usuário um ambiente interativo para visualizar, manipular e editar imagens e dados cartográficos, estando integrado a um ambiente de banco de dados para arquivar e recuperar dados espaciais e seus atributos e dispondo de uma biblioteca de classes em C++ para compor um sistema extensível para desenvolvimento de novas aplicações em Processamento de Imagens e GIS (sistemas de informação geográfica). O SPRING está disponível, em código-fonte e sem custos, para fins de pesquisa e desenvolvimento [CAM92].

-IRENE

O IRENE é um sistema para classificação de imagens multiespectrais. A classificação é feita em duas etapas. Primeiro, é realizado uma *clusterização* dos alvos baseada em Redes Neurais Auto-organizadas (Self-Organizing Maps - SOM). Após, é usado um algoritmo conhecido como Quantização Vetorial Adaptativa (LVQ-Learning Vector Quantization) para a classificação da imagem. A primeira etapa é realizada de forma autônoma, enquanto que a segunda demanda a supervisão do usuário. O sistema oferece uma solução bastante interessante aos diversos problemas envolvendo o processamento de imagens [ENG94].

-KHOROS

O Khoros é um software com ambiente integrado para visualização de dados, programação e simulação visual e desenvolvimento de software. Esse software, que roda em ambiente Unix, é usado por cientistas e engenheiros para resolver problemas em processamento de imagens, tratamento de imagens médicas, controle de processos, processamento de sinal e análise numérica.

-NICE (Neural Image Classification Environment):

É um ambiente para geração de mapas temáticos desenvolvido pela IBM. Faz a segmentação e a classificação de imagens de sensoriamento remoto, combinando técnicas de redes neurais (algoritmo Backpropagation) com lógica nebulosa, permitindo lidar com problemas tais como transições graduais entre classes e a ocorrência de interferências, como nuvens e sombras. O NICE foi utilizado em conjunto com o INPE - Instituto Nacional de Pesquisas Espaciais - no desenvolvimento de um classificador neural para avaliação do desflorestamento na Amazônia.

3. Redes Neurais

Tradicionalmente, a inteligência artificial (I.A.) divide-se em duas grandes áreas: a I.A. tradicional e a Inteligência Computacional. Enquanto a primeira utiliza conceitos simbólicos de inteligência, a segunda investiga sistemas computacionais que apresentam comportamento inteligente a partir de técnicas sub-simbólicas, baseadas em modelos matemáticos e técnicas de cálculo numérico. As principais vantagens destas técnicas estão ligadas ao ferramental matemático disponível para a simulação do comportamento destes sistemas, bem como sua implementação em hardware. A área de inteligência computacional utiliza principalmente três técnicas: Redes Neurais, Lógica Fuzzy e Algoritmos Evolutivos. Entre as principais aplicações dos sistemas computacionais inteligentes podemos destacar o processamento inteligente de imagens digitais, o controle de processos industriais e o controle de robôs autônomos.

Neste capítulo, será apresentada uma breve visão sobre redes neurais artificiais, ou simplesmente redes neurais (R.N.), a fim de subsidiar o entendimento dos próximos capítulos. Dentre os vários modelos de redes neurais, será dada uma ênfase maior à rede B.P.N.(Backpropagation network) pois esta foi escolhida para ser utilizada na implementação do sistema.

3.1 Introdução

O estudo de redes neurais surgiu a partir do objetivo de criar sistemas inteligentes imitando-se a inteligência humana. Partindo-se do fato de que o sistema nervoso é o responsável pelas funções da inteligência (raciocínio, sensações, tarefas automatizadas, integração de idéias, adaptação, etc), partiu-se para a tentativa de imitação do seu funcionamento.

Dada a complexidade do sistema nervoso humano, foi necessário simplificar esta estrutura e extrair o funcionamento e os elementos básicos do mesmo. O elemento básico do cérebro humano é o neurônio (FIGURA 3.1). Os neurônios são células que recebem, processam e transmitem sinais químicos e elétricos. Estão presentes em nosso cérebro numa quantidade da ordem de 10^{11} com cerca de 10^{15} conexões entre eles.

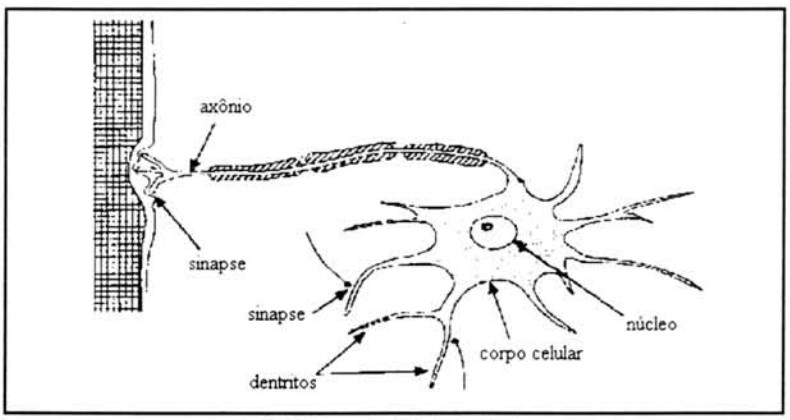

FIGURA 3.1-Neurônio.

Os sinais de entrada (estímulos) chegam ao neurônio através dos dentritos, e o sinal de saída é enviado através do axônio. Cada neurônio é uma unidade independente de processamento de informações que está conectada a diversos outros neurônios através da ligação axônio-dentrito, formando um complexo circuito cerebral. Essas conexões são chamadas de sinapses. Os contatos sinápticos determinam a forma com que os sinais passam entre os neurônios, através de um processo eletro-químico com a ajuda dos neurotransmissores. Outra característica importante é o processamento altamente paralelo do cérebro, que também foi repetido na arquitetura das redes neurais.

3.2 Modelo de neurônio

A partir dessas simplificadas constatações biológicas, foi proposto o conceito de elemento processador (PE - "Processing Element") da rede neural (FIGURA 3.2). O elemento processador possui diversas entradas (x_i) e um valor associado a cada entrada chamado peso sináptico (w_{ji}). O elemento processador tem a função de fazer a integração dos sinais de entrada através de uma regra de propagação, gerando um valor (*net*). A saída passa por uma função de ativação ou "threshold" *f*(*net*) gerando s_j que é a saída do elemento processador. Este sinal será propagado de acordo com a topologia de interconexões da rede de neurônios. Uma característica importante deste modelo é a possibilidade de os neurônios atuarem em paralelo.

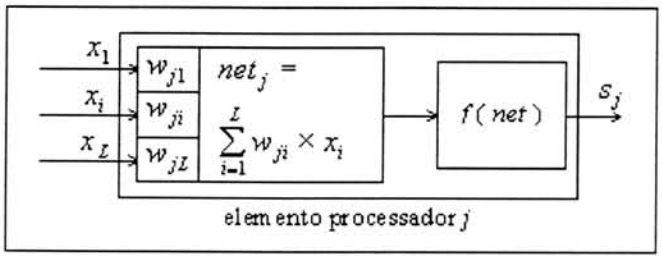

FIGURA 3.2-Elemento processador da rede neural.

Apesar das simplificações, as redes neurais exibem várias características similares ao cérebro humano, entre as quais podemos citar: o aprendizado construído pela experiência, a capacidade de generalizar exemplos anteriores para novos gerando a habilidade de lidar com ruído e distorção na entrada e a robustez, fazendo com que a perda de um elemento processador não cause o mal funcionamento do sistema.

3.2.1 Função de ativação

Existem alguns tipos de função de ativação *f*(*net*), entre os quais a função degrau, a função semi-linear e a função sigmóide. O comportamento dessas funções está descrito na FIGURA 3.3.

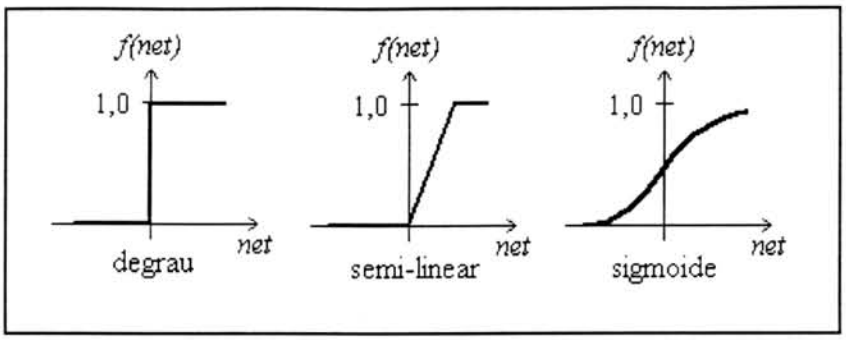

FIGURA 3.3-Tipos de função de ativação.

3.3 Histórico

A área de redes neurais é relativamente nova. Em 1943, McCulloch e Pitts [MCC43] criaram o primeiro modelo computacional, o Psychon. O Psychon é um dispositivo lógico de dois estados que gera um sinal binário de saída quando as entradas somadas ultrapassam um valor limite de excitação. Entretanto, este modelo não previa a capacidade de adaptação.

Em 1949, D.O.Hebb publicou a sua importante obra "The organization of Behavior" [HEB49]. O trabalho de Hebb não resultou diretamente em um modelo específico e bem formalizado de uma rede neural, mas sim em uma análise das características e formas de comportamento que uma rede neural deveria possuir. Hebb propôs que a capacidade de aprendizado de uma rede neural reside na auto-organização das suas ligações sinápticas, ou seja, o aprendizado ou memória reside na distribuição de intensidades das ligações entre os neurônios. Afirmava Hebb que pares de neurônios que são ativados simultaneamente tornam-se mais fortes através de alterações em suas sinapses. Ele estabeleceu que as sinapses mais freqüentemente ativadas devem ter maior chance de se tornarem ativas novamente.

Em 1959, Frank Rosenblatt criou o Perceptron que tem até hoje uma grande influência nos estudos sobre redes neurais [ROS59]. Nesta época, também foram desenvolvidos outros modelos similares ao Perceptron como é o caso do Adaline, criado por Bernard Widrow em 1962 [WID62]. Estes modelos são baseados na correção de erros e formam uma importante classe de redes neurais.

Entre 1969 e o começo da década de 80, as redes neurais perderam entusiasmo. A publicação feita por Minsky e Paper provou matematicamente que os modelos de redes neurais usados até então não eram capazes de aprender uma simples função lógica "XOR" (ou exclusivo) [MIN69].

O ressurgimento do interesse pela área veio em 1982 com o modelo de Hopfield o qual se utilizava dos conceitos de aprendizado definidos por Hebb [HOP82]. Nessa mesma década, onde surgiram computadores mais velozes e poderosos, também causou o aparecimento de uma grande quantidade de modelos importantes de redes neurais. O modelo multinível, que se utiliza da regra de aprendizado BackPropagation, foi primeiramente formalizado por Werbos [WER74], e posteriormente apresentado por Parker [PAR85] e Rumelhart e McClelland [CLE86]. Essa rede podia resolver o problema do "XOR". Dentre os diversos modelos que surgiram nessa época, estão dois muito importantes: os chamados mapas auto-organizáveis (SOM) criado por Teuvo Kohonen [KOH82] e o modelo ART criado por G. Carpenter e S. Grossberg [CAR83].

3.4 Classificação dos modelos de redes neurais

As redes neurais podem ser classificadas de acordo com as seguintes características:

-Tipo de treinamento e regra de aprendizado: O treinamento pode ser supervisionado (supervised learning) ou não-supervisionado (unsupervised learning) [FRE91]. O treinamento supervisionado consiste em apresentar à rede um padrão a ser reconhecido juntamente com a resposta que a rede deve fornecer ao reconhecer novamente este mesmo padrão. Geralmente, neste tipo de treinamento temos uma regra de aprendizado do tipo correção de erros. Esta regra está baseada no princípio de adaptação e correção dos pesos de atuação de cada neurônio, até que este responda da maneira desejada. Um exemplo é a regra Delta Genealizado explicada no item 3.6. O treinamento não-supervisionado ou auto-aprendizado consiste apenas em apresentar os padrões que se quer reconhecer à rede e esta deverá ser capaz de agrupar os padrões que possuem propriedades similares. Este processo também é chamado de clusterização. Geralmente, neste tipo de treinamento, temos uma regra de aprendizado do tipo competitivo a qual se caracteriza pelas conexões laterais dos neurônios com os seus vizinhos FIGURA 3.4c), estabelecendo assim uma "competição entre os neurônios" que levará a rede a um estado estável.

-Topologia da rede: As redes podem ter um único nível ou mais de um nível (multilayer). As FIGURA 3.4 a, b, c) ilustram redes de um único nível e a FIGURA 3.4d) ilustra uma rede multinível.

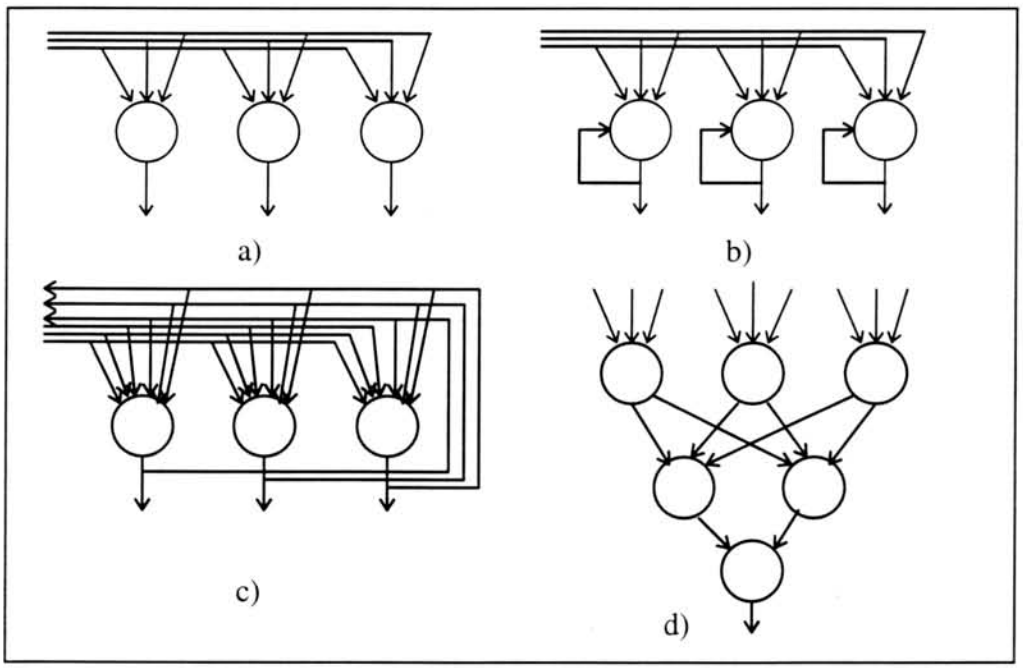

FIGURA 3.4-Topologia de redes.

-Interconexão dos neurônios: Podem ser sem ou com realimentação (*feed-forward ou feedback*). As primeiras possuem neurônios cujas saídas conectam-se apenas com os níveis superiores da rede, não havendo realimentação de suas entradas. A saída de um

neurônio não influenciará a sua própria resposta. No segundo tipo, um neurônio pode se conectar a uma de suas próprias entradas ou à entrada de um neurônio em um nível inferior da rede. Desta forma, a rede do tio *feedback* possui neurônios cuja saída é capaz de influenciar de uma maneira direta ou indireta o seu próprio comportamento. Um exemplo típico desta rede é o modelo de Hopfield.

3.5 Aplicações

As redes neurais possuem um espectro muito grande de aplicações, dentre as quais podemos destacar trabalhos no reconhecimento de padrões (imagens, voz, etc), na área financeira, no controle de processos, na previsão do tempo, na área biomédica, na otimização de problemas combinatórios, etc.

3.6 Rede Multicamada de perceptons com algoritmo de treinamento Backpropagation

A rede neural MLP (Multi-Layer Percepton) merece ser vista com um nível de detalhamento maior visto que é o modelo que será usando na implementação do sistema IMASEG. Contudo, para uma completa referência do modelo, sugerimos [WER74], [PAR85] e [CLE86].

Este modelo é um dos mais estudados e adotados pela sua grande capacidade de adaptação a qualquer tipo de padrão. A rede MLP, juntamente com o modelo de Hopfield, foram os responsáveis pelo ressurgimento do interesse nas redes neurais na década de 80.

Esse modelo, também conhecido como BPN (backpropagation network), é uma rede *feedfoward*, com treinamento supervisionado, formada por três ou mais níveis (*multilayer*): um nível de entrada (*input layer*), um ou mais níveis intermediários ou ocultos (*hidden layers*) e um nível de saída (*output layer*), conforme FIGURA 3.5. Na grande maioria dos casos práticos, a camada oculta tem um só nível. O número de neurônios do nível de entrada e saída pode ser estimado de acordo com as características do problema. Entretanto, não existe uma técnica precisa para estimar o número de neurônios da camada oculta, devendo ser estabelecido de uma maneira intuitiva.

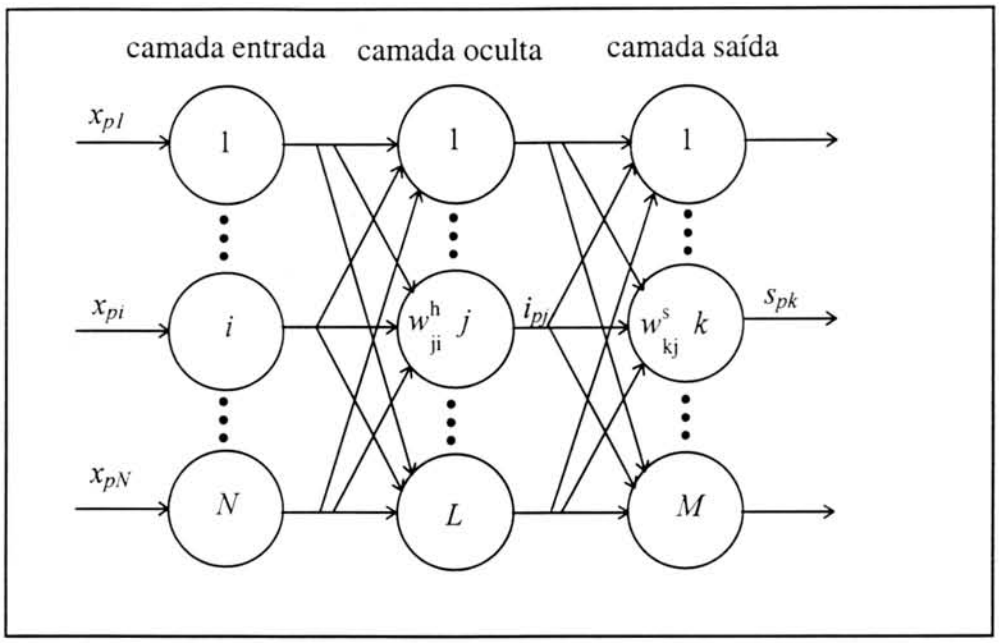

FIGURA 3.5-Backpropagation network.

A rede BPN possui treinamento supervisionado baseado em duas fases: propagação e adaptação.

Procedimento para treinamento (Regra Delta Generalizada):
1- Aplicar um vetor de entrada, propagar e calcular os valores de saída (propagação).
2- Determinar o erro, em relação aos valores desejados na saída.
3- Determinar as correções nos pesos (descida mais íngreme do gradiente do erro).
4- Aplicar as correções (adaptação).
5- Repetir os passos de 1 à 4 até que o erro médio quadrado E seja suficientemente pequeno ou atingir um número determinado de iterações.

O erro médio quadrado pode ser calculado por $E = \frac{1}{M}\sum_{p=1}^{M}\left(y_{pk} - s_{pk}\right)^2$ onde y_{pk} é a saída desejada.

Propagação: A função de ativação na propagação é uma função do tipo sigmóide. Neste trabalho, utilizou-se a função tangente hiperbólica (tanh(x)).

A saída da camada de saída é dada por [FRE91]:

$s_{pk} = f_k^s\left(\text{net}^s_{pk}\right)$, onde $\text{net}^s_{pk} = \sum_{j=1}^{L} w^s_{kj} \,.\, i_{pj}$ e f é a função de ativação.

A saída da camada oculta é dada por [FRE91]:

$i_{pj} = f_j^h\left(\text{net}^h_{pj}\right)$, onde $\text{net}^h_{pj} = \sum_{i=1}^{N} w^h_{ji} \,.\, x_{pi}$.

Adaptação: É baseado na retro-propagação do erro para os níveis anteriores da rede, de acordo com o grau de participação que cada neurônio teve no erro do nível posterior, derivando o nome de Backpropagation.

A atualização de um peso genérico da camada de saída é expressa por [FRE91]:
$w_{kj}^{s} = w_{kj}^{s} + \mu . \delta_{pk}^{s} . i_{pj}$, onde $\delta_{p}^{s} = \left(y_{p} - s_{p}\right).\left(1 - s_{p}^{2}\right)$.

A atualização de um peso genérico da camada escondida é expressa por [FRE91]:
$w_{ji}^{h} = w_{ji}^{h} + \mu .\ \delta_{pj}^{h} . x_{pi}$, onde $\delta_{pj}^{h} = \sum_{k=1}^{M} \delta_{pk}^{s} . w_{kj}^{s} . \left(1 - i_{pj}^{2}\right)$ e μ é a taxa de aprendizado.

3.7 Redes neurais e classificação estatística

Os modelos de redes neurais e a abordagem estatística (vide item 2.7.1) mantém um estreito relacionamento. Algumas semelhanças são conhecidas, como por exemplo, o fato de que muitos classificadores de redes neurais fornecem respostas na camada de saída que estimam probabilidade a posteriori Bayesianas. A precisão dessas estimativas depende da complexidade da rede, a quantidade de dados de treinamento utilizada e o grau em que a amostra de dados de treinamento reflete de forma verdadeira a distribuição de probabilidade das classes e as probabilidades a priori das classes [LIP91]. A rede BPN é um excelente classificador de padrões e bastante robusto. Sua taxa de convergência é alta. O treinamento supervisionado é ideal no caso desta implementação pois conhecemos a priori qual a saída que queremos para cada amostra das classes que o especialista irá selecionar.

4. Descrição do sistema IMASEG

4.1 Funcionamento do sistema

O objetivo principal do sistema é, a partir de uma imagem de entrada com regiões de texturas diferentes, gerar uma imagem de saída segmentada, onde cada textura da imagem original é pintada com uma cor diferente indicando a que classe pertence.

O sistema deverá carregar uma imagem, processá-la e gerar uma imagem de saída. Este processo pode ser resumido em três etapas:

A) Seleção de amostras.
B) Treinamento da rede neural.
C) Classificação da imagem.

4.1.1 Etapa A) Seleção de amostras

Inicialmente, deve-se identificar regiões da imagem características da classe a ser amostrada. Então, deve-se selecionar amostras representativas dessas classes de interesse. Esta seleção se dará através de uma janela de dimensão 3×3, 5×5 ou 7×7. A FIGURA 4.1 apresenta um exemplo, onde temos 4 texturas diferentes e o usuário selecionou 16 amostras características, 4 para cada textura (classe). Com isso, será especificado um arquivo, que será usado na fase de treinamento, contendo os valores dos *pixels*, e um rótulo binário correspondente a uma classe. A razão para usar uma janela de dimensões ímpar deve-se ao fato do pixel que está sendo analisado estar sempre no centro da janela. Isso é importante para o correto cálculo dos parâmetros de entrada conforme será visto no item 4.1.2.

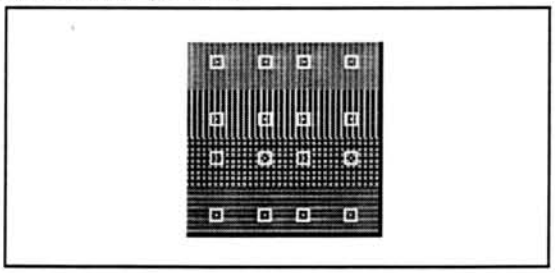

FIGURA 4.1-Seleção de amostras.

4.1.2 Etapa B) Treinamento da rede.

A rede neural será treinada a partir do arquivo de treinamento baseado nas amostras selecionadas na etapa anterior. A rede neural usada é o BPN. Conforme visto no item 3.6, treinar essa rede significa fornecer os valores de entrada e modificar os pesos conforme a saída desejada. Temos, então, que mapear um problema do mundo real para uma rede neural, gerando as entradas a partir das amostras selecionadas. Este talvez seja o processo mais crítico na aplicação de redes neurais para solução de problemas reais. No nosso caso específico, este mapeamento foi feito de duas formas

diferentes e independentes, usando as abordagens A ou B. Em ambas, durante esses processo, são apresentadas à rede as amostras das respectivas entradas e saídas desejadas contidas no arquivo de treinamento. O treinamento é encerrado quando a rede converge para um erro médio quadrado menor que 0,001.

Abordagem-A Treinamento com parâmetros de textura

Usaremos como entrada para a MLP, três parâmetros extraídos de cada amostra: média, desvio-padrão e uniformidade (FIGURA 4.2). A análise da escolha desses três parâmetros está no item 4.2. Desta forma, estamos quantificando a textura a fim de fornecer valores numéricos para a rede. A quantificação da textura foi abordada no item 2.10.5. Para calcular a média utilizou-se a equação (2.6) e para o desvio-padrão, utilizou-se as equações (2.9) e (2.8). A uniformidade é obtida através da equação (2.10), sendo que para isso é necessário antes calcular a matriz de coocorrência.

No caso do algoritmo de treinamento Backpropagation, os parâmetros devem ser escalonados para a faixa de intervalo entre −1 e +1, já que estamos usando a tangente hiperbólica como função de ativação. No item 4.4, será explicado como é feito o escalonamento. A rede terá 3 neurônios de entrada, um para cada parâmetro, e o número de neurônios da camada oculta é 4. Essa quantidade foi suficiente para classificar corretamente os padrões e fazer a rede convergir. Na camada de saída, teremos 3 neurônios, que podem classificar até 8 classes.

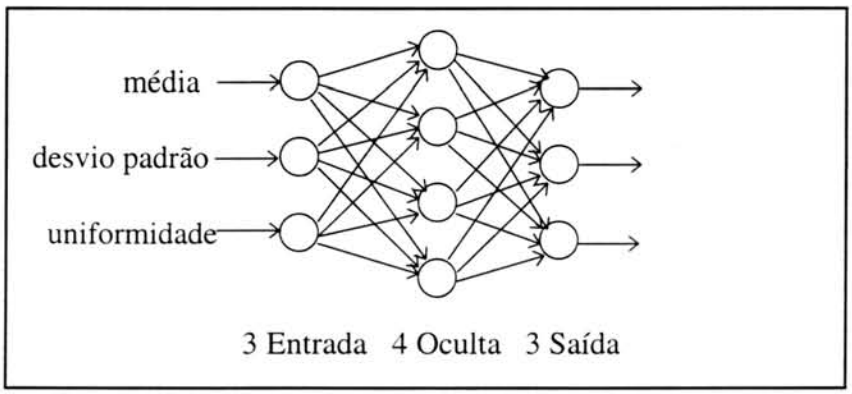

FIGURA 4.2-Rede MLP usada na Abordagem-A.

Abordagem-B Treinamento com níveis de cinza

Nesta abordagem, o mapeamento é feito diretamente a partir do nível de cinza dos *pixels* da janela para os neurônios de entrada. Os valores dos níveis de cinza dos *pixels* de uma janela são usados como vetor de entrada. O número de neurônios da camada de entrada será o mesmo que o número de pixels da janela. O escalonamento é mais simples (item 4.4), pois as entradas são todas do mesmo tipo. O número de neurônios da camada oculta foi estabelecido conforme cada caso. Na camada de saída, temos 3 neurônios, que podem classificar até 8 classes, a exemplo da abordagem-A. A FIGURA 4.3 apresenta um exemplo dessa abordagem, onde a janela tem dimensão (3×3) e a camada oculta tem 10 neurônios.

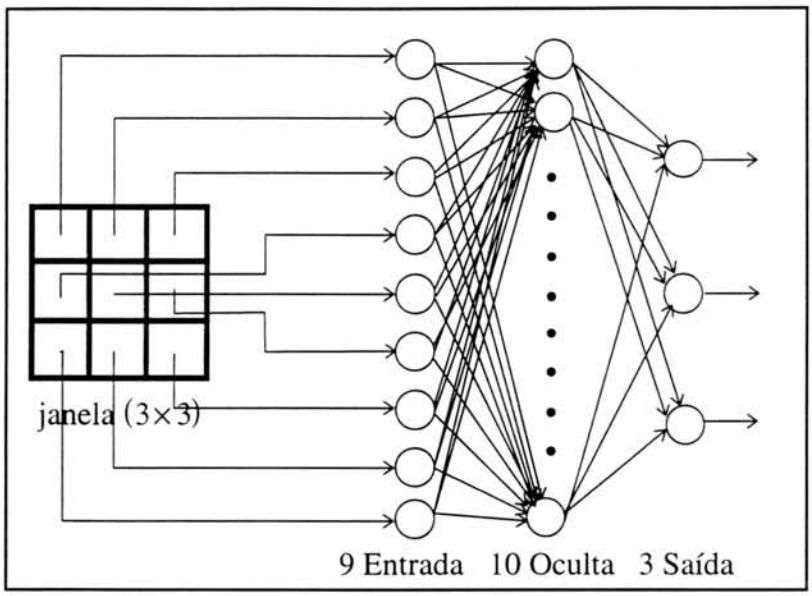

FIGURA 4.3-Rede MLP usada na Abordagem-B.

As relações espaciais entre os *pixels*, que aparentemente nesta abordagem, não estariam sendo levados em consideração, são indiretamente computados pois a posição em que cada pixel entra na rede neural, indica uma relação de vizinhança. Com a apresentação sucessiva e repetitiva de amostras de uma textura, espera-se que a RN aprenda automaticamente as feições médias dessa textura.

4.1.3 Etapa C) Classificação da imagem

Após o treinamento, processa-se a imagem, utilizando a mesma dimensão da janela utilizada na etapa A). O processamento é feito através da varredura da imagem. A janela é deslocada de unidade em unidade (*pixel*) da esquerda para a direita, linha por linha. A cada nova posição da janela calcula-se as entradas que são propagadas pela rede, gerando-se a saída da RN. É gerada uma imagem de saída onde o pixel central da janela é pintado com a cor correspondente à classe identificada pela RN de forma que, no final do processo, a imagem de saída esteja segmentada em regiões com cores diferentes, cada cor significando uma classe (textura).

4.2 Análise dos possíveis parâmetros de entrada da R.N.

A princípio, qualquer feição extraída da matriz de coocorrência é candidata a ser usada como parâmetro de entrada para a R.N. Conforme mencionado no item 2.10.5, Haralick propões várias feições que podem ser extraídas dessa matriz [HAR73]. Para selecionar as feições que seriam úteis para o trabalho, primeiramente descartou-se as que são de difícil ou demorado cálculo. Este é o caso da Correlação. Algumas outras também foram descartadas pois estão diretamente relacionadas, como é o caso da Variância e da Diferença de Variância. Feita essa pré-análise, os principais candidatos à parâmetro de entrada para a MLP são a média, variância, uniformidade, entropia e contraste. O objetivo é poder quantificar e diferenciar uma grande variedade de texturas

e estudar o comportamento e a utilidade desses parâmetros. Espera-se que a média seja o principal parâmetro a diferenciar as texturas. Em caso de duas texturas diferentes com sobreposição de valores da média, espera-se que o desvio-padrão auxilie na diferenciação. Se mesmo assim, houver sobreposição dos valores da média e desvio-padrão, e não for possível ainda a classificação, espera-se que a uniformidade das duas texturas sejam diferentes e assim por diante. Se, ao final, todos os parâmetros das duas texturas forem iguais ou parecidos, espera-se afirmar que elas são iguais ou muito parecidas.

No sistema IMASEG, foi feita uma análise desses parâmetros para verificar se realmente estavam cumprindo a função esperada. De imediato, a variância foi substituída pelo desvio-padrão pois conforme explicado no item 2.8.1, a variância, como medida de dispersão, apresenta a desvantagem de possuir dimensão diferente da dos dados observados, sendo esse problema eliminado pela utilização do desvio-padrão. Além dessa modificação, outras foram realizadas em função das observações do comportamento dos demais parâmetros durante a fase de testes do sistema IMASEG. Após a análise de diversas imagens, calculando-se os valores dos parâmetros, constatou-se algumas relações entre eles, descritas a seguir.

4.2.1 Relação entre desvio-padrão e contraste

A partir do cálculo e da tabulação dos valores do desvio-padrão e contraste em várias imagens, constatou-se que esses dois parâmetros tem comportamentos muito parecidos. Em alguns casos, eles tem valores idênticos. Como exemplos característicos, temos as TABELA 4.1 e TABELA 4.2 . Os dados da TABELA 4.1 foram gerados a partir das 16 amostras (janelas 7×7) indicadas na imagem da FIGURA 5.4a), enquanto que os dados da TABELA 4.2 correspondem à 8 amostras (janelas 5×5) extraídas de uma imagem médica (cineangiografia do ventrículo esquerdo). A TABELA 4.3 é a união dessas duas tabelas anteriores onde foram classificados em ordem ascendente os valores do desvio-padrão. Todos os valores das tabelas foram escalonados para a faixa entre -1.0 e +1.0, usando a equação da FIGURA 4.18. Os gráficos dos dados dessas tabelas, apresentados respectivamente nas FIGURA 4.4, FIGURA 4.5 e FIGURA 4.6, deixam clara essa relação.

Conclui-se, então, que não é útil usar esses dois parâmetros simultaneamente como entradas para a R.N. pois eles não cumprem o objetivo de diferenciar as texturas. A partir dessa constatação, decidiu-se que é mais vantajoso e mais rápido, usar um ou outro. Optou-se pelo desvio-padrão, pois ele é mais rápido de ser calculado em função de não depender da matriz de coocorrência.

TABELA 4.1-Desvio-padrão e contraste das amostras da imagem da FIGURA 5.4a.

	Desvio-padrão	Contraste
Janela A-1	+0,496	-0,280
Janela A-2	-0,984	-1,000
Janela A-3	-0,666	-0,900
Janela A-4	+1,000	+1,000
Janela A-5	+0,192	+0,135
Janela A-6	-0,074	-0,208
Janela A-7	+0,266	+0,259
Janela A-8	+0,242	+0,383
Janela A-9	-0,010	+0,165
Janela A-10	-1,000	-0,802
Janela A-11	-0,311	-0,044
Janela A-12	+0,097	-0,260
Janela A-13	+0,270	+0,242
Janela A-14	+0,515	+0,288
Janela A-15	-0,124	+0,167
Janela A-16	-0,348	-0,577

A FIGURA 4.4 apresenta o gráfico dos dados da TABELA 4.1.

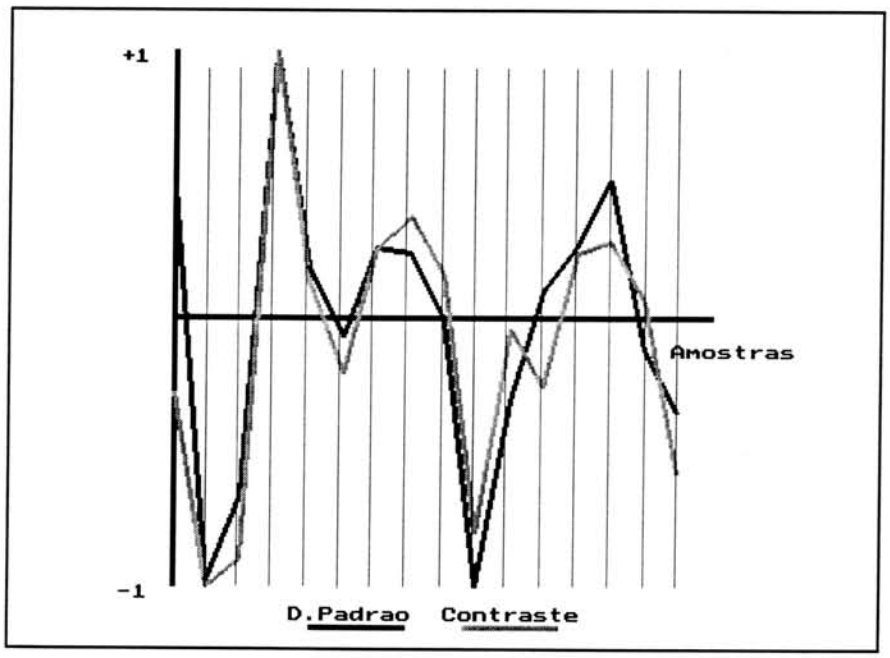

FIGURA 4.4-Gráfico dos dados da TABELA 4.1.

TABELA 4.2-Desvio-padrão e contraste das janelas da imagem médica.

	Desvio-padrão	Contraste
Janela B-1	+0,536	+0,425
Janela B-2	+0,492	+0,530
Janela B-3	+1,000	+1,000
Janela B-4	+0,456	+0,251
Janela B-5	-0,701	-0,840
Janela B-6	-0,644	-0,947
Janela B-7	-1,000	-1,000
Janela B-8	-0,222	-0,862

A FIGURA 4.5 apresenta o gráfico dos dados da TABELA 4.2.

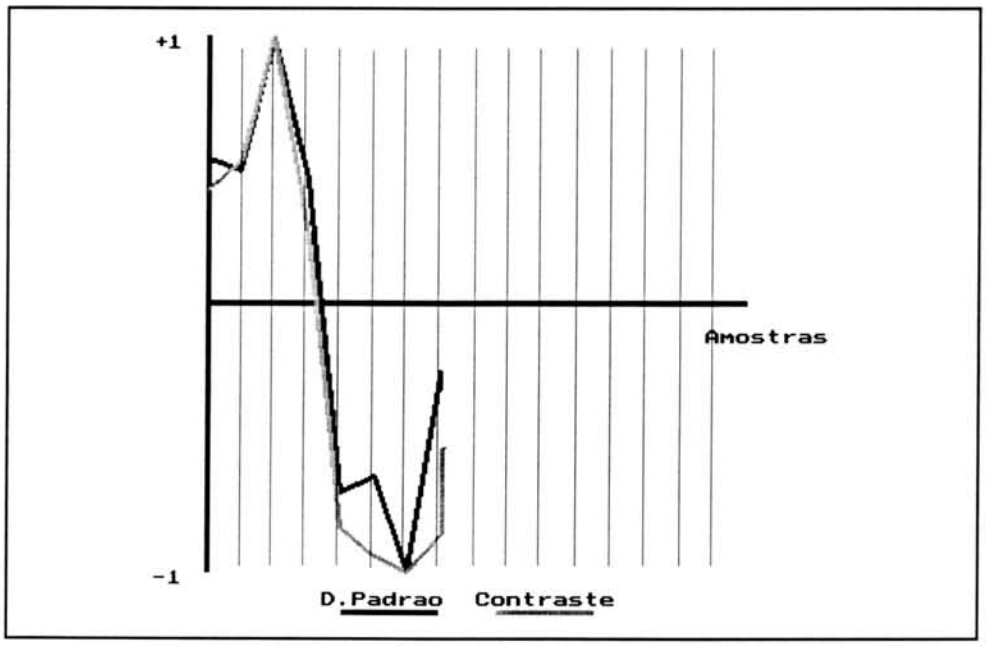

FIGURA 4.5-Gráfico dos dados da TABELA 4.2

TABELA 4.3-Junção das duas tabelas anteriores e ordenação pelo desvio-padrão.

	Desvio-padrão	Contraste
Janela A-10	-1,000	-0,802
Janela B-7	-1,000	-1,000
Janela A-2	-0,984	-1,000
Janela B-5	-0,701	-0,840
Janela A-3	-0,666	-0,900
Janela B-6	-0,644	-0,947
Janela A-16	-0,348	-0,577
Janela A-11	-0,311	-0,044
Janela B-8	-0,222	-0,862
Janela A-15	-0,124	+0,167
Janela A-6	-0,074	-0,208
Janela A-9	-0,010	+0,165
Janela A-12	+0,097	-0,260
Janela A-5	+0,192	+0,135
Janela A-8	+0,242	+0,383
Janela A-7	+0,266	+0,259
Janela A-13	+0,270	+0,242
Janela B-4	+0,456	+0,251
Janela B-2	+0,492	+0,530
Janela A-1	+0,496	-0,280
Janela A-14	+0,515	+0,288
Janela B-1	+0,536	+0,425
Janela A-4	+1,000	+1,000
Janela B-3	+1,000	+1,000

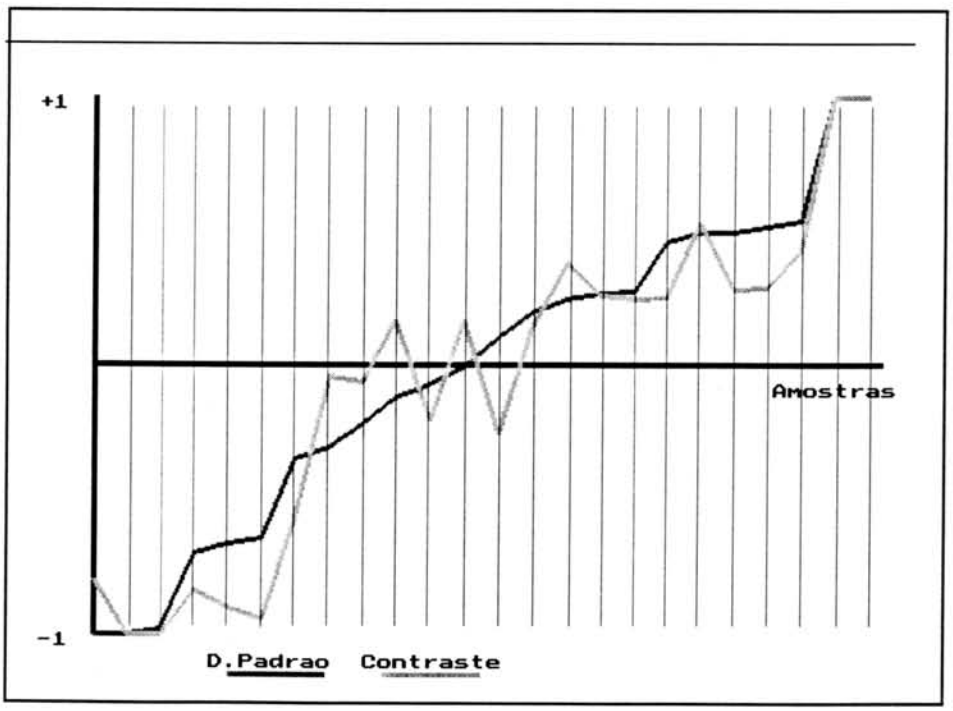

FIGURA 4.6-Gráfico dos dados da TABELA 4.3.

4.2.2 Relação entre uniformidade e entropia

Através das mesmas imagens utilizadas no item anterior, constatou-se que a uniformidade e a entropia tem comportamentos simétricos. Em seu trabalho, Haralick já havia observado esse fato [HAR73]. Essa relação é demostrada pelos dados das TABELA 4.4 e TABELA 4.5, as quais, a exemplo do item 4.2.1, foram extraídas das mesmas imagens. A TABELA 4.6 apresenta a junção das entradas das tabelas anteriores após a ordenação ascendente dos valores da uniformidade. Todos os valores das tabelas foram escalonados para a faixa entre -1.0 e +1.0, usando a equação da FIGURA 4.18

Percebe-se, então, que também não é útil usar esses dois parâmetros simultaneamente como entrada para a R.N. pois eles não cumprem o objetivo de diferenciar as texturas. Existe uma relação fixa entre eles. Esta relação pode ser aproximada em algumas situações por $h = -e$. Para um determinado valor de uniformidade, existirá apenas um valor correspondente para a entropia, fazendo com que seja mais vantajoso e mais rápido, usar um ou outro. Optou-se pela uniformidade, mas também poderíamos ter optado pela entropia sem maiores implicações.

TABELA 4.4-Uniformidade e entropia na imagem FIGURA 5.4a.

	Uniformidade	Entropia
Janela A-1	-0,375	+0,070
Janela A-2	+0,062	-0,322
Janela A-3	-0,887	+0,772
Janela A-4	-0,375	+0,092
Janela A-5	-0,362	+0,059
Janela A-6	-0,800	+0,635
Janela A-7	-0,900	+0,723
Janela A-8	-1,000	+1,000
Janela A-9	-0,825	+0,750
Janela A-10	+1,000	-1,000
Janela A-11	+0,062	-0,086
Janela A-12	-0,175	-0,051
Janela A-13	-0,800	+0,569
Janela A-14	-0,750	+0,564
Janela A-15	-0,850	+0,656
Janela A-16	-0,800	+0,589

A FIGURA 4.7 apresenta o gráfico dos dados da TABELA 4.4.

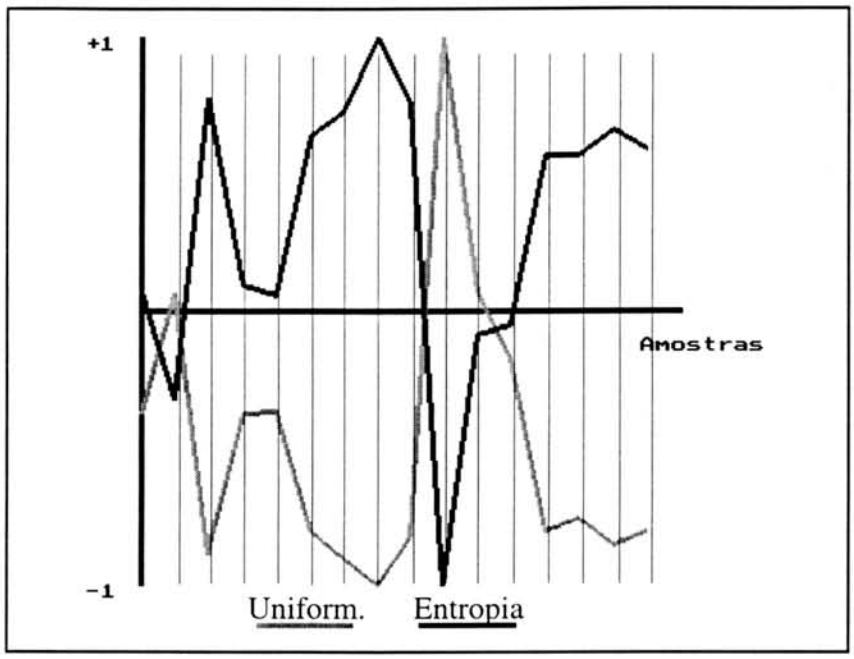

FIGURA 4.7-Gráfico dos dados da TABELA 4.4.

TABELA 4.5-Uniformidade e entropia na imagem médica.

	Uniformidade	Entropia
Janela B-1	-0,899	+0,815
Janela B-2	-0,662	+0,470
Janela B-3	-0,912	+0,846
Janela B-4	-1,000	+1,000
Janela B-5	+0,604	-0,620
Janela B-6	-0,209	-0,082
Janela B-7	+1,000	-1,000
Janela B-8	-0,209	-0,067

A FIGURA 4.8 apresenta o gráfico da TABELA 4.5.

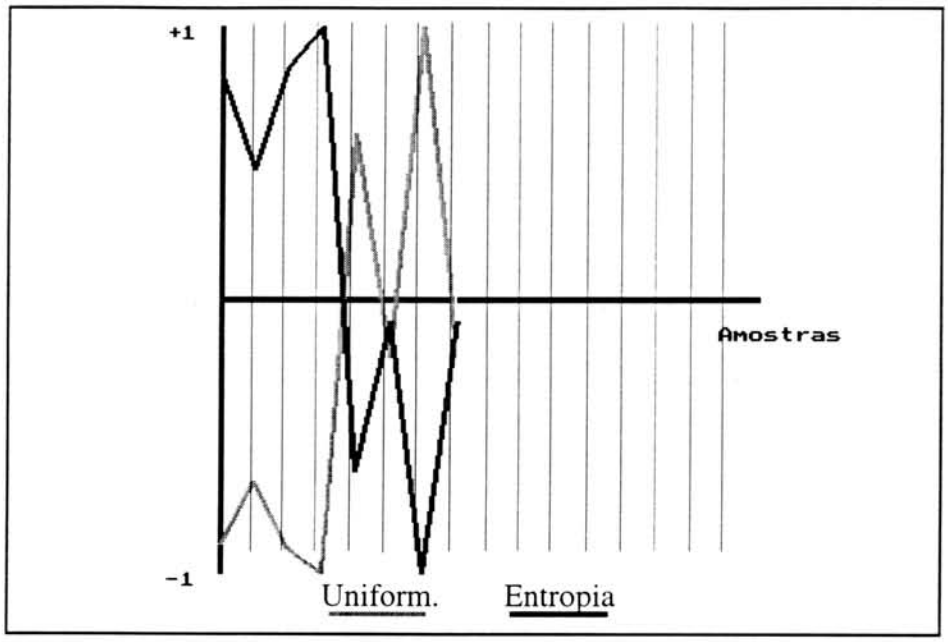

FIGURA 4.8-Gráfico dos dados da TABELA 4.5.

TABELA 4.6-Junção das tabelas anteriores ordenada pela uniformidade.

	Uniformidade	Entropia
Janela B-4	-1,000	+1,000
Janela A-8	-1,000	+1,000
Janela B-3	-0,912	+0,846
Janela A-7	-0,900	+0,723
Janela B-1	-0,899	+0,815
Janela A-3	-0,887	+0,772
Janela A-15	-0,850	+0,656
Janela A-9	-0,825	+0,750
Janela A-6	-0,800	+0,635
Janela A-13	-0,800	+0,569
Janela A-16	-0,800	+0,589
Janela A-14	-0,750	+0,564
Janela B-2	-0,662	+0,470
Janela A-1	-0,375	+0,070
Janela A-4	-0,375	+0,092
Janela A-5	-0,362	+0,059
Janela B-6	-0,209	-0,082
Janela B-8	-0,209	-0,067
Janela A-12	-0,175	-0,051
Janela A-2	+0,062	-0,322
Janela A-11	+0,062	-0,086
Janela B-5	+0,604	-0,620
Janela B-7	+1,000	-1,000
Janela A-10	+1,000	-1,000

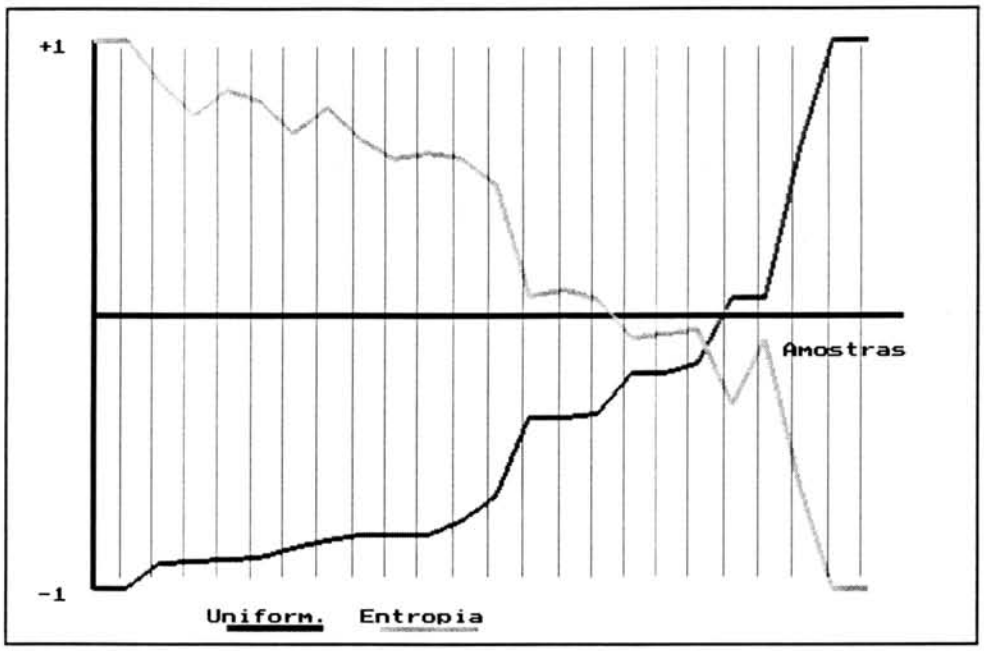

FIGURA 4.9-Gráfico dos dados da TABELA 4.6.

4.2.3 Redimensionamento da uniformidade

Durante os teste, descobriu-se uma característica importante da uniformidade. Para imagens com 256 níveis de cinza, esse parâmetro não atinge o seu objetivo de diferenciar imagens com uniformidades diferentes. Isso deve-se ao fato de que nessas imagens, existe pouca probabilidade de dois pixels vizinhos terem exatamente o mesmo nível de cinza, que é o fator que aumenta a uniformidade da imagem. Por exemplo, observando-se as imagens da FIGURA 4.10, percebe-se que a imagem da esquerda possui uma uniformidade visual muito maior do que a imagem da direita. Entretanto, devido à forma como a matriz de coocorrência e a uniformidade são calculadas, essas duas imagens terão o mesmo valor da uniformidade pois possuem todos os pixels diferentes entre si, fazendo com que a uniformidade seja mínima (vide item 4.3.1). Para evitar esse problema e fazer com que o parâmetro retrate realmente a uniformidade da imagem, reduz-se a imagem de 256 para 16 níveis de cinza, atribuindo-se o valor inteiro da divisão do valor do *pixel* por 16. Essa redução é feita apenas para cálculo da uniformidade, sem afetar a imagem original que continua com 256 níveis de cinza. Para Essa modificação nos leva aos resultados da FIGURA 4.11, onde os valores da uniformidade são realmente diferentes, retratando a realidade com mais fidelidade. Para a imagem da esquerda, o valor de h passou de 0,003205128 para 0,151442 e na imagem da direita o valor de h passou de 0,003205128 para 0,0013971 que também é um valor baixo pois realmente essa imagem não é nada uniforme.

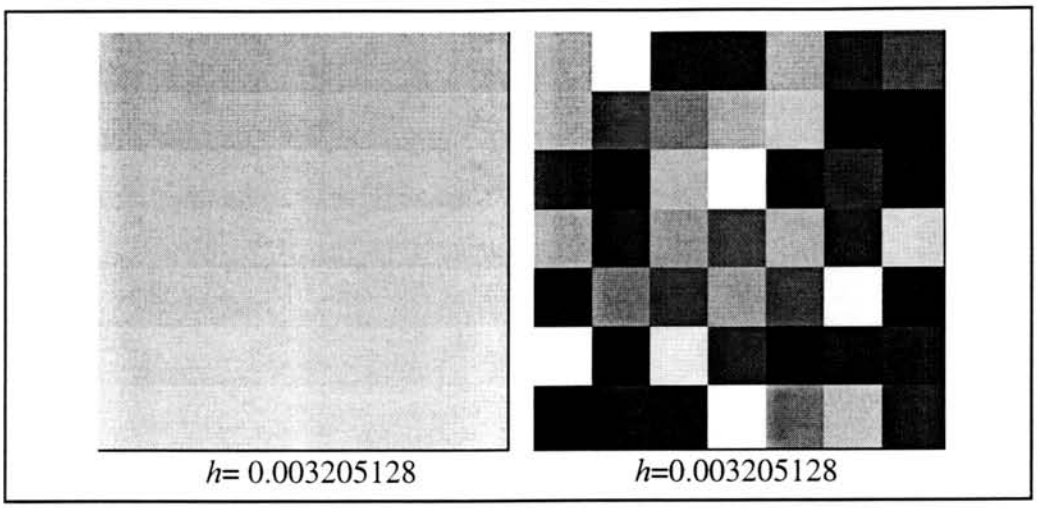

FIGURA 4.10-Duas imagens diferentes com mesma uniformidade.

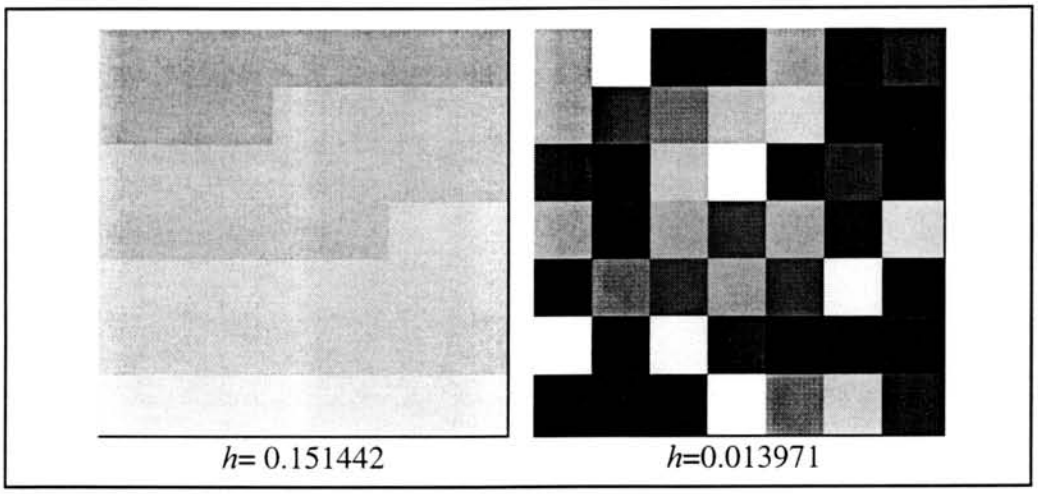

FIGURA 4.11-As mesmas imagens após a redução para 16 níveis de cinza.

4.3 Comportamento esperado dos três parâmetros de entrada da RN

Após a análise feita no item 4.2, ficamos, então, com três parâmetros para serem utilizados no sistema: média, desvio-padrão e uniformidade. Consideraremos algumas situações diferentes que exemplificam a utilidade desses três parâmetros. Essas situações serão reproduzidas nos testes do capítulo 5.

1) Considere a imagem (12×12) da FIGURA 4.12. Nela, temos duas regiões com texturas diferentes (A e B). Para cada textura, temos 2 amostras representativas (janelas 3×3). Se calcularmos a média das amostras, veremos que o intervalo de valores das amostras da textura A são diferentes e não se sobrepõem ao intervalo de valores das amostras da textura B. Portanto, neste caso, o uso de somente um parâmetro, a média, já é suficiente para diferenciar essas duas texturas.

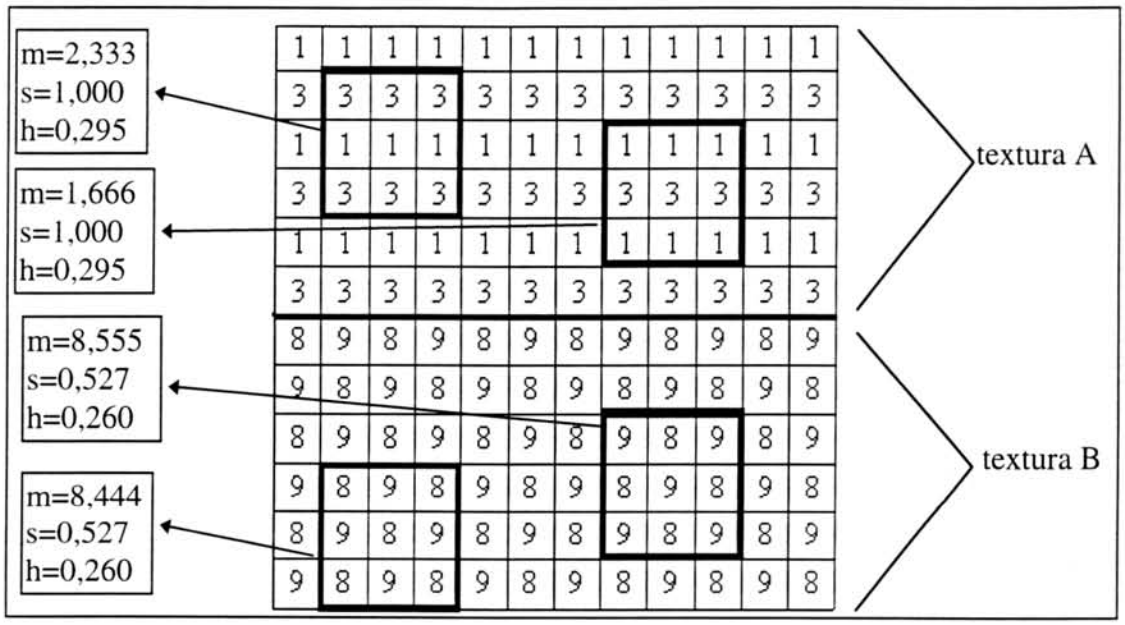

FIGURA 4.12-Duas texturas com médias diferentes.

2) Considere agora, a imagem (12×12) da FIGURA 4.13. As amostras das texturas A e B tem intervalo de valores da média que se sobrepõem. Essa situação atrapalha a RN, pois estamos associando valores de entrada semelhantes para classes diferentes. Portanto, só conseguimos distingui-las, calculando também o desvio-padrão. Neste caso, verificamos que os intervalos de valores para o desvio-padrão não estão se sobrepondo.

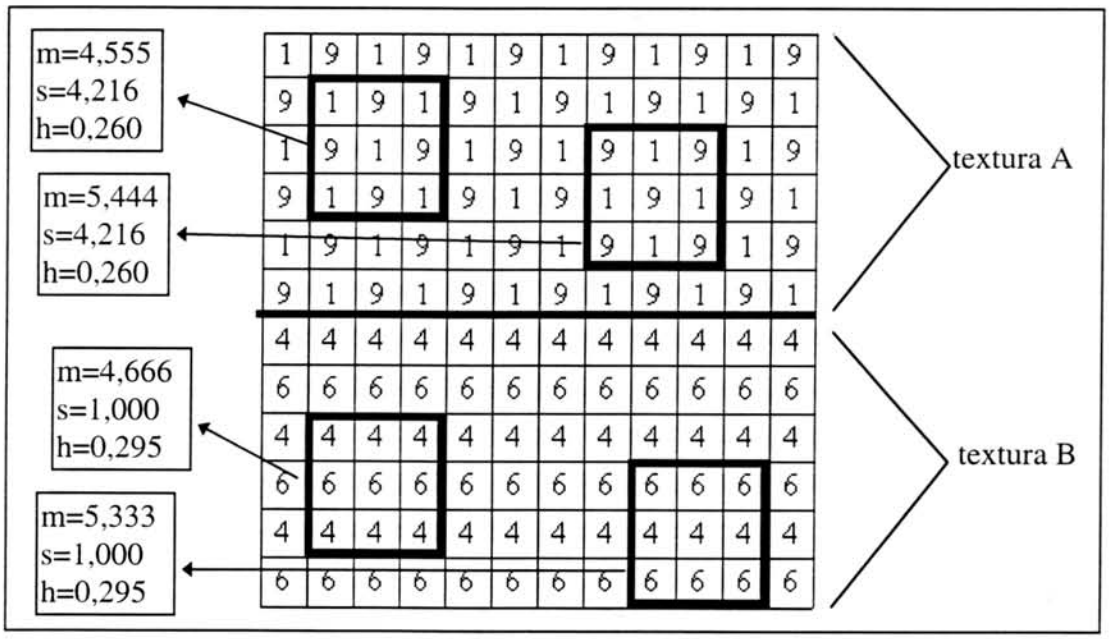

FIGURA 4.13-Duas texturas com médias próximas e desvio-padrão diferentes.

3) Considere, nesta terceira situação, a imagem da FIGURA 4.14. As amostras das texturas A e B tem intervalos de valores da média e da desvio-padrão que se sobrepõem. Repetindo a situação anterior, a classificação da RN fica mais confusa ainda, apesar do

erro médio do algoritmo BP convergir corretamente. Neste caso, a única característica que podemos usar para distinguir as texturas é a uniformidade (h).

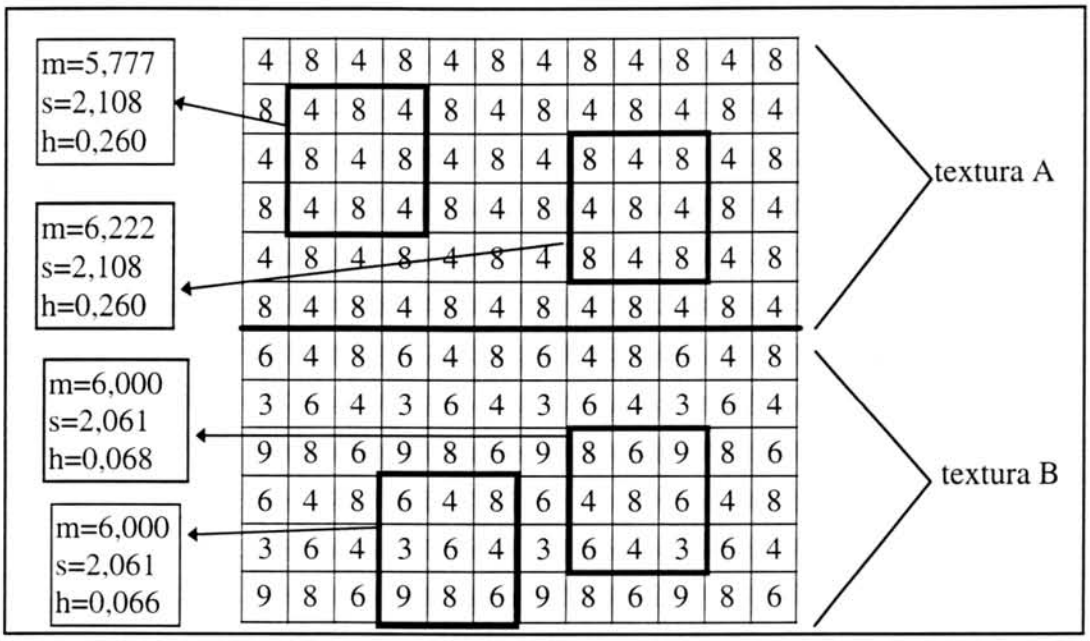

FIGURA 4.14-Duas texturas com médias e desvio-padrão próximos mas com uniformidade diferentes.

4.3.1 Valores mínimo e máximo absolutos dos parâmetros de entrada

O cálculo dos valores mínimo e máximo absolutos é útil para sabermos como se comportam esses parâmetros e para podermos compará-los. A partir desses valores, foi possível fazer uma análise sobre qual o melhor escalonamento a aplicar sobre os mesmos (item 4.4)

Média: Na equação da média (expressão 2.6), podemos substituir |P| por M×N.

Mínimo: Vamos supor que todos os pixels tenham o mesmo valor e sejam iguais à zero. Neste caso, podemos substituir $\sum_{ij} P_{ij}$ por $0\times(M\times N)$. Então: $min = \frac{0\times(\mathrm{M}\times\mathrm{N})}{(\mathrm{M}\times\mathrm{N})} = 0$.

Máximo: Assumindo todos os pixels iguais à L, temos: $max = \frac{L\times(\mathrm{M}\times\mathrm{N})}{(\mathrm{M}\times\mathrm{N})} = L$.

Desvio-padrão: Usando a equação da expressão 2.8, temos:

Mínimo: Para que a desvio-padrão seja mínimo, a diferença do numerador deve ser mínima, portanto P_{ij} deve ser igual à $\overline{m}$. Supondo todos os pixels iguais à um valor qualquer K, então $\overline{m} = K$. Substituindo, temos:

$$min = \sqrt{\frac{(K-K)^2\times(M\times N)}{(M\times N)-1}} = \sqrt{\frac{0\times(M\times N)}{(M\times N)-1}} = 0$$

Máximo: Analogamente, para que a desvio-padrão seja máxima, a diferença do numerador deve ser máxima. Chega-se à esta situação, assumindo que metade dos pixels

da janela tem valor 0 e a outra metade valor L. Neste caso, $\overline{m} = \left(L/2\right)$, fazendo com que a diferença $\left(P_{ij} - \overline{m}\right)^2$ seja máxima. Substituindo:

$$max = \sqrt{\frac{\left(0-\frac{L}{2}\right)^2 \times \left(\frac{M \times N}{2}\right) + \left(L-\frac{L}{2}\right)^2 \times \left(\frac{M \times N}{2}\right)}{M \times N}} =$$

$$= \sqrt{\frac{\left(\left(-\frac{L}{2}\right)^2 + \left(L-\frac{L}{2}\right)^2\right) \times \left(\frac{M \times N}{2}\right)}{M \times N - 1}} = \sqrt{\frac{2 \times \left(\frac{L}{2}\right)^2 \times \frac{(M \times N)}{2}}{M \times N - 1}} = \frac{L}{2} \times \sqrt{\frac{M \times N}{M \times N - 1}}.$$

Uniformidade: Usando a equação da expressão 2.10, temos:

Mínimo: O mínimo seria obtido se todos elementos da matriz C^N tivessem valor 0. Entretanto, pela maneira como a matriz é construída, isso não é possível, pois sabemos que seu somatório é igual à 1. Então, para que o somatório seja mínimo, é necessário que os elementos da matriz tenham valor o mais baixo possível. Isto é conseguido, supondo que todos os pixels da imagem que gerou a matriz de coocorrência sejam diferentes entre si. A matriz de coocorrência normalizada terá n elementos com valor igual à $1/\sum_{i,j} C_{ij}^S$, onde n é a quantidade de pares de pixels vizinhos da imagem ($\#R^S$), calculado no item 2.10.2. Podemos substituir $\#R^S$ por $\sum_{i,j} C_{ij}^S$, conforme item 4.3.2. Substituindo, temos:

$$min = \left(1/\sum_{i,j} C_{ij}^S\right)^2 \times \sum_{i,j} C_{ij}^S = 1/\sum_{i,j} C_{ij}^S$$

Máximo: Ao contrário, para que o somatório seja máximo, deve existir apenas um elemento da matriz (n=1) com valor máximo e todos os outros iguais à zero. Isto é conseguido, quando todos os pixels da imagem são iguais entre si. Este elemento diferente de zero terá valor igual ao somatório da matriz normalizada, ou seja, 1. Portanto,

$max = 1^2 \times 1 = 1$

Entropia: Usando a equação da expressão 2.11, temos:

Mínimo: Para obter-se um mínimo de entropia, é necessário que apenas um elemento da matriz de coocorrência normalizada tenha valor 1, pois neste caso 1.log1 = 0. Da mesma forma que no item anterior, isso é conseguido a partir de uma imagem com todos pixels iguais entre si. Portanto, $min = 1 \times \log 1 \times 1 = 0$.

Máximo: O máximo seria obtido se todos elementos da matriz C^N tivessem valor de 0,36 pois nesse caso $-0{,}36 \times \log(0{,}36)$ atinge o valor máximo, conforme gráfico da FIGURA 2.8. Entretanto, pela maneira como a matriz é construída, isso não é possível. Então, para que a entropia seja máxima, é desejável que todos valores da matriz estejam o mais próximo possível de 0,36. Isto é conseguido supondo que todos os pixels da imagem que gerou a matriz de coocorrência sejam diferentes entre si, reproduzindo a situação de uniformidade mínima. Desta forma, substituindo na equação, temos:

$$max = -\left(1/\sum_{i,j} C_{ij}^{S}\right) \times \log\left(1/\sum_{i,j} C_{ij}^{S}\right) \times \sum_{i,j} C_{ij}^{S} = -\log\left(1/\sum_{i,j} C_{ij}^{S}\right)$$

Contraste: Usando a equação da expressão 2.12, temos:

Mínimo: O mínimo de contraste é obtido fazendo com que qualquer elemento da matriz de coocorrência diferente de 0 fique situado na diagonal principal, fazendo com que li-jl seja zero anulando o somatório. Se, numa imagem com 256 níveis de cinza, todos os pixels forem iguais a um valor qualquer Z, então a matriz de coocorrência normalizada terá a seguinte forma:

	0	...	Z	...	256
0	0	...	0	...	0
	...	...	0	...	...
Z	0	0	1	0	0
	...	...	0	...	...
256	0	...	0	...	0

Desta forma, o contraste mínimo será $(Z-Z)^2 \times 1 = 0$.

Máximo: Para obtermos máximo de contraste, os elementos $(0,L)$ e $(L,0)$ da matriz de coocorrência devem ser máximos, onde L=255, fazendo com que $(L-0)^2$ também tenha valor máximo. Após várias simulações intuitivas, esta situação foi obtida a partir de uma imagem com as características abaixo (à esquerda na FIGURA 4.15). Assim, a matriz de coocorrência terá a forma abaixo (à direita na FIGURA 4.15).

0	0	0	0	0
L	L	L	L	L
0	0	0	0	0
L	L	L	L	L
0	0	0	0	0

	0	.....	L
0	$\frac{\frac{3}{7}\times \# R^{H}}{\# R^{S}}$	...	$\frac{\#R^{V}+\#R^{DD}+\#R^{DE}}{2\times \#R^{S}}$
	...	...	...
L	$\frac{\#R^{V}+\#R^{DD}+\#R^{DE}}{2\times \#R^{S}}$	...	$\frac{\frac{4}{7}\times \# R^{H}}{\# R^{S}}$

FIGURA 4.15-Imagem que gera valor máximo de contraste e sua respectiva matriz de coocorrência.

O máximo será dado por:

$$2 \times \frac{\# R^{V} + \# R^{DD} + \# R^{DE}}{2 \times \# R^{S}} \times (L-0)^2 = \frac{\# R^{V} + \# R^{DD} + \# R^{DE}}{\# R^{S}} \times L^2$$

A FIGURA 4.16 tabula todos os valores mínimo e máximo dos parâmetros de entrada para a rede neural.

Média	$min = 0$	$max = L - 1$
D.Padrão	$min = 0$	$max = \frac{L}{2} \times \sqrt{\frac{M \times N}{M \times N - 1}}$
Homogen.	$min = 1 / \sum_{i,j} C_{ij}^{S}$	$max = 1$
Entropia	$min = 0$	$max = -\log\left(1 / \sum_{i,j} C_{ij}^{S}\right)$
Contraste	$min = 0$	$max = \frac{\#R^{V} + \#R^{DD} + \#R^{DE}}{\#R^{S}} \times L^{2}$

FIGURA 4.16-Valores mínimo e máximo absolutos dos parâmetros de entrada.

Analisando os dados da FIGURA 4.16, concluímos que o intervalo de valores da média depende somente da quantidade de níveis de cinza (L) da janela, o desvio-padrão depende de L e da dimensão da janela ($M \times N$), enquanto que os demais dependem da matriz de coocorrência normalizada C^{N} que por sua vez depende das dimensões da janela ($M \times N$), que foi usada para gerá-la.

4.3.2 Somatório dos elementos da matriz de coocorrência

O somatório dos elementos da matriz de coocorrência $\sum_{i,j} C_{ij}$ será exatamente igual à quantidade de pares de pixels vizinhos da imagem usada para gerá-la, conforme FIGURA 4.17.

$$\sum_{i,j} C_{ij}^{H} = \#R^{H} \quad , \quad \sum_{i,j} C_{ij}^{V} = \#R^{V},$$
$$\sum_{i,j} C_{ij}^{DD} = \#R^{DD} \quad , \quad \sum_{i,j} C_{ij}^{DE} = \#R^{DE},$$
$$\sum_{i,j} C_{ij}^{S} = \#R^{S} \quad , \quad \sum_{i,j} C_{ij}^{N} = 1.0$$

FIGURA 4.17-Somatório dos elementos da matriz de coocorrência.

4.4 Escalonamento dos parâmetros de entrada

No sistema IMASEG, o algoritmo BackPropagation foi implementado usando função tangente hiperbólica como função de ativação. Portanto, os valores de entrada devem se situar entre −1 e +1 para que se consiga convergir o erro. Caso alguma entrada tenha valor muito fora desta faixa, ela irá saturar a rede. Devemos então, escalonar os valores dos parâmetros para esse intervalo. Esta tarefa não é trivial e é fundamental para o funcionamento do sistema.

Os três parâmetros de entrada da rede B.P. ($\overline{m}$, s, h) tem valores que se situam em faixas completamente diferentes (vide item 4.3.1). Além disso, seus intervalos de variação não são sempre os mesmos, variando conforme o número de níveis de cinza da imagens e conforme o tamanho da janela. Portanto, a tarefa de escaloná-los para a faixa

entre −1 e +1 é delicada. É preciso fazer uma análise do comportamento dessas variáveis em diversas situações, levando-se em consideração que a textura apresenta-se sob uma grande variedade de formas. Ela pode variar com a distribuição dos pixels dentro da janela e com os valores de nível de cinza desses pixels. Essa combinação nos leva à uma infinidade de texturas diferentes. Abaixo, propomos 3 formas para realizar o escalonamento: pelos valores mínimo e máximo absolutos, pelos valores mínimo e máximo relativos ou pela função tanh(x). Cada uma delas apresenta vantagens e desvantagens e sua adequação dependerá das características das texturas presentes na imagem que estamos querendo classificar.

4.4.1 Escalonamento linear pelos valores *min* e *max* absolutos.

Nesta abordagem, escalona-se os parâmetros tomando como base os valores mínimo (*min*) e máximo (*max*) absolutos que os parâmetros podem assumir (item 4.3.1). Aplica-se esses valores na equação da FIGURA 4.18, obtendo o valor do parâmetro escalonado.

$$E_{escalonada} = 2 \times \left(\frac{E - min}{max - min} \right) - 1.0$$

FIGURA 4.18-Escalonamento das entradas (E) para [-1,+1].

A vantagem desse tipo de escalonamento é que não precisamos ter um conhecimento das características da imagem, pois já conhecemos *min* e *max* a priori, que é igual para todas imagens (para um mesmo tamanho de amostra). Portanto, esse é um método bastante genérico. Entretanto, corre-se o risco bastante grande de igualar valores que teriam que ser diferenciados. Por exemplo, na TABELA 4.7, temos os valores da média e da média escalonada das quatro amostras (duas para cada textura) da imagem na FIGURA 4.12. Percebe-se que, após o escalonamento, os valores da média escalonada das duas texturas ficam muito próximos, quando o que se pretende é exatamente o contrário, ou seja, diferenciar essas 2 texturas. Os valores da tabela foram calculados levando-se em consideração os valores 0 e 255 para o mínimo e o máximo da média respectivamente.

TABELA 4.7-Média e média escalonada pelo escalonamento linear absoluto.

	Média	Média escalonada
Textura A-amostra 1	2.333	–0,9817019
Textura A-amostra 2	1.666	–0,9869333
Textura B-amostra 1	8.444	–0,9337725
Textura B-amostra 2	8.555	–0,9329019

4.4.2 Escalonamento linear pelos valores *min* e *max* relativos

Para solucionar o problema gerado pelo escalonamento descrito acima, os valores *min* e *max* do parâmetros não serão mais os seus valores absolutos e sim o mínimo e o máximo encontrados nas amostras da própria imagem. Aplica-se esses valores na equação da FIGURA 4.18, porém *min* e *max* serão os valores mínimo e máximo relativos e não absolutos. Desta forma, após o escalonamento, os valores estarão bem distribuídos entre −1 e +1, evitando-se a "perda de informação" (TABELA 4.8).

TABELA 4.8-Média e média escalonada pelo escalonamento linear relativo.

	Média	Média escalonada
Textura A-amostra 1	2.333	−0,8063579
Textura A-amostra 2	1.666	−1.0000000
Textura B-amostra 1	8.444	+0.9677747
Textura B-amostra 2	8.555	+1.0000000

Mesmo assim, em algumas situações, esse escalonamento também não será satisfatório. Por exemplo, considere a TABELA 4.9 que representa os valores da média de 3 texturas da imagem da FIGURA 4.19. Como a média das amostras das texturas A e B está muito afastada da média da amostra da textura C, os valores escalonados da média das amostras das texturas A e B acabam ficando praticamente iguais, quando o que se pretendia era diferenciá-los.

Outra desvantagem dessa técnica é que precisamos realizar um pré-processamento da imagem para obter os valores mínimo e máximo relativos. Porém, isso não é demorado pois necessita-se de uma única varredura na imagem para obter esses valores.

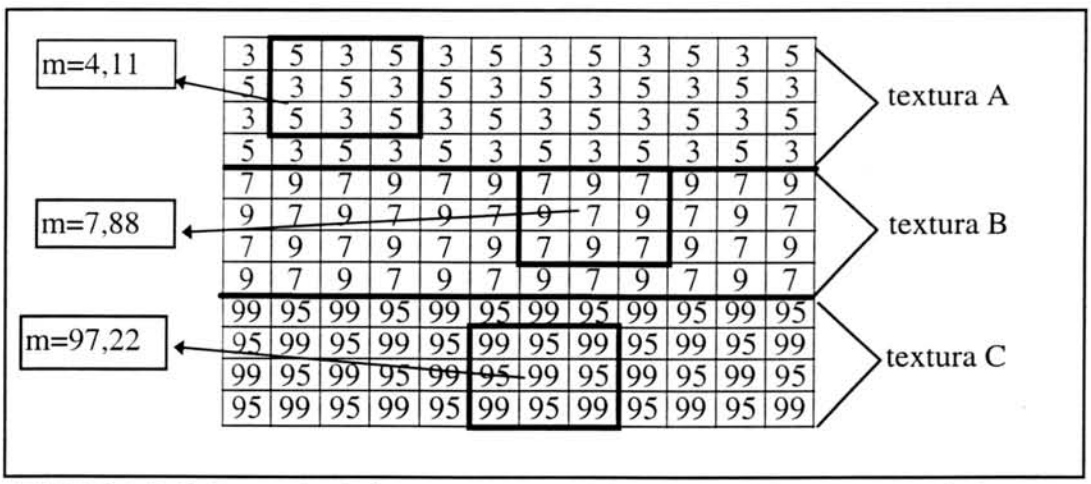

FIGURA 4.19-Exemplo de imagem com problema no escalonamento relativo.

TABELA 4.9-Problema na média escalonada gerado pelo escalonamento relativo.

	Média	Média escalonada
Textura A-amostra 1	4,11	−1,000
Textura B-amostra 1	7,88	−0,961
Textura C-amostra 1	97,22	+1,000

4.4.3 Escalonamento com auxílio da função tangente hiperbólica

Considere o comportamento da função $\tanh(\alpha \times (x - \lambda))$desenhada na FIGURA 4.20. Esse tipo de função resolveria o problema criado na situação anterior, onde os valores estão mal distribuídos. A inclinação da curva deve ser ajustada para cobrir a região onde os valores do parâmetro estão mais concentrados de forma a distribuí-los mais igualmente no intervalo −1;+1. Para isso, a função pode ser "calibrada" através dos seus parâmetros α, λ. Com α, regula-se a inclinação da curva e com λ, o deslocamento. Por exemplo, na FIGURA 4.20 a), os valores estão concentrado em torno de 0,8. Portanto, o valor de λ será 0,8. Já na FIGURA 4.20b), o valor de λ será 1,5, pois a concentração de valores está nessa faixa. Na FIGURA 4.20c), o intervalo de valores é mais alongado, fazendo com que o parâmetro α tenha valor 2, enquanto que as duas funções anteriores tinham valor 4. Entretanto, essa técnica exige, além de um pré-processamento, uma análise estatística do comportamento da distribuição dos valores dos parâmetros da imagem. Essa tarefa deve ser feito fora do sistema, manualmente. Porém, se a intenção for classificar imagens sempre do mesmo tipo, então essa análise seria feita uma única vez, e seu resultado embutido no programa, que seria capaz de classificar apenas imagens daquele tipo específico.

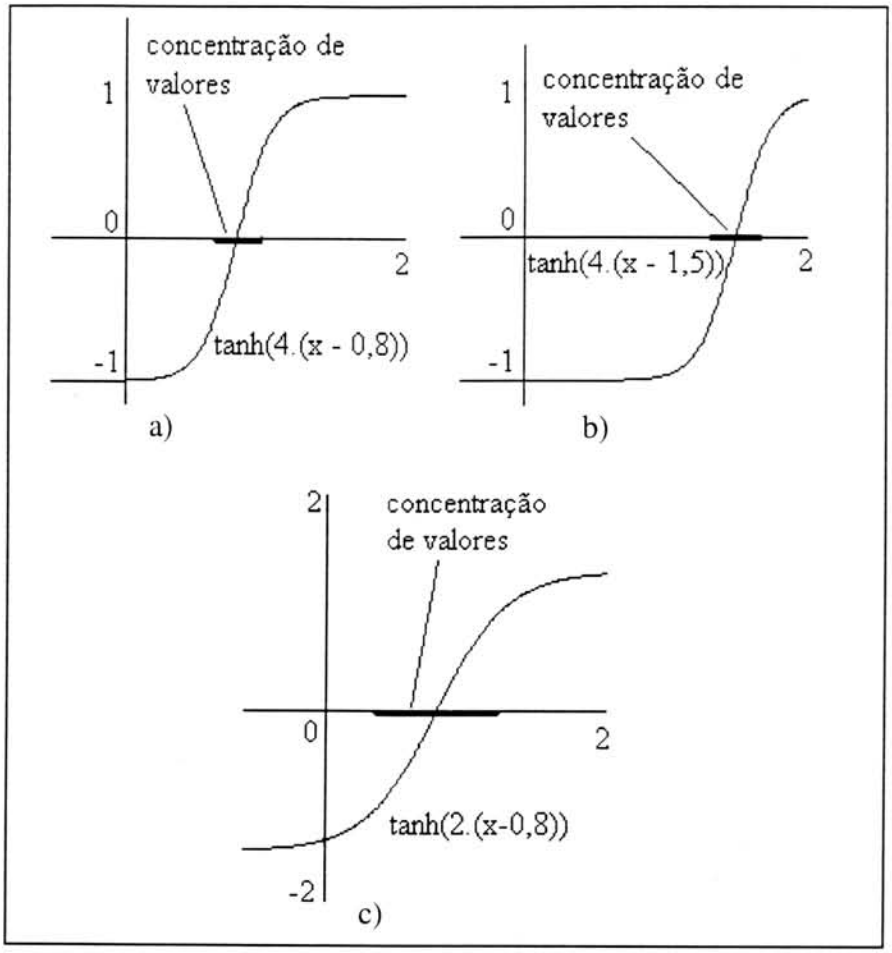

FIGURA 4.20-Escalonamento pela função tangente hiperbólica.

5. Resultados e Discussão

Neste capítulo, são mostrados e analisados os resultados obtidos com o processamento de algumas imagens pelo sistema descrito no capítulo 4. São descritas as imagens utilizadas para testes e todos os dados e parâmetros utilizados, visando possibilitar a repetição dos experimentos com a mesma precisão.

5.1 Descrição das imagens utilizadas

Todas as imagens são sintéticas monocromáticas com 256 níveis de cinza com dimensão 100×100 *pixels*. Essa dimensão é suficiente para testar o sistema, pois as janelas de deslocamento tem dimensão de 3×3, 5×5 e 7×7. As texturas são bastante finas sendo bem caracterizadas com janelas dessas dimensões. Todas imagens possuem 4 texturas diferentes a classificar. Essas texturas podem aparecer em mais de uma região da imagem. O usuário seleciona até quatro amostras representativas de cada textura.

5.2 Valores de variáveis e parâmetros utilizadas.

5.2.1 Valores usados dos parâmetros da rede Backpropagation

Os pesos (w) foram inicializados com valores randômicos entre −1,00 e +1,00. O valor usado para μ foi de 0,02. Na fase de treinamento, o número de iterações variou de imagem para imagem, e foi estabelecido como final de treinamento um erro médio quadrado menor que 0,001. A quantidade de neurônios na camada oculta foi 4 para a abordagem-A. Essa quantidade foi suficiente para classificar corretamente os padrões e fazer a rede convergir. Já na abordagem-B, o dimensionamento da camada oculta é mais difícil. O número de neurônios variou de 10 à 25, dependendo da imagem. Com menos de 10 neurônios, a rede torna-se instável, não convergindo em alguns casos. Os demais dados (forma de treinamento, valores de entrada, etc) estão descritos no item 4.1.

5.2.2 Valores usados no cálculo da matriz de coocorrência (Abordagem-A)

Conforme visto no item 2.10, existem várias matrizes de coocorrência dependendo da orientação espacial que estamos interessados. No nosso caso, como não há um conhecimento a priori das texturas à classificar, utilizamos as quatro orientações possíveis (horizontal, vertical, diagonal direita e diagonal esquerda) para gerar a matriz de coocorrência normalizada C^N. A FIGURA 5.1 descreve o cálculo de cada elemento da matriz de coocorrência usada no sistema. Esta abordagem consumiu um tempo um pouco maior no cálculo da matriz, mas garantiu que nenhuma feição da textura fosse ignorada.

$$C_{ij}^{N} = \frac{C^{H} + C^{V} + C^{DD} + C^{DE}}{\#R^{H} + \#R^{V} + \#R^{DD} + \#R^{DE}}$$

FIGURA 5.1-Matriz de coocorrência utilizada no sistema.

Todos elementos da equação FIGURA 5.1 foram vistos no item 2.10. Essa equação também garante que a matriz seja invariante sob rotação da imagem, fazendo com que o valor da uniformidade seja igual. Por exemplo, na FIGURA 5.2, a imagem da direita está rotacionada de 90^0 em relação a imagem da esquerda. Mesmo assim, a matriz de coocorrência e o valor da uniformidade serão os mesmos para as duas imagens, garantindo que texturas iguais terão valores da homogeneidade iguais mesmo que rotacionadas.

Para o parâmetro *d* (distância entre pixels) foi utilizado o valor unitário, pois é o valor que apresenta melhores resultados nos testes realizados por diversos pesquisadores. Apesar de poder assumir qualquer valor, na prática valores maiores que 1 não são úteis pois pares de pixels separados por distância maior que 1 tornam-se não correlacionados [HAR79].

53	50	53	50	53	50	53
53	50	53	50	53	50	53
53	50	53	50	53	50	53
53	50	53	50	53	50	53
53	50	53	50	53	50	53
53	50	53	50	53	50	53
53	50	53	50	53	50	53

h= 0.303994

53	53	53	53	53	53	53
50	50	50	50	50	50	50
53	53	53	53	53	53	53
50	50	50	50	50	50	50
53	53	53	53	53	53	53
50	50	50	50	50	50	50
53	53	53	53	53	53	53

h=0.303994

FIGURA 5.2-Invariância da matriz de coocorrência quanto à rotação da imagem.

Um grande problema encontrado na implementação do sistema IMASEG foi quanto à dimensão da matriz de coocorrência $\left(N_g \times N_g\right)$ onde N_g é a quantidade de níveis de cinza da imagem. Como estamos trabalhando com imagem com 256 níveis de cinza, a matriz terá dimensão muito grande (256×256). A geração dessa matriz e o cálculo para extrair o parâmetro *h* a partir dela se tornam muito lentos, dependendo do tipo de imagem. Para contornar esse problema, foi implementado uma maneira de diminuir a dimensão da matriz: Procura-se, na imagem, o valor mínimo G_{min} e o valor máximo G_{max} de nível de cinza. Subtraindo de todos os pixels o valor G_m , teremos a redução da dimensão de (256×256) para $\left(G_{max} - G_{min}\right) \times \left(G_{max} - G_{min}\right)$, aumentando a performance do algoritmo e utilizando menos memória do sistema.

5.2.3 Tamanho da janela e quantidade de amostras.

O tamanho da janela para seleção de amostras e para o processamento das imagens pode ser definido entre três opções: 7×7, 5×5 ou 3×3. A escolha correta

depende do tipo de imagem que for processar. A janela deve ser proporcional ao tamanho da primitiva textural fazendo com que quanto mais finas as texturas, menor a janela e vice-versa. O sistema pode classificar até 4 texturas e para cada uma, devem ser selecionadas quatro amostras representativas.

5.2.4 Forma de escalonamento utilizada

Neste trabalho, optou-se por fazer o escalonamento linear pelos valores *min* e *max* relativos. Essa opção apresentou melhores resultados do que o escalonamento linear pelos valores *min* e *max* absolutos. O escalonamento pela função tangente hiperbólica foi apresentada como uma alternativa para ser utilizado em situações particulares descritas no item 4.4.3, não sendo genérico.

5.3 Análise dos resultados

As imagens originais foram processadas gerando como saída uma imagem classificada. Nesse trabalho, foram selecionadas 6 imagens características para serem mostradas. Cada uma dessas 6 imagens representa um exemplo característico de um conjunto de imagens com propriedades semelhantes que também obtiveram resultados semelhantes na classificação. Em cima de cada figura está a imagem original, em baixo á esquerda a imagem classificada pela abordagem-A e em baixo à direita, a imagem classificada pela abordagem-B. As amostras selecionadas pelo usuário podem ser identificadas pelos pequenos retângulos pretos com um ponto central. Para cada textura identificada foi atribuída uma cor diferente (azul, verde, ciano, roxo) para diferenciá-las. Quando a rede não conseguiu classificar em nenhum desses quatro casos, então foi atribuída a cor vermelha. Em cada imagem, é feita uma análise dos possíveis fatores que influenciaram na qualidade dos resultados.

Obs: Na impressão desse trabalho, as cores azul, verde, ciano, roxo e vermelho são substituídas respectivamente por cinza escuro, cinza médio escuro, cinza médio claro, cinza claro e branco.

Imagem 1

A imagem 1 possui 4 texturas diferentes as quais possuem amostras com intervalos de valores da média bem separados. O mesmo não acontece com os valores das amostras do desvio-padrão e da uniformidade que estão sobrepostos. A FIGURA 5.3 ilustra essa distribuição de valores após o escalonamento para o intervalo [−1;+1]. A janela para seleção de amostras tem dimensão 7×7. Os percentuais de *pixels* corretamente classificados nas FIGURA 5.4b) e c) são 86,3% e 95,1%, respectivamente.

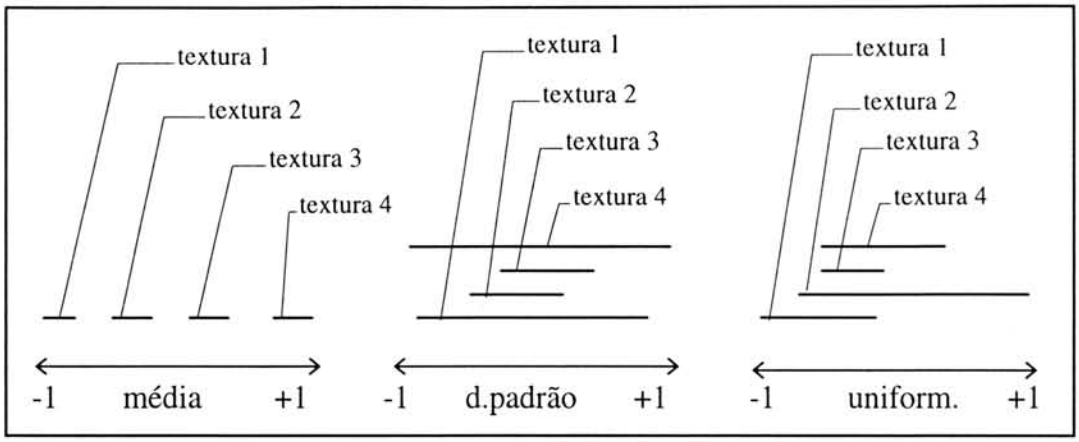

FIGURA 5.3-Distribuição dos parâmetros da imagem 1.

A FIGURA 5.4 mostra os resultados obtidos com a classificação da imagem 1.

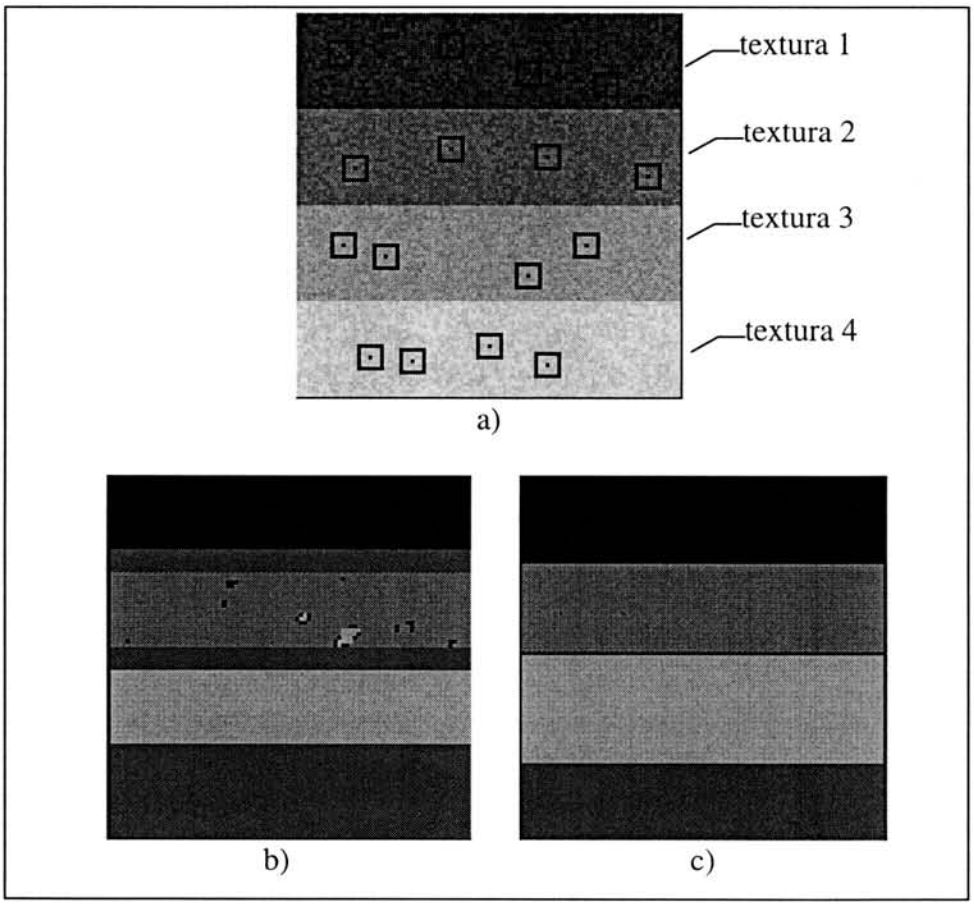

FIGURA 5.4-Classificação da imagem 1.

O principal problema encontrado na abordagem-A se dá exatamente quando a janela 7×7 está na transição de dois padrões diferentes (FIGURA 5.4b). O ponto central da janela pertence à região que se quer classificar, mas a área da janela abrange duas regiões distintas. Em algumas destas situações, a região é classificada incorretamente. A explicação está no fato de que ao calcular os parâmetros $(\overline{m}, s, h)$ desta janela, pode haver uma coincidência de valores com alguma outra classe presente na imagem, confundindo a classificação. Nesse caso específico, para amenizar este problema, a

solução pode estar em diminuir o tamanho da janela. Outra solução seria criar uma janela de tamanho variável. Na transição entre texturas, o tamanho da janela deve ser diminuído dinamicamente para que não englobe duas texturas diferentes. Quando a janela estiver totalmente dentro de uma classe, então seu tamanho pode retornar ao original. Na classificação pela abordagem-B (FIGURA 5.4c), o resultado é visivelmente melhor, mas também apresenta o mesmo problema na transição entre classes.

Imagem 2

A imagem 2 possui as mesmas características da imagem 1, porém neste caso, o tamanho da janela utilizada foi 3×3, para que se possa comparar o rendimento com a imagem 1 na transição entre texturas. A FIGURA 5.5 apresenta os resultados. Os percentuais de *pixels* corretamente classificados nas FIGURA 5.5b) e c) são 80,1% e 78,6%, respectivamente.

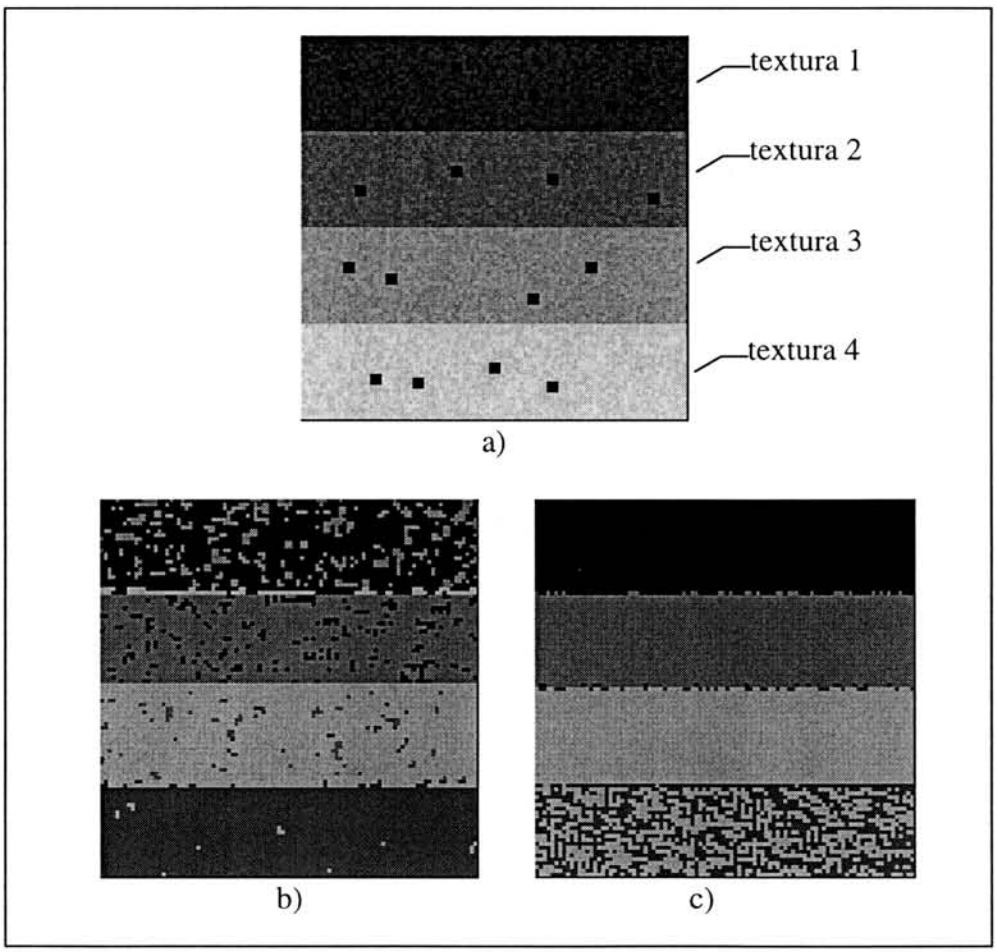

FIGURA 5.5-Classificação da imagem 2.

Utilizando a janela menor, obtém-se uma precisão maior na classificação quando a janela encontra-se na transição entre duas classes. Entretanto, ganha-se novo problema: no interior da textura, a classificação teve resultado pior. Na classificação pela abordagem-A (FIGURA 5.5b), as texturas 1 e 2 são confundidas uma com a outra em várias ocasiões. Na classificação pela abordagem-B (FIGURA 5.5c), a textura 4 é confundida com a textura 3. A explicação para essa degradação, pode residir no fato de

que janelas menores se tornam mais vulneráveis a flutuações nos valores dos *pixels* ou pelo fato da janela 3×3 não ser grande o suficiente para captar a granularidade da textura. Essa situação pode ser exemplificada da seguinte forma: Suponha uma janela 7×7 com todos seus 49 *pixels* com valor 10. Essa janela terá média com valor igual à 10, pois $49 \times 10 \div 49 = 10$. Se por acaso, dois desses *pixels* tiverem valor 50, então a média aumenta para $\frac{47 \times 10 + 2 \times 50}{49} = 11{,}632$. Entretanto, se a janela for 3×3 e repetirmos a situação anterior, a média dá um salto muito maior, indo de 10 para $\frac{7 \times 10 + 2 \times 50}{9} = 18{,}888$. Para o desvio-padrão e a uniformidade, essa instabilidade é ainda maior (vide TABELA 5.1).

TABELA 5.1-Comparação dos parâmetros entre janelas de dimensões diferentes.

	Média	D.Padrão	Unifor.
janela 7×7 com todos pixels com valor 10	10,0	0,0	1,0
janela 7×7 com 2 pixels com valor 50 e os demais com valor 10	11,632	7,996	0,913
janela 3×3 com todos pixels com valor 10	10,0	0,0	1,0
janela 3×3 com 2 pixels com valor 50 e os demais com valor 10	18,888	17,638	0,47

Imagem 3

Nesta imagem, as 4 texturas possuem intervalos de valores do desvio-padrão bem separados enquanto a média e uniformidade ficam sobrepostos após o escalonamento para o intervalo [−1;+1], conforme ilustra FIGURA 5.6. A janela de seleção de amostras tem dimensão 7×7.

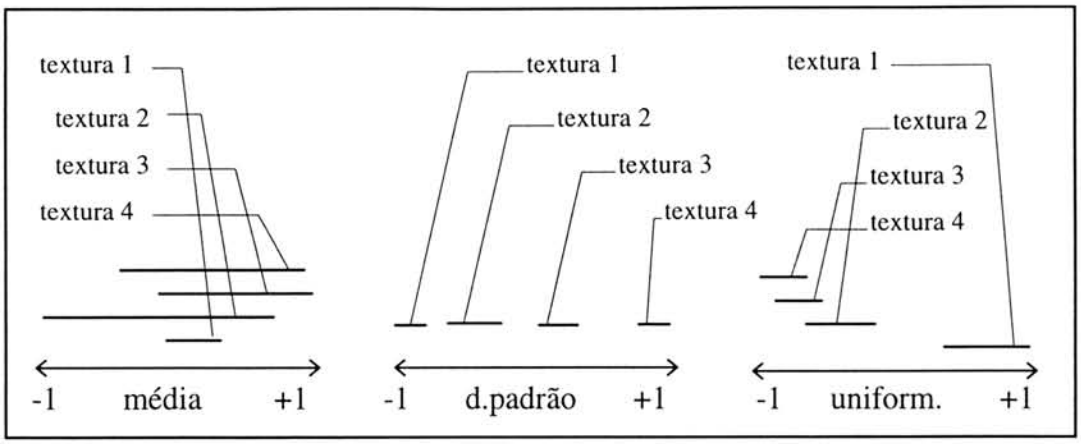

FIGURA 5.6-Distribuição dos parâmetros da imagem 3.

A FIGURA 5.7 apresenta os resultados obtidos com a classificação. Os percentuais de *pixels* corretamente classificados nas FIGURA 5.7b) e c) são 89,5% e 38,4%, respectivamente.

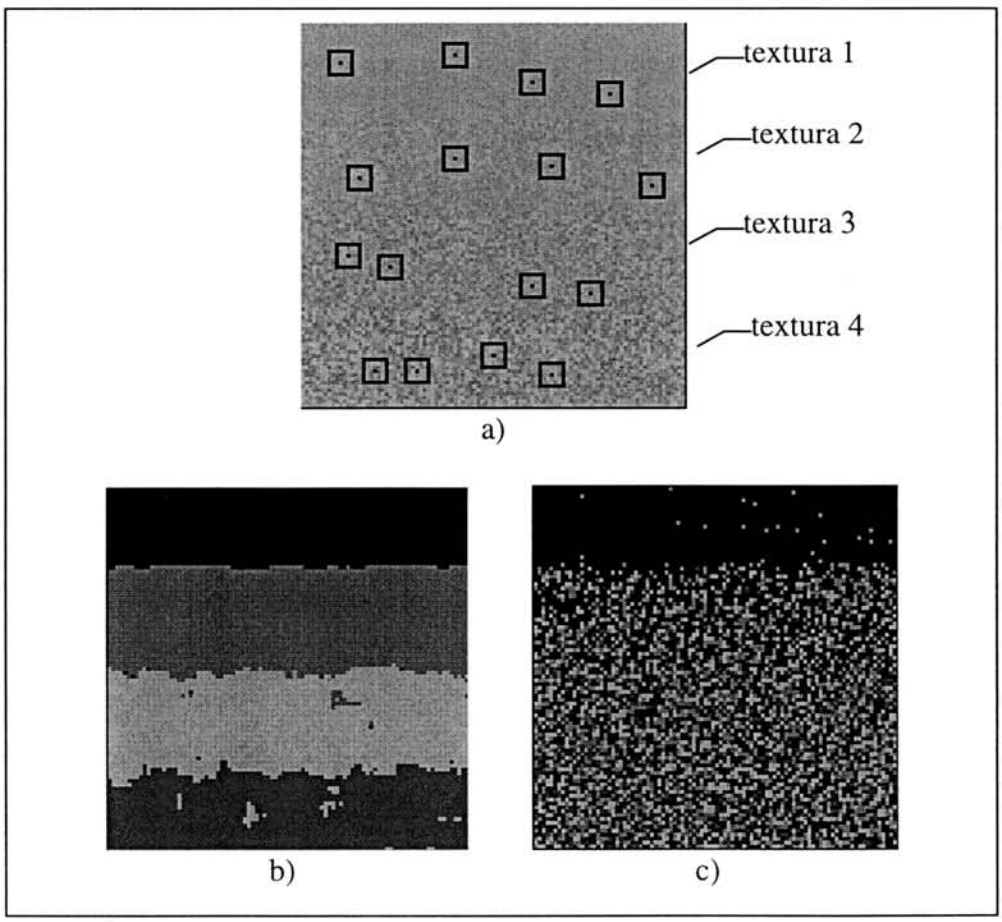

FIGURA 5.7-Classificação da imagem 3.

Na abordagem-A, quando há transição de uma textura para outra, a classificação ora aponta para a primeira ora para a segunda textura (FIGURA 5.7b), mas sem o erro ocorrido nas imagens 1 e 2, onde, na transição, a classificação aponta para uma terceira classe. Na imagem processada pela abordagem-B (FIGURA 5.7c), não foi atingido o

objetivo (a imagem de saída apresenta um efeito "degradée). Isso pode significar que a rede neural, com as configurações usadas e com as amostras escolhidas, não foi capaz de extrair o parâmetro desvio-padrão. Para tentar reverter a situação, foram feitos testes com outras configurações da rede neural, aumentando-se o número de neurônios da camada oculta e criando-se uma segunda camada oculta. Mesmo assim, os resultados não mudaram.

Imagem 4

Nesta imagem, o parâmetro que vai diferenciar as 4 texturas é a uniformidade, pois os valores da uniformidade das amostras de cada classe não se sobrepõem. Ao contrário, a média e o desvio-padrão possuem intervalo de valores que se sobrepõem após o escalonamento para o intervalo [−1;+1], conforme mostra a FIGURA 5.8. A janela de seleção de amostras tem dimensão 7×7. Essa imagem também demonstra as conseqüências do uso de uma janela pequena em relação ao tamanho da primitiva de textura.

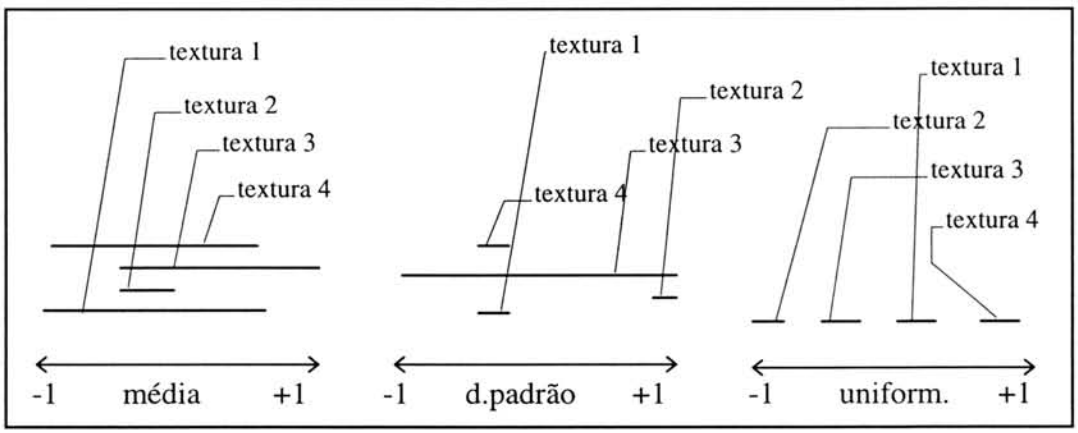

FIGURA 5.8-Distribuição dos parâmetros da imagem 4.

A FIGURA 5.9 mostra o resultado obtido com a classificação da imagem 4. Os percentuais de *pixels* corretamente classificados nas FIGURA 5.9b) e c) são 85,3% e 71,2%, respectivamente.

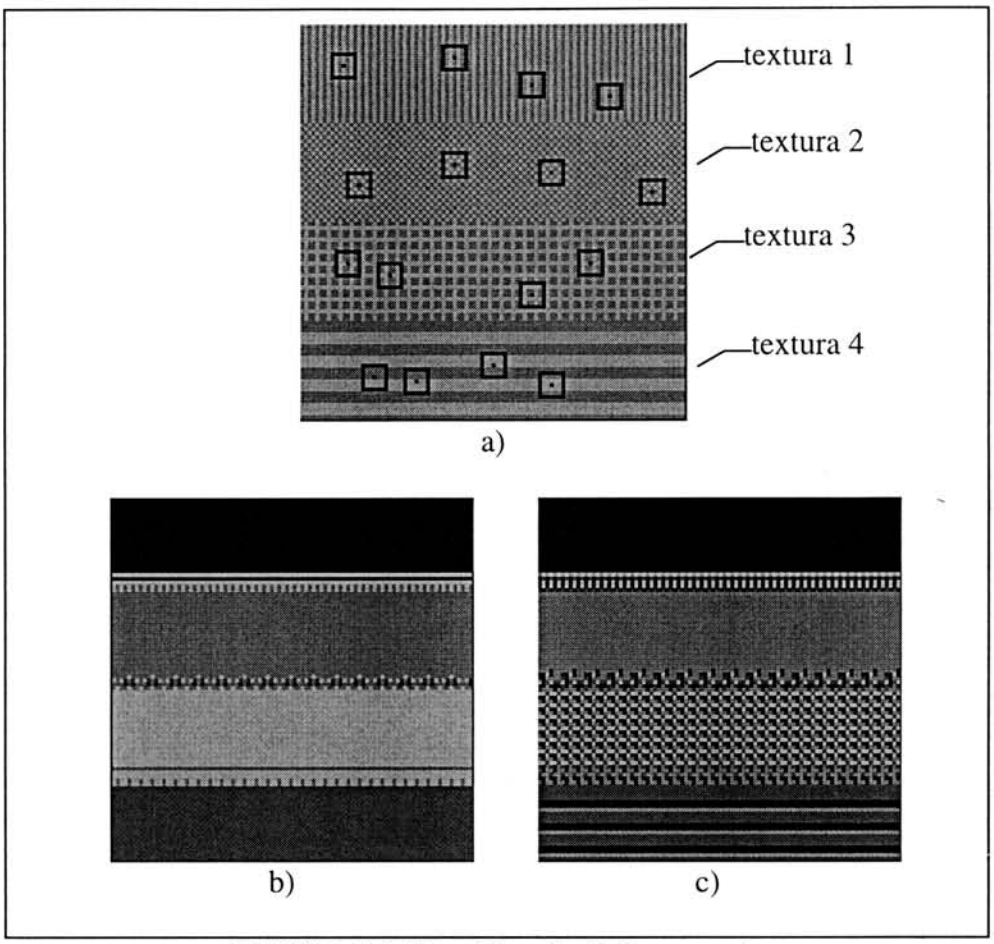

FIGURA 5.9-Classificação da imagem 4.

Como neste caso, temos 4 texturas com a média e o desvio-padrão parecidos, o que as diferencia é a distribuição espacial dos pixels, ou seja, as 4 texturas tem pixels com nível de cinza de valor muito próximos porém a disposição deles é diferente. Novamente, nesse exemplo, a classificação pela abordagem-A é bem sucedida, mostrando que sempre que tivermos um parâmetro (média, desvio-padrão ou homogeneidade) com intervalo de valores separados, então o sistema atinge seu objetivo. Na abordagem-B, ocorre um problema na classificação das duas texturas de baixo. A provável causa pode ser explicada pelos mesmos motivos apontados no caso da imagem 3.

Imagem 5

A distribuição dos valores dos parâmetros da imagem 5 é semelhante à imagem 1. A diferença é quanto ao formato das primitivas texturais que nesse caso tem forma retangular. A classe de cima repete-se no círculo de baixo e a classe de baixo repete-se no círculo de cima. A FIGURA 5.10 apresenta os resultados obtidos. A janela de seleção de amostras tem dimensão 7×7. Os percentuais de *pixels* corretamente classificados nas FIGURA 5.10b) e c) são 69,9% e 70,5%, respectivamente.

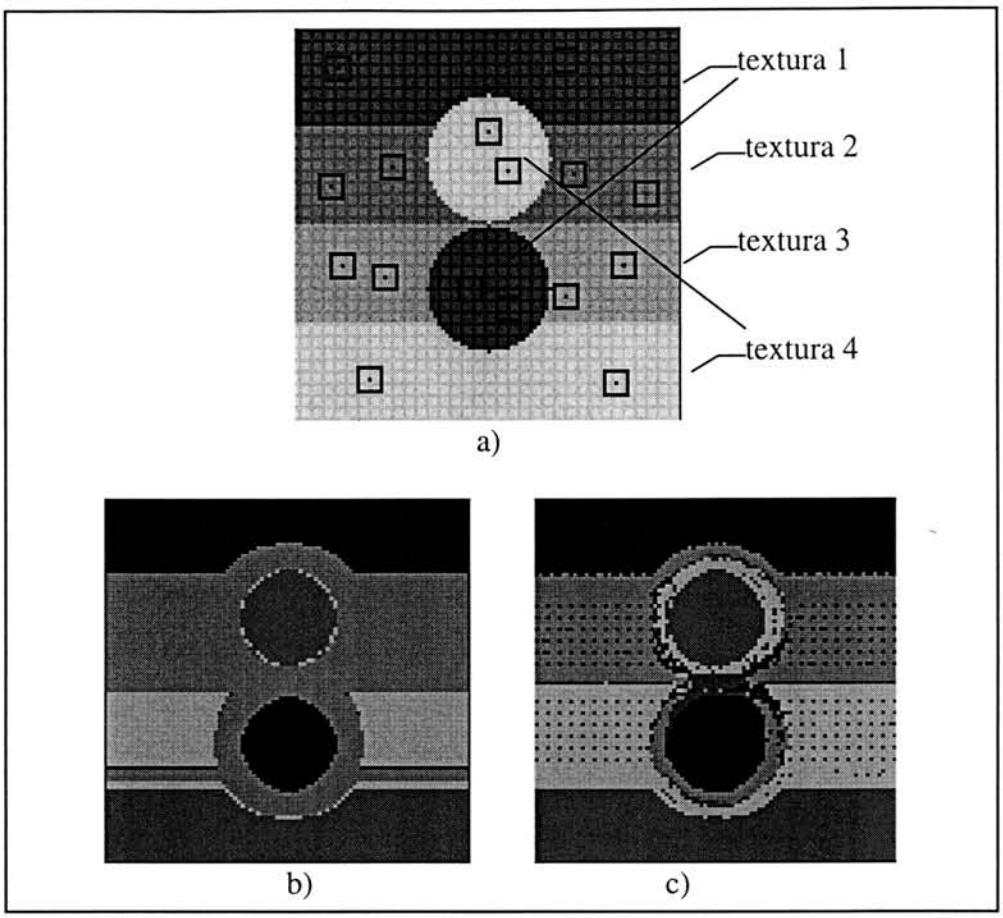

FIGURA 5.10-Classificação da imagem 5.

O resultado é semelhante à da figura 1, onde novamente o problema encontrado foi na transição entre classes.

Imagem 6

A imagem 6 têm primitivas texturais semelhantes a imagem 5. Nessa, porém, não existe um parâmetro cujos valores estejam separados. Em todos eles, existe sobreposição na distribuição dos valores, conforme mostra a FIGURA 5.11. A sobreposição dos valores visa aproximar as imagens sintéticas das imagens captadas do mundo real, onde os parâmetros se sobrepõem, dificultando a classificação. Quanto maior a sobreposição, maior a dificuldade de classificar. Além disso, visa testar a capacidade de generalização da rede neural e do sistema IMASEG como um todo quando a imagem é mais complexa. A FIGURA 5.12 mostra os resultados obtidos. Os percentuais de *pixels* corretamente classificados nas FIGURA 5.12b) e c) são 73,0% e 53,3%, respectivamente.

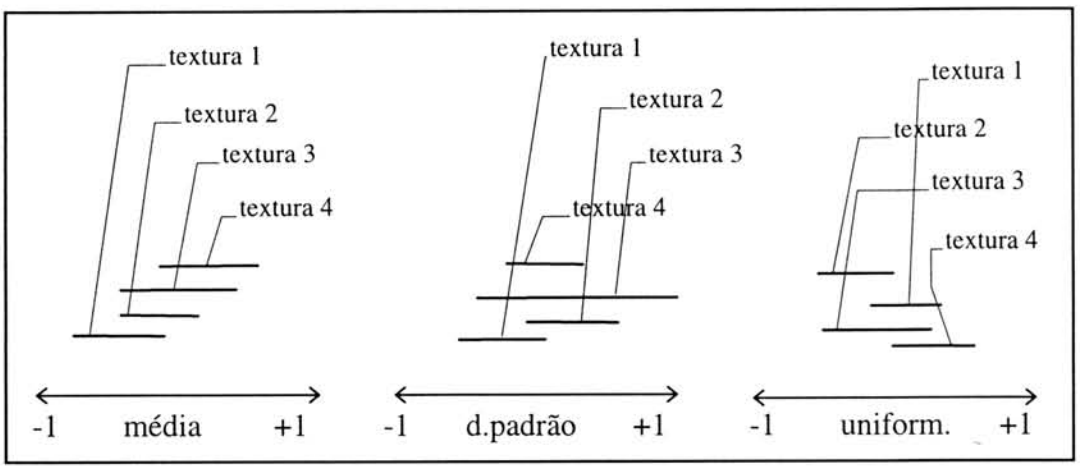

FIGURA 5.11-Distribuição dos parâmetros da imagem 6.

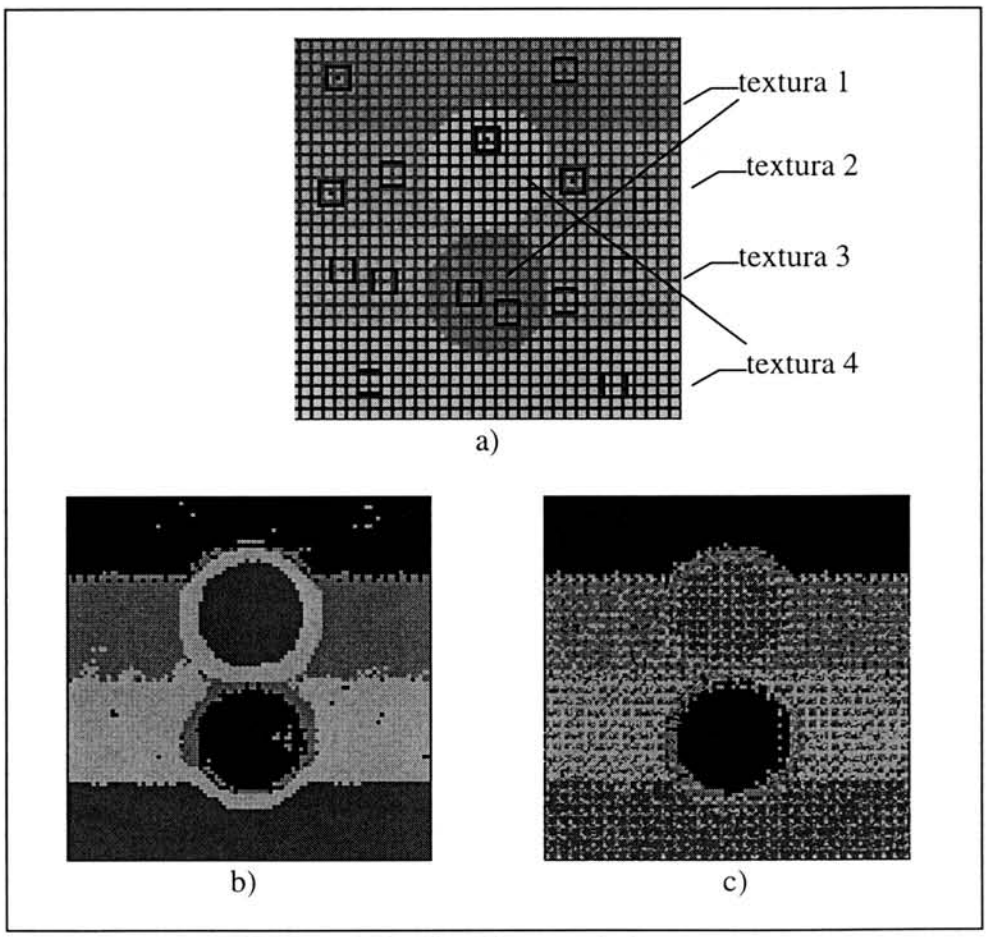

FIGURA 5.12-Classificação da imagem 6.

Como era esperado, a classificação sofreu uma degradação devido ao fato da sobreposição dos valores dos parâmetros. Mas, pode-se dizer que ainda houve relativo sucesso.

5.4 Variáveis que influenciam no resultado

Tamanho da janela:

A determinação do tamanho da janela é fundamental para o sucesso do sistema. A janela deve ser grande o suficiente para captar todas as feições de texturas grossas e pequena o suficiente para não englobar mais de uma textura dentro de sua área. A transição entre duas classes também é afetada pelo tamanho da janela.

Quantidade e qualidade das amostras:

O usuário tem que selecionar 4 amostras que sejam realmente representativas daquela textura. Se houver variação na aparência da mesma textura, essas variações tem que ser expressas pelas amostras. Quanto mais amostras forem selecionadas, mais preciso será o treinamento da rede neural.

6. Conclusões e perspectivas futuras

Este trabalho teve por objetivo o estudo da classificação de imagens através da textura usando redes neurais. Para isso, foi proposto, implementado e testado um sistema para classificação de imagens digitais utilizando a rede BPN e atributos espaciais. Quando se fala em textura, deve-se ter em mente a grande variedade de formas existentes no mundo real ou criadas sinteticamente (granularidade, resolução, número de cores, iluminação, contraste, etc). Qualquer sistema de processamento de imagens deve se restringir à um determinado tipo de imagem pois dificilmente será genérico. Dentro desse contexto, foram criadas e testadas várias imagens sintéticas onde procurou-se abranger diversas complexidades de textura e variações nos valores dos parâmetros utilizados para quantificar a textura (média, desvio-padrão e uniformidade).

O sistema IMASEG, desenvolvido nesse trabalho, possui duas abordagens para realizar o mapeamento para a rede neural. Através dos exemplos descritos no capítulo 5 e em diversas outras imagens, mostrou-se que a classificação pela primeira abordagem obteve resultados satisfatórios para todas as imagens testadas, mostrando que é um sistema bastante genérico e robusto. O seu sucesso está diretamente associado à distribuição dos parâmetros de entrada da rede neural. Quanto mais sobrepostos estiverem esses valores, maior a dificuldade de distinguir as classes presentes na imagem. A segunda abordagem obteve bons resultados em situações em que a média estava bem distribuída. No caso da imagens 3 e 4, onde os resultados foram ruins, constatou-se que a extração do desvio-padrão e uniformidade pela própria rede não foi possível. Isso pode indicar que a rede neural utilizada consegue inferir a média dos níveis de cinza das janelas, mas, com a configuração utilizada, foi incapaz de abstrair informações espaciais mais complexas.

A textura, atributo espacial da imagem, possui diversas feições que podem ser extraídas da matriz de coocorrência dos níveis de cinza [HAR73]. No decorrer do trabalho, foi feito um estudo sobre o comportamento dessas feições e constatou-se a relação entre algumas delas, especialmente entre desvio-padrão e contraste os quais são diretamente proporcionais e entre uniformidade e entropia as quais são inversamente proporcionais. Diversas outras feições propostas por Haralick são passíveis de investigação a respeito de suas relações.

A rede neural MLP com algoritmo Backpropagation é um excelente classificador quando as suas entradas são escolhidas corretamente e estão bem distribuídas. Entretanto, uma das tarefas que consumiu mais tempo nesse trabalho, foi justamente analisar quais parâmetros deveriam ser utilizados como entrada, qual melhor forma de escaloná-los para a rede e como isso influencia no comportamento da mesma.

A escolha correta da dimensão da janela para selecionar as amostras características de cada classe é de fundamental importância para o sucesso do sistema. O tamanho da janela deve ser proporcional ao tamanho das primitivas texturais.

Conclui-se que o uso de atributos espaciais juntamente com a rede neural para a classificação de imagens mostrou-se eficaz, abrindo espaço para a análise de sua possível generalização e utilidade para outras situações.

Seria interessante, como trabalho futuro, um estudo comparativo quantitativo entre diversos métodos para classificação de imagens e aplicação com imagens reais.

Anexo A-1 Código fonte (principais funções)

Neste capítulo, são listados os trechos do código fonte de maior importância utilizados na construção do sistema. Basicamente, esses trechos são o cálculo dos parâmetro de entrada da Rede Neural (média, desvio-padrão, matriz de coocorrência e uniformidade) e os algoritmos de propagação e adaptação do Backpropagation utilizados na rede MLP.

Cálculo dos parâmetros de entrada e matriz de coocorrência:

```
#define ROWS 100
#define COLS 100
#define JANMAX 7
#define LEVELS 256
#define PADROES 16
#define INPUT 4
#define HIDEN 10
#define OUTPUT 3
int janela[JANMAX][JANMAX];
unsigned char image[ROWS][COLS];          // Recebe inteiros apesar de ser char
unsigned char aCoocor1[LEVELS][LEVELS]; // Recebe inteiros apesar de ser char
unsigned char aCoocor2[LEVELS][LEVELS]; // Recebe inteiros apesar de ser char
int aAmostra[PADROES][2]; // coordenadas das amostras selecionadas pelo usuario
float aEntrada[PADROES][INPUT+1];         // matriz com as entradas
float aIpj[HIDEN+1];          // vetor de saida da camada oculta
float aOpk[OUTPUT];           // vetor de saida da camada de saida
float aPesoj[HIDEN][INPUT+1];   // matriz de pesos da camada de entrada
float aPesok[OUTPUT][HIDEN+1];  // matriz de pesos da camada oculta
int  aDesej[PADROES][OUTPUT];  // matriz dos valores desejados
int nIteracoes;
float nMi = 0.02;         // usado no treinamento do BP
float nLimiar;      // usado na segmentacao
int nEntradas = 3;  // nro. de entradas(5+1 Abordagem A ou 49+1 Abordagem B)
int nPadroes = 16;  // nro. de padroes/amostras.
int nSaidas = 3;      // nro. de saidas do BP
int nEscondidos;      // nro. de neuronios na camada escondida do BP
float nMinM, nMaxM, nMinV, nMaxV, nMinH, nMaxH, nMinE, nMaxE, nMinK, nMaxK;
float nMinLevel, nMaxLevel;
//---------------------------------------------------------------------------
float CalcMedia(int nDimJan){
int i,j, nTotCinza=0;
float nMedia;
for(j=0; j< nDimJan; j++)
for(i=0; i< nDimJan; i++)
                nTotCinza += janela[i][j];
nMedia = nTotCinza / pow(nDimJan,2);
return nMedia;
}
//---------------------------------------------------------------------------
float CalcVariancia( float nMedia, int nDimJan ){
int i,j;
float nDif=0.0, nVariancia;
for(j=0; j< nDimJan; j++)
        for(i=0; i< nDimJan; i++)
                nDif += pow( janela[i][j] - nMedia, 2 );
nVariancia = nDif / pow(nDimJan,2);
```

```
return nVariancia;
}
//---------------------------------------------------------------------------
void CalcCoocor( int *nDimensao, int *nSomatorio, int nOrientacoes, int nDimJan ){
// Parametros:
// nDimensao: Eh a dimensao da mat. de coocor. apos sua reducao.
// nSomatorio: Eh o fator de normalizacao. (Eh igual ao somatorio da matriz Cs).
// nOrientacoes: 1-Horiz, 2-Vert, 3-Usar Horiz. e Vert., 4-Usar as 4 direcoes.
// Observacoes:
// 1)Faz reducao do tamanho da matriz de coocorrencia.
// 2)Este algoritmo trabalha com mat. de coocor. normalizada. Para gerar a matriz normalizada(Cn),
//   soma-se todas as matrizes individuais, gerando a matriz Cs e divide-se cada elemento pelo fator de
// normalizacao que eh o somatorio da matriz Cs. Entretanto essa divisao geraria uma matriz FLOAT!
// 3)Na pratica, a matriz Cs serah a soma de duas matrizes C1+C2. A mat. C1 tem a soma das
//   matrizes Cv e Ch, e a matriz C2 tem a soma das matrizes Cdd e Cde.
// 4)Quando quisermos ler um elemento de Cn, soma-se os respectivos elementos
//  de C1+C2 e divide-se pelo fator de normalizacao em tempo de execucao.
//  Desta forma, temos duas matrizes CHAR (C1 e C2) e nenhuma matriz FLOAT.
int i,j;
int nMaxLevel=0, nMinLevel=LEVELS+1, nSomaCoocor=0;
double nSomaNormalizada=0.0, nElemento;
// Procura pelos valores minimo e maximo de cinza da janela
for(j=0; j<nDimJan; j++)
        for(i=0; i<nDimJan; i++){
                nMinLevel = (janela[i][j] < nMinLevel) ? janela[i][j] : nMinLevel;
                nMaxLevel = (janela[i][j] > nMaxLevel) ? janela[i][j] : nMaxLevel;
        }
*nDimensao = nMaxLevel - nMinLevel + 1;
// Inicia matriz de coocorrencia reduzida com zero.
for(j=0; j<*nDimensao; j++)
        for(i=0; i<*nDimensao; i++){
                aCoocor1[i][j] = 0;
                aCoocor2[i][j] = 0;
        }
for(j=0; j<nDimJan; j++)
        for(i=0; i<nDimJan; i++){
                if(i-1 > -1)        // Orientacao: Horizontal. A esquerda
                        aCoocor1[janela[i][j]-nMinLevel][janela[i-1][j]-nMinLevel] ++;
                if(i+1 < nDimJan)           // Orientacao: Horizontal. A direita
                        aCoocor1[janela[i][j]-nMinLevel][janela[i+1][j]-nMinLevel] ++;
                if(i-1 > -1)        // Orientacao: Vertical- A cima
                        aCoocor1[janela[j][i]-nMinLevel][janela[j][i-1]-nMinLevel] ++;
                if(i+1 < nDimJan) // Orientacao: Vertical- A baixo
                        aCoocor1[janela[j][i]-nMinLevel][janela[j][i+1]-nMinLevel] ++;
                if((i-1 > -1) && (j+1 < nDimJan))  // Orientacao: 45 - A cima
                        aCoocor2[janela[j][i]-nMinLevel][janela[j+1][i-1]-nMinLevel] ++;
                if((i+1 < nDimJan) && (j-1 > -1)) // Orientacao: 45 - A baixo
                        aCoocor2[janela[j][i]-nMinLevel][janela[j-1][i+1]-nMinLevel] ++;
                if((i+1 < nDimJan) && (j+1 < nDimJan)) // Orientacao: 135 - A cima
                        aCoocor2[janela[j][i]-nMinLevel][janela[j+1][i+1]-nMinLevel] ++;
                if((i-1 > -1) && (j-1 > -1))                    // Orientacao: 135 - A baixo
                        aCoocor2[janela[j][i]-nMinLevel][janela[j-1][i-1]-nMinLevel] ++;
        }
        break;
}
// Confere a Mat.de coocor. e mostra o somatorio.Pode ser eliminada mais tarde.
for(j=0; j<*nDimensao; j++)
        for(i=0; i<*nDimensao; i++){
                nSomaCoocor += aCoocor1[i][j] + aCoocor2[i][j];
```

```
                if( aCoocor1[i][j] != aCoocor1[j][i] )
                        gAviso(100,460,"Matriz de coocorrencia1 nao eh simetrica",RED);
                if( aCoocor2[i][j] != aCoocor2[j][i] )
                        gAviso(100,460,"Matriz de coocorrencia2 nao eh simetrica",RED);
        }
*nSomatorio = 2*2*(nDimJan-1)*nDimJan;
*nSomatorio += 2*2*(nDimJan-1)*(nDimJan-1);
}
if(nSomaCoocor != *nSomatorio )
        gAviso(100,460,"Erro na soma da matriz coocorrencia",RED);
for(j=0; j<*nDimensao; j++)
        for(i=0; i<*nDimensao; i++){
                nElemento = (double)(aCoocor1[j][i]+aCoocor2[j][i]) / *nSomatorio;
                nSomaNormalizada += nElemento;
        }
if((nSomaNormalizada < 0.99999 )||(nSomaNormalizada > 1.00001 ))
        gAviso(100,460,"Erro na soma da matriz normalizada",RED);
}
//---------------------------------------------------------------------------
float CalcUniformidade( int nDimensao, int nNormalizacao ){
int i,j;
double nHomog=0.0;
double nElemNormalizado, aux,nElemento;
for(j=0; j<nDimensao; j++)
        for(i=0; i<nDimensao; i++)
                if((aCoocor1[i][j]+aCoocor2[i][j]) != 0)
                    nHomog += (pow((double)(aCoocor1[i][j]+aCoocor2[i][j]) / nNormalizacao, 2));
return nHomog;
}
//---------------------------------------------------------------------------
float CalcEntropia( int nDimensao, int nNormalizacao ){
int i,j;
float nEntropia=0.0;
float Aux,Aux1;
for(j=0; j<nDimensao; j++)
        for(i=0; i<nDimensao; i++){
                if(aCoocor1[i][j]+aCoocor2[i][j] != 0){
                        Aux1 = (float)(aCoocor1[i][j]+aCoocor2[i][j]) / nNormalizacao;
                        Aux = Aux1 * log10(Aux1);
                        if( Aux > 0.0 )
                                gAviso(10,460,"Erro no Calculo da entropia!",RED);
                        nEntropia += Aux;
                }
        }
return -nEntropia;
}
//---------------------------------------------------------------------------
float CalcContraste( int nDimensao, int nNormalizacao ){
int i,j;
float nContraste=0.0;
for(j=0; j<nDimensao; j++)
        for(i=0; i<nDimensao; i++)
                        nContraste  +=  pow((j-i),2)  *  (double)(aCoocor1[i][j]+aCoocor2[i][j])  /
nNormalizacao;
return nContraste;
}
//---------------------------------------------------------------------------
float Escalona( int nTipo, float nValor, float nMin, float nMax ){
float nRetorno;
```

```
if(nMin == nMax) //nMin nao pode ser igual a nMax devido a divisao por 0!.
        nRetorno = 0.0;
else{
switch( nTipo ){
        case 0:  nRetorno =  255.0 * (nValor - nMin ) / (nMax - nMin) ;
                 break;  // Escalona entre 0 e 255
        case 1:  nRetorno =  1.0 * (nValor - nMin ) / (nMax - nMin) ;
                 break;  // Escalona entre 0 e 1
        case 2:  nRetorno =  (2.0 * (nValor - nMin ) / (nMax - nMin) ) - 1.0;
        break; // Escalona entre  -1 e +1
}
}
return(nRetorno);}

//---------------------------------------------------------------------------
// Algorítmo BackPropagaton
//---------------------------------------------------------------------------
void imaseg31(){
int f,i,j,lCancel=0;

sprintf(str,"Neuronios da camada H (sugestao->4):");
gPrintf(10,10,str);
nEscondidos = gReadInt( 300, 10, BLACK, WHITE,3 );
sprintf(str,"Valor de Mi (Sugestao->0.02):");
gPrintf(10,20,str);
nMi = gReadFloat( 260, 20, BLACK, WHITE, 6 );
sprintf(str,"Nro. de iteracoes:");
gPrintf(10,30,str);
nIteracoes = gReadInt( 170, 30, BLACK, WHITE, 6 );
//////////  Inicializacao dos vetores   ////////////
InitInput();
IniciaDesejados();
InitWeight();
///////  Mostra o erro medio quadrado inicial  /////////
sprintf(str,"Erro inicial: %+f", ErroMedio());
gPrintf(420,110,str);
gAviso(10,460,"Tecle Enter para iniciar treinamento",BLUE);
///////////  Treinamento  ////////////////////////////
for( f = 0; f < nIteracoes; f++ ){
        for( i = 0; i < nPadroes; i++ ){
                Propagation(i);
                Adaptation(i);
        }
        if(fmod(f,(nIteracoes/10))==0){
                sprintf(str,"Iteracao:%d", f+1);
                gPrintf(420,120,str);
                sprintf(str,"Erro:%+f", ErroMedio());
                gPrintf(420,130,str);
        }
        if(kbhit()){
                lCancel=1;
                break;
        }
}
if(lCancel==1)
        gAviso(10,460,"Treinamento cancelado",RED);
else{
        sprintf(str,"Erro:%+f", ErroMedio());
        gPrintf(420,130,str);
```

```
        DispWeight();
        MostraTabela(); // mostra tabela com as entradas, vlr.desejados e saida
        sprintf(str,"Rede Treinada para %d padroes", nPadroes);
        gPrintf(10,450,str);
        gAviso(10,460,"Tecle Enter para voltar",BLUE);
}}
//---------------------------------------------------------------------------
static void InitInput() {
// Entrada[p][0]=1.0, Entrada[p][1]=Media,  Entrada[p][2]=Homog.
// Entrada[p][3]=Entropia, Entrada[p][4]=Contraste
int f,i,j;
int nLeft, nTop, nRight, nBottom;
int nDimensao, nNormalizacao;
float nMedia=-1.0, nVariancia=-1.0, nHomog=-1.0, nEntropia=-1.0, nContraste=-1.0;
for(f=0; f<nPadroes; f++){
        CalcJanela(aAmostra[f][0],aAmostra[f][1], &nLeft, &nTop, &nRight, &nBottom);
        for(i=0; i<nDimJan; i++)
                for(j=0; j<nDimJan; j++)
                        janela[j][i] = image[nLeft+j][nTop+i];
        nMedia = CalcMedia();
        if(nEntradas > 1)
                nVariancia = CalcVariancia( nMedia );
        CalcCoocor( &nDimensao,&nNormalizacao,ORIENTACAO);
        if(nEntradas > 2)
                nHomog = CalcUniformidade( nDimensao, nNormalizacao );
        if(nEntradas > 3)
                nEntropia = CalcEntropia( nDimensao, nNormalizacao );
        if(nEntradas > 4)
                nContraste = CalcContraste( nDimensao, nNormalizacao );
// Escalona as entradas
        aEntrada[f][0] = 1.0;
        aEntrada[f][1] = Escalona(2,nMedia,nMinM,nMaxM);
        aEntrada[f][2] = Escalona(2,nVariancia,nMinV,nMaxV);
        aEntrada[f][3] = Escalona(2,nHomog,nMinH,nMaxH);
        aEntrada[f][4] = Escalona(2,nEntropia,nMinE,nMaxE);
        aEntrada[f][5] = Escalona(2,nContraste,nMinK,nMaxK);
}}
//---------------------------------------------------------------------------
void InitWeight(){
int s,f;
// aleatorios entre -1.00 e +1.00
for(s=0; s < nEscondidos; s++)
        for(f=0; f < nEntradas + 1; f++)
                aPesoj[s][f] = random(200)/100.0 - 1.0;
for(s=0; s < nSaidas; s++)
        for(f=0; f < nEscondidos + 1; f++)
                aPesok[s][f] = random(200)/100.0 - 1.0;
}
//---------------------------------------------------------------------------
void IniciaDesejados(){
int aDesej[16][3] =  {{1, 1, 1},{1, 1, 1},{1, 1, 1},{1, 1, 1},{1, 1,-1},{1, 1,-1},{1, 1,-1},{1, 1,-1},
                      {1,-1, 1},{1,-1, 1},{1,-1, 1},{1,-1, 1},{1,-1,-1},{1,-1,-1},{1,-1,-1}, {1,-1,-1}};
}
//---------------------------------------------------------------------------
void Propagation( int i ){
int f,j;
for( f = 0; f < nEscondidos + 1; f++ )
        aIpj[f] = 0.0;
for( j = 0; j < nSaidas; j++ )
```

```
        aOpk[j] = 0.0;
// Camada j
for( f = 0; f < nEscondidos; f++ ){
        for( j = 0; j < nEntradas + 1; j++ )
                aIpj[f] += aEntrada[i][j] * aPesoj[f][j];
        aIpj[f] = tanh( aIpj[f] );
}
aIpj[nEscondidos] = 1.0;  // A ultima entrada eh sempre fixa em 1.
// Camada k
for( j = 0; j < nSaidas; j++ ){
        for( f = 0; f < nEscondidos + 1; f++ )
                aOpk[j] += aIpj[f] * aPesok[j][f];
        aOpk[j] = tanh( aOpk[j] );
}}
//---------------------------------------------------------------------------
void Adaptation( int n ){
int j,i;
float erro_j[HIDEN+1],aux;
float erro_k[OUTPUT];
for(i=0;i<nSaidas;i++){
        erro_k[i] = (aDesej[n][i]-aOpk[i]) * (1 - pow(aOpk[i],2));
}
for(i=0;i<nEscondidos;i++){
        aux=0;
        for(j=0;j<nSaidas;j++){
                aux += erro_k[j] * aPesok[j][i];
        }
        erro_j[i] = aux * (1 - pow(aIpj[i],2));
}
for(i=0;i<nSaidas;i++){
        for(j=0;j<nEscondidos+1;j++){
                aPesok[i][j] += nMi * erro_k[i] * aIpj[j];
        }
}
for(i=0;i<nEscondidos;i++){
        for(j=0;j<nEntradas+1;j++){
                aPesoj[i][j] += nMi * erro_j[i] * aEntrada[n][j];
        }
}}
//---------------------------------------------------------------------------
float ErroMedio( void ){
float erro = 0.00;
int  i,j;
for( i = 0; i < nPadroes; i++ ){
        Propagation( i );
        for( j = 0; j < nSaidas; j++ ){
                erro += pow( ( aDesej[i][j] - aOpk[j] ), 2 );
        }
}
return erro/2.0;
}
```

Bibliografia

[BAR94] BARBOSA, Valmir C.; MACHADO, Ricardo J.; LIPORACE, Frederico. A Neural System for Deforestation Monitoring on Landsat Imagens of the Amazon Region. **International Journal of Approximate Reasoning**, New York, v.11, p.321-359, 1994.

[BAR69] BARTELS, P.; BAHR,G.; WEID, G. Cell recognition from line scan transition probability profiles. **Acta Cytol.**, v.13, p.210-217, 1969.

[BIS92] BISCHOF, H. SCHNEIDER, W. PINZ, J. Multispectral Classification of Landsat-Images using Neural Networks. **IEEE Transactions on geoscience and remote sensing**, New York, v. 30, n.3, May 1992.

[CAM92] CÂMARA, Gilberto. Spring: Processamento de imagens e dados georefereciados. In: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO GRÁFICA E PROCESSAMENTO DE IMAGENS, SIBGRAPI, 5., 1992, Águas de Lindóia. **Anais do SIBGRAPI V**, São Paulo: SBC/INPE, 1992. p.233-242.

[CAR83] CARPENTER, G.; GROSSBERG, S. A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine. **Computer Vision, Graphics and Image Processing**, Duluth, v.37, p.54-115,1983.

[CHI74] CHIEN, Y.P.; FU, K.S. Recognition of X-ray picture patterns. **IEEE Trans. Syst., Man, Cybern.**, New York, v.SMC-4, p.145-156, Mar. 1974.

[CHR78] CHRISTMANN, Raul Udo. **Estatística Aplicada.** São Paulo: E. Blucher, 1978.

[CLA95] CLARO, Luis Otávio Thompson. **Texturas de imagens utilizando conceitos de morfologia matemática**. Dissertação do CPERSM.

[CLE86] McCLELLAND, James; RUMERLHART, David. **Parallel Distributed Processing.** Cambridge, MA: MIT Press, 1986, v.1

[COG94] COGO, Sandra Eliza Vielmo. **Feições de textura para classificação de imagens**. Dissertação CPERSM.

[DAR68] DARLING, E.M.; JOSEPH, R.D. Pattern recognition from satellite altitudes. **IEEE Trans. Syst., Man, Cybern**, New York, v. SMC-4, p.38-47, Mar.1968.

[ENG94] ENGEL, Paulo M.; NUNES, Rodrigo. IRENE: Intelingent Processing of Multispectral Images by Self-Organizing Maps. In: INTERNATIONAL SYPOSIUM ON ARTIFICIAL NEURAL NETWORKS, ISANN, 1994, Taiwan. **Proceedings of the 194 international symposium on artificial neural networks-ISANN.** Taiwan, 1994.

[FAC93] FACON, Jaques. **Processamento e Análise de Imagens**. Córdoba: EBAI, 1993., v.6, Córdoba, 1993.

[FRE91] FREEMAN, J.A.; SKAPURA, D.M. **Neural Networks, Algorithms, Application and Programming Tecniques**. Reading: Addison-Wesley,1991.

[HAR73] HARALICK, Robert M.; SHANMUGAM, K.; ITS'HAK DINSTEIN. Textural Features for Image Classification. **IEEE Trans. on Systems, Man, and Cybernetics**, New York, v.3, n.6 , Nov. 1973.

[HAR79] HARALICK, Robert M. Statistical and Structural Approches to Texture. **Proceedings of the Ieee**, Ney York, v.67, n.5 ,May 1979.

[HEB49] HEBB, D.O. **The organization of behavior.** New York: John Wiley&Sons, 1949.

[HOP82] HOPFIELD, J.J. Neural Networks and Physical Systems with Emergent Computacional Abilities. **In: Proceeding of the National Academy of Sciences**, Washington, v.79, p.2554-2558, April 1982.

[HOU62] HOUGH,P.V.C. **Method and means for recognizing complex patterns**. U.S.Patent 3,069654, Dec. 1962.

[JUL62] JULESZ, B. Visual pattern discrimination. **IRE Trans. Inform. Theory**, v. 8, n.2, p.84-92, Feb 1962.

[KOH82] KOHONEN, Tuevo. Self-Organized formation of topologically correct feature maps. **Biological Cybernetics**,. Berlin, v.43, p.59-69,1982.

[LIP91] LIPPMANN, R. Pattern classification using neural networks. **IEE Communications Magazine**, Nov. 1989.

[MAR95] MARANA, A.N. et all. Análise de textura utilizando transformada de Hough e morfologia matemática. In: SIMPÓSIO BRASILEIRO DECOMPUTAÇÃO GRÁFICA E PROCESSAMENTO DE IMAGENS, SIBGRAPI, 8., **Anais do VIII Sibgrapi**. p.135-142, 1995.

[MAT67] MATHERON, G. **Elements pour une theorie des Milieux Poreux.** Masson, 1967.

[MCC43] McCULLOCH, W.S.; PITTS, W. A logical Calculus of the Ideas Imminent in Nervous Activity. **Bulletin of Mathematical Biophysics.** p.115-133, 1943.

[MIN69] MINSKY, M.; PAPERT, S. **Perceptons: An Introduction to Computational Geometry.** Cambridge: MIT Press, 1969.

[OSO91] OSORIO, Fernando Santos. **Um estudo sobre reconhecimento visual de caracteres através de Redes Neurais.** Porto Alegre: CPGCC, 1991. Dissertação de Mestrado.

[PAR85] PARKER, D.B. **Learning Logic.** MIT, Cambridge: MIT, 1985. (Technical Report TR-47).

[PAV82] PAVLIDIS, Theo. **Algorithms for Graphics and Image Processing**. Addson-Wesley,1991.

[PER95] PERELMUTER, Guy et all. Reconhecimento de imagens bidimensionais utilizando redes neurais artificiais. In: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO GRÁFICA E PROCESSAMENTO DE IMAGENS, SIBGRAPI, 8., **Anais do VIII Sibgrapi**, p.197-203, 1995.

[PRE76] PRESSMAN, N.J. Markovian analysis of cervical cell images. **J. Histochem. Cytochem**. vol. 24. n.1, p.138-144, 1976.

[RIC86] RICHARDS, John A. **Remote Sensing Digital Image Analysis.** Springer-Verlag.

[ROS59] ROSENBLAT, R. **Principles of Neurodynamics.** New York: Spartan Books, 1959.

[SUS95] SUSSNER, M. Segmentation and Edge-Detection of Echocardiograms using Artificial Neuronal Networks. **Proceedings of the international conference EANN.** 1995.

[TRE89] TRELEAVEN, P.; VELLASCO, M.; PACHECO, M. VLSI Architectures for Neural Networks. **IEEE Micro**, New York, v. 9, n.6, p.8-27, Dec 1989.

[WAT93] WATKINS, Christopher D.; SADUN, Alberto; MARENKA, Stephen. **Modern Image Processing**: Warping, Morphing, and Classical Techniques. Academic Press, 1993.

[WER74] WERBOS, P. **Beyond Regression:** New Tools for Prediction and Analysis in the Behavioral Sciences, Harvard, Cambridge, MA, 1974. PhD Thesis.

[WID62] WIDROW, B. Generalization and Information Storage in Networks of ADALINE Neurons. In: **Self-Organization Systems.** Washington: Spartan Books, 1962. p.435-461.

[YAU90] YAU, H-C. MANARY, M.T. Iterative improvement of Gaussian classifier. **Neural Networks**, v. 3, p.437-443, 1990.

Informática  UFRGS

CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

" Classificação de Imagens Digitais por Textura Usando Redes Neurais "

por

Felipe Liberman

Dissertação apresentada aos Senhores:

Prof. Dr. Dante Augusto Couto Barone

Prof. Dr. Philippe Olivier Alexandre Navaux

Prof. Dr. Lincoln de Assis Moura Jr. (USP)

Vista e permitida a impressão.
Porto Alegre, 22/11/97.

Prof. Dr. Paulo Martins Engel,
Orientador.

Prof. Flávio Rech Wagner
Coordenador da [illegible]
em Ciência da Computação
Instituto de Informática