CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Aplicação dos Algoritmos SIFT e SURF na Classificação de Sub-Imagens por Discriminação de Textura

JOZIEL PAULA NOVAIS

Marília, 2016

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Aplicação dos Algoritmos SIFT e SURF na Classificação de Sub-Imagens por Discriminação de Textura

Monografia apresentada ao Centro Universitário Eurípides de Marília como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Me. Rodolfo Barros Chiaramonte

Marília, 2016

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA - UNIVEM
MANTIDO PELA FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Joziel Paula Novais

Aplicação dos Algoritmos SIFT e SURF na Classificação de Sub-Imagens por Discriminação de Textura.

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Ciência da Computação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Ciência da Computação.

Nota: 10 (Dez)

Orientador:Rodolfo Barros Chiaramonte

1°.Examinador:Mauricio Duarte

2°.Examinador:Luan Cardoso dos Santos

Marília, 06 de dezembro de 2016.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me capacitado e guiado até aqui.

Agradeço grandemente a minha família, especialmente aos meus pais e minha irmã, por todo o incentivo e apoio que me deram.

Agradeço a todos os professores, pela paciência, dedicação e o ótimo trabalho que desempenharam durante todos esses anos, contribuindo muito para a minha formação.

Agradeço ao meu professor e orientador Rodolfo Barros Chiaramonte, por toda a orientação, pelas dicas, críticas, sugestões e esclarecimentos que tornaram possível a realização deste trabalho.

Agradeço aos meus colegas de classe pela companhia, troca de ideias e ajudas durante o curso.

Agradeço a toda a equipe de funcionários do Centro Universitário Eurípedes de Marília, pelo ótimo trabalho realizado junto a instituição, que contribuiu positivamente para a conclusão do meu trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS

ART ..... Adaptive Resonance Theory
BMU ..... Best Matching Unit
BRIEF ..... Binary Robust Independent Elementary Features
BRISK ..... Binary Robust Invariant Scalable Keypoints
DoG ..... Difference of Gaussian
EM ..... Expectation Maximization
FAST ..... Features from Accelerated Segment Test
FREAK ..... Fast Retina Keypoint
LoG ..... Laplacian of Gaussian
MCP ..... McCulloch e Pitts
MLP ..... Multilayer Perceptron
MPM ..... Maximization of the Posterior Marginals
MSER ..... Maximally Stable Extremal Regions
ORB ..... Oriented FAST and Rotated BRIEF
RBF ..... Radial Basis Function
RISAR ..... Rotation Invariant Simultaneous Autoregressive
RNA ..... Rede Neural Artificial
SAR ..... Simultaneous Autoregressive
SGLD ..... Spatial Gray Level Dependence
SIFT ..... Scale Invariant Features Transform
SOM ..... Self-Organizing Maps
SURF ..... Speeded Up Robust Features

NOVAIS, Joziel Paula. **Aplicação dos Algoritmos SIFT e SURF na Classificação de Sub-Imagens por Discriminação de Textura**. 2016. f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro Universitário Eurípides de Marília, Fundação de Ensino "Eurípides Soares da Rocha", Marília, 2016.

RESUMO

A segmentação de imagens está presente no desenvolvimento de muitas tecnologias e aplicações. A proposta deste trabalho é avaliar e comparar o desempenho dos algoritmos SIFT e SURF em um processo de classificação de sub-imagens, onde esta classificação é feita considerando as diferentes texturas da imagem. Para a proposta deste trabalho, as texturas da imagem podem ser naturais, com comportamentos aleatórios e complexos, ou formada por objetos similares de natureza complexa. Este trabalho pode ser útil para auxiliar em processos de segmentação de imagens. No desenvolvimento deste trabalho são utilizadas redes neurais artificiais para classificar os vetores descritores gerados com os algoritmos SIFT e SURF, quando aplicados em diferentes sub-imagens. As redes neurais utilizadas são do tipo SOM, portanto, utilizam um aprendizado não-supervisionado em seu treinamento. O treinamento das redes neurais é feito com padrões de entrada gerados através dos algoritmos SIFT e SURF ao serem aplicados em algumas sub-imagens. Para a realização do trabalho foram feitos estudos sobre métodos de segmentação de imagens, redes neurais artificiais e o funcionamento dos algoritmos SIFT e SURF.

Palavras-Chave: classificação, sub-imagens, extração de características, redes neurais artificiais, aprendizado não-supervisionado, segmentação de imagens

NOVAIS, Joziel Paula. **Aplicação dos Algoritmos SIFT e SURF na Classificação de Sub-Imagens por Discriminação de Textura**. 2016. f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro Universitário Eurípides de Marília, Fundação de Ensino "Eurípides Soares da Rocha", Marília, 2016.

ABSTRACT

Image segmentation is present in the development of many technologies and applications. The proposal of this work is to evaluate and compare the performance of SIFT and SURF algorithms in a classification process of sub-images, where this classification is done considering the different textures in the image. For the purpose of this work, the textures of the image may be natural, with random and complex behaviors, or being formed by similar objects of complex nature. This work may be useful to assist in image segmentation processes. In the development of this work, artificial neural networks are used to classify the descriptors vectors generated with SIFT and SURF algorithms, when applied on different sub-images. The neural networks used are of SOM type, therefore use a non-supervised learning for training. The training of the neural networks is done with input patterns generated through SIFT and SURF algorithms when applied on some sub-images. To carry out the work studies were done on image segmentation methods, artificial neural networks and the functioning of SIFT and SURF algorithms.

Keywords: classification, sub-images, feature extraction, artificial neural networks, unsupervised learning, image segmentation

CAPÍTULO 1 – INTRODUÇÃO

Devido aos crescentes avanços tecnológicos, a automatização de tarefas e processos vem sendo cada vez mais buscada em máquinas e dispositivos diversos. A automatização proporciona mais facilidade e eficiência na execução de tarefas e processos, além de ser ideal para ambientes e cenários de risco, onde seria inviável ou até mesmo impossível um ser humano realizar determinadas ações (DINIZ, 2012).

A visão é um dos mais poderosos e complexos sentidos do ser humano, sendo responsável pela maior parte de nossa compreensão das posições e propriedades dos objetos, assim como suas relações no ambiente que os cerca (COUTO, 2012). Grande parte da percepção humana é dada pela visão, e com essa percepção é possível a tomada de decisão para executar determinadas tarefas (PARKER, 2010).

A visão computacional é muito importante e fundamental em grande parte dos processos de automatização, pois ela busca emular a visão humana para avaliar o cenário ou ambiente, podendo auxiliar a tomada de decisão automática nas máquinas e dispositivos. O principal objetivo da visão computacional é fazer com que um computador possa interpretar uma imagem por meio de sistemas artificiais implementados por hardware e software, possibilitando que as máquinas possam ter uma determinada visão do ambiente externo, e executar ações com base nas informações obtidas (MAIA, 2010).

Existem muitas aplicações para a área da visão computacional e muitas delas estão bem presentes no cotidiano, como por exemplo, no reconhecimento facial em smartphones, aplicativos para câmera, robôs, veículos autônomos, análises de exames médicos, análise de imagens meteorológicas, sistemas de segurança, entre outras. A visão computacional pode ser dividida em vários subdomínios, como por exemplo, o reconhecimento de objetos, a detecção de movimentos, a reconstrução de cena, e a segmentação de imagens.

A segmentação de imagens é um dos principais subdomínios da visão computacional e está presente no desenvolvimento de várias tecnologias e aplicações como: a localização de objetos, reconhecimento facial, reconstrução 3D, controle de sistemas de tráfego, recuperação de imagens baseado em conteúdo e análises de imagens. A segmentação de imagens é um processo no qual uma imagem é subdividida em regiões distintas (BALAN, 2003). Já foram desenvolvidos vários algoritmos e técnicas para fazer a segmentação de imagens, no entanto, ainda não existe uma solução geral para o problema, sendo muitas vezes, necessária a combinação de várias técnicas para se adaptar a um determinado domínio de problema (BEJA, 2013).

1.1 – Objetivos

Neste trabalho o principal objetivo é avaliar e comparar o desempenho dos algoritmos SIFT e SURF quando aplicados em um processo de classificação de sub-imagens, onde a classificação é feita considerando as diferentes texturas da imagem, sendo que as texturas podem ser naturais, com comportamentos aleatórios complexos, ou formada por objetos similares de natureza complexa. Assim, a proposta deste trabalho pode ser útil para auxiliar na tarefa de segmentação de imagens que possuam diferentes texturas. Este tipo de segmentação de imagens é desejado em muitos projetos de automatização de processos, reconhecimento de regiões, entre outros.

Desta forma, este trabalho tem como objetivos específicos:

- Utilizar os algoritmos SIFT e SURF para a extrair características de sub-imagens;
- Utilizar redes neurais artificiais do tipo SOM para classificar vetores descritores gerados com os algoritmos SIFT e SURF, quando aplicados em sub-imagens;
- Medir os erros gerados pelas redes neurais, durante a classificação das sub-imagens, de acordo com as diferentes texturas da imagem.

1.2 – Materiais e métodos

Para o desenvolvimento do projeto, faz-se necessário seguir as seguintes etapas:

- Levantamento bibliográfico e pesquisa acerca de vários estudos sobre as principais técnicas de segmentação de imagens, sobre redes neurais artificiais, e também sobre os algoritmos SIFT e SURF;
- Implementação dos algoritmos SIFT e SURF para a detecção e descrição de pontos de interesse em imagens, através da biblioteca de visão computacional OpenCV 3.0;
- Implementação de uma rede neural auto organizável do tipo SOM adaptada para o descritor gerado com o algoritmo SIFT, utilizando o framework Encog;
- Implementação de uma rede neural auto organizável do tipo SOM adaptada para o descritor gerado com o algoritmo SURF, utilizando o framework Encog;
- Realização do treinamento das redes neurais com os descritores gerados através dos algoritmos SIFT e SURF, a partir de diferentes sub-imagens;
- Execução de testes para medir e comparar o desempenho dos algoritmos SIFT e SURF no processo de classificação de sub-imagens proposto neste trabalho.

1.3 – Trabalhos correlatos

Gomes (2009), propõe um método de segmentação por discriminação de texturas, que utiliza uma abordagem similar com a proposta deste trabalho, devido ao uso de redes neurais auto organizáveis na classificação dos pixels da imagem, e a utilização de funções arbitrárias para extrair as características das texturas da imagem, e assim descrevê-las.

Em (BELUCO, 2002) é proposto um método para a classificação de imagens por textura, que utiliza a transformada de Fourier ao aplicar filtros de Gabor para descrever as texturas. Neste método, uma rede neural MLP é responsável por fazer a classificação das imagens.

De Freitas et al (2015), propõe um método de classificação de texturas baseado também no uso de uma rede neural MLP para classificar as diferentes texturas. Os descritores de Haralick são aplicados para extrair características e consequentemente fazer a descrição das texturas.

Liberman (1997) faz o uso de alguns descritores extraídos através de uma matriz de co-ocorrência, para caracterizar as texturas de imagens digitais e assim classificá-las com o auxílio de uma rede neural MLP.

Em (LEE et al, 2014) é apresentada uma comparação feita entre os algoritmos SIFT, SURF e ORB, considerando o desempenho obtido com cada um deles na classificação de texturas. Em conjunto com cada um dos algoritmos é utilizado o algoritmo de agrupamento *K-means* para gerar histogramas, que são utilizados na descrição das texturas. Foram realizados vários testes com diferentes classificadores, incluindo uma rede neural.

Soottitantawat et al (2011) propõe um método de classificação de texturas, que utiliza o algoritmo SURF em conjunto com o algoritmo *K-means* para fazer a descrição de texturas. Para fazer a classificação das diferentes texturas, é utilizada uma máquina de vetores suporte.

Nos capítulos subsequentes serão mostrados alguns estudos feitos sobre segmentação de imagens, algoritmos detectores e descritores de pontos chave para extração de características com foco nos algoritmos SIFT e SURF, e também um estudo sobre redes neurais artificiais.

CAPÍTULO 2 – SEGMENTAÇÃO DE IMAGENS

A segmentação é um dos processos mais importantes da visão humana. Ela pode ser descrita como uma forma de dividir a imagem capturada pelo olho humano em regiões de natureza similar. A segmentação é fundamental para a percepção do mundo externo, sendo um processo que ocorre de forma natural na visão humana, influenciando muitas de nossas ações.

Na visão computacional, a segmentação de imagens subdivide uma imagem em regiões ou objetos que a compõem, assim, o resultado gerado é um conjunto de regiões, objetos ou contornos extraídos da imagem (SANTOS, 2012).

O nível de detalhe em que deve ser realizada a subdivisão depende do problema a ser resolvido. O processo de segmentação para quando os objetos ou as regiões de interesse de uma aplicação forem detectadas (GONZALES, 2011).

A tarefa de segmentação de imagens não triviais é uma das mais difíceis no processamento de imagens, e a precisão desta tarefa é responsável por determinar o sucesso ou o fracasso final dos procedimentos de análise computadorizada (GONZALES, 2011).

Grande parte das técnicas de segmentação de imagens utilizam em seu processamento, algumas propriedades computacionais dos pixels da imagem como intensidade, cor e textura (GOMES, 2009).

De acordo com Gonzales (2011), sendo $\boldsymbol{R}$ a região total da imagem, e $\boldsymbol{R_i}$ uma das $\boldsymbol{n}$ regiões em que a imagem é dividida, as seguintes propriedades devem ser verdadeiras:

- $\boldsymbol{R_1} \cup \boldsymbol{R_2} \cup \ldots \cup \boldsymbol{R_n} = \boldsymbol{R}$
- $\boldsymbol{R_i}$ é uma região conexa com $\boldsymbol{i} = 1, 2, \ldots, \boldsymbol{n}$
- $\boldsymbol{R_i} \cap \boldsymbol{R_j} = \emptyset$ para todo $\boldsymbol{i}$ e $\boldsymbol{j}$, onde $\boldsymbol{i} \neq \boldsymbol{j}$
- $\boldsymbol{P}(\boldsymbol{R_i}) =$ **VERDADEIRO** para todo $\boldsymbol{i} = 1, 2, \ldots,$ n
- $\boldsymbol{P}(\boldsymbol{R_i} \cup \boldsymbol{R_j}) =$ **FALSO** para todo $\boldsymbol{i} \neq \boldsymbol{j}$

Onde $\boldsymbol{P}(\boldsymbol{R_i})$ é um predicado lógico definido sobre todos os pixels da região $\boldsymbol{R_i}$ que possuem características similares.

A seguir serão apresentadas algumas técnicas de segmentação de imagens desenvolvidas ao longo do tempo.

2.1 – Técnicas baseadas em similaridades

Algumas técnicas de segmentação são baseadas no conceito de similaridade, que consiste na divisão da imagem em regiões que sejam semelhantes de acordo com um conjunto de critérios predefinido (GONZALES, 2011).

2.1.1 – Limiarização

A técnica de limiarização geralmente usa a informação do nível de cinza dos pixels de uma imagem como base para fazer a segmentação. O nível de cinza de cada pixel é comparado com o valor de um determinado limiar **T**, também chamado de *"threshold"*. Assim, através deste limiar **T**, é possível fazer a separação entre os pixels que possuem um nível de cinza superior a ele, e os pixels que possuem um nível de cinza inferior (GONZALES, 2011).

Geralmente usada em imagens com áreas homogêneas sobre um plano de fundo uniforme, a limiarização pode ser aplicada, por exemplo, na extração de texto de imagens, e na separação de fundo e objetos de uma imagem.

A partir de uma imagem de entrada $\boldsymbol{f}$ com $\boldsymbol{N}$ níveis de cinza, a operação de limiarização produz como saída uma imagem $\boldsymbol{g}$, onde o número de níveis de cinza é menor que $\boldsymbol{N}$. Normalmente, a imagem $\boldsymbol{g}$ apresenta dois níveis de cinza.

Assim sendo, quando for utilizado apenas um limiar $\mathbf{T}$, cada pixel $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y})$ da imagem gerada através da limiarização, é dado por:

$$\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) = \begin{cases} \boldsymbol{A}, \text{se } \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}) > \mathbf{T} \\ \boldsymbol{B}, \text{se } \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}) \leq \mathbf{T} \end{cases}$$

Onde $\boldsymbol{A}$ e $\boldsymbol{B}$ podem ser quaisquer dois valores distintos para representar a intensidade do nível de cinza.

Quando existir apenas um limiar, esta operação é chamada de **limiarização simples**, enquanto que o termo **limiarização múltipla** é usado quando houver dois ou mais limiares envolvidos (GONZALES, 2011).

Quando o limiar **T** é aplicado na imagem inteira, o processo de limiarização é conhecido como **limiarização global**. No entanto se o valor de **T** mudar ao longo da imagem, é usado o termo **limiarização variável**. O termo **limiarização local** ou **limiarização regional**, geralmente é usado para representar a limiarização variável na qual o valor de **T** em qualquer ponto $(\boldsymbol{x}, \boldsymbol{y})$ de uma imagem, depende das propriedades dos pixels que formam sua vizinhança (GONZALES, 2011).

O valor do limiar pode ser obtido por meio de uma inspeção visual do histograma da imagem, que fornece a informação sobre a quantidade de pixels dentro de cada nível de cinza. O valor escolhido para o limiar geralmente se localiza nos pontos de mínimo do histograma, conhecidos também como vales, assim como é mostrado na Figura 2.1, onde **T** é o limiar escolhido.

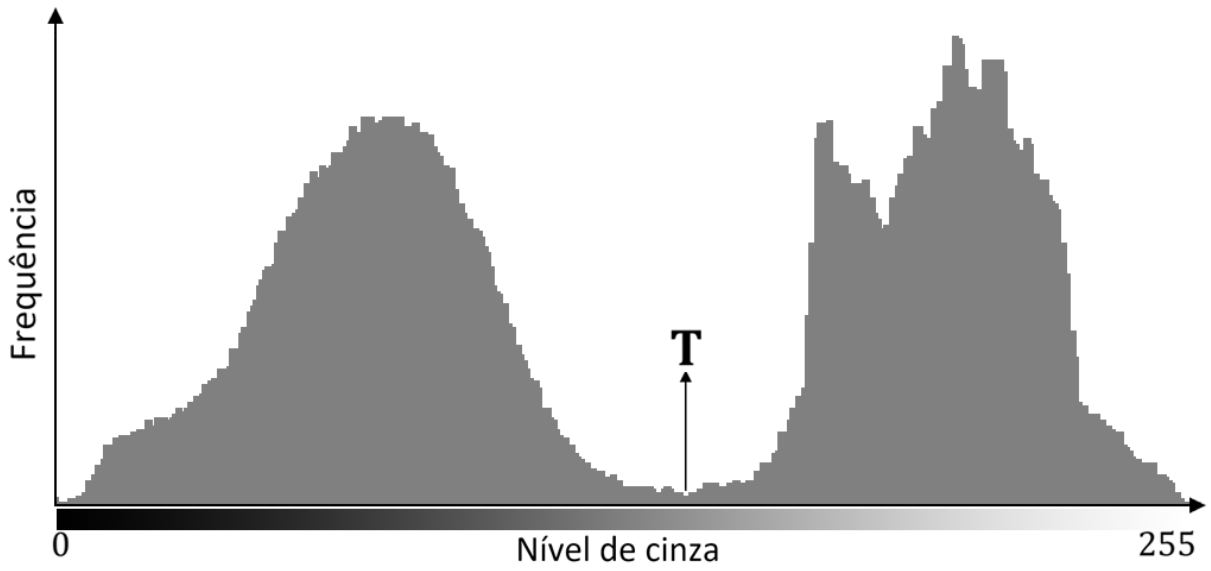

Figura 2.1 – Exemplo de Histograma de Imagem. Fonte: Elaborada pelo autor.

2.1.2 – Aglomeração

A técnica de aglomeração vem sendo muito utilizada na área de segmentação de imagens, sendo baseada no conceito de agrupamento dos pixels que possuem características similares com base em critérios de semelhança, formando aglomerados, também chamados de *"clusters"* (HIRATA JR, 1997).

Os critérios de semelhança podem ser baseados em características como por exemplo, o tom de cinza, os valores do espaço de cores utilizado, a localização do pixel e até mesmo a textura presente em uma janela de pixels, onde o pixel analisado está localizado no centro (HIRATA JR, 1997).

Ao longo do tempo, várias técnicas de aglomeração foram desenvolvidas para fazer a segmentação de imagens, entre elas, as mais populares são: o algoritmo *K-means* (MACQUEEN et al, 1967), o algoritmo EM (*Expectation Maximization*) (DEMPSTER et al, 1977), o algoritmo *Mean Shift* (COMANICIU et al, 2002), a técnica de segmentação por *Watershed* (MEYER, 1994) e a segmentação através de corte em grafos (SHI et al, 2000).

2.1.3 – Crescimento de regiões

Segundo Gonzales (2011), este método consiste em agregar conjuntos de pixels em regiões maiores com base em alguns critérios de semelhança predefinidos.

Nesta técnica, o processamento é iniciado com um pixel ou um grupo de pixels, chamados de "sementes", que são escolhidos por pertencerem a áreas de interesse na imagem. A escolha das sementes pode ser feita de forma manual ou automática. Na próxima etapa, os pixels vizinhos de cada semente são analisados, para que caso algum deles satisfaça os critérios de semelhança, ele possa ser adicionado à região da semente ou grupo de sementes iniciais, fazendo com que haja o crescimento desta região. Este processo se repete até o momento em que mais nenhum pixel possa ser adicionado.

Embora esta técnica seja simples, ela pode apresentar alguns problemas, como por exemplo, a dificuldade em selecionar as sementes que possam representar as regiões de interesse e gerar um bom resultado de segmentação, e também a dificuldade em selecionar os critérios de semelhança ideais para o tipo de imagem utilizada (GONZALES, 2011).

2.1.4 – Separação e junção

O método de separação e junção tem uma estratégia oposta à de crescimento de regiões. Nesta técnica, o processo de segmentação se inicia a partir de toda a imagem, a qual é subdividida em um conjunto de regiões distintas e arbitrárias, em seguida, é feita a fusão entre algumas destas regiões, e também a divisão de outras regiões, de acordo com uma propriedade de homogeneidade $\boldsymbol{Q}$ predefinida. Este processo se repete até que todas as subdivisões da imagem satisfaçam a propriedade $\boldsymbol{Q}$ (GONZALES, 2011).

Esta técnica em particular pode ser representada convenientemente na forma dos chamados *"quadtrees"*, que são árvores em que cada nó dá origem a quatro descendentes, assim como é mostrado na Figura 2.2 (a), onde o nó raiz **R** representa a imagem inteira, e os nós filhos representam as subdivisões. O processo de divisão da imagem é ilustrado na Figura 2.2 (b).

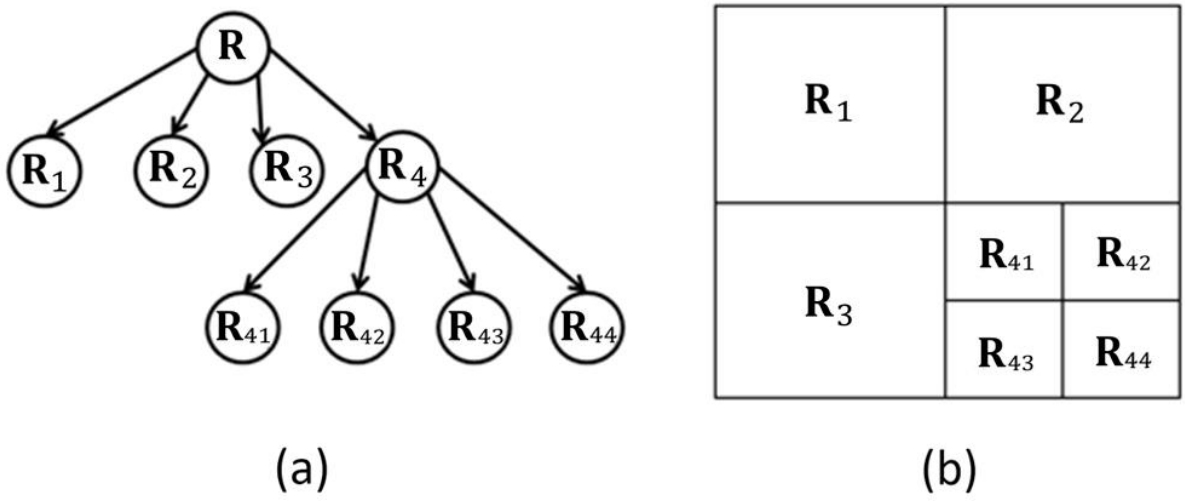

Figura 2.2 – Exemplo de Quadtree (a) e Divisão de imagem por Quadtree (b). Fonte: Elaborada pelo autor.

Esta técnica de segmentação pode ser resumida pelo procedimento a seguir, onde em qualquer etapa é possível:

1. Dividir em quatro quadrantes distintos qualquer região $\mathbf{R_i}$ para a qual: $\boldsymbol{Q}(\mathbf{R_i}) = \mathbf{FALSO}$.
2. Quando não for possível continuar dividindo, fundir as regiões adjacentes $\mathbf{R_j}$ e $\mathbf{R_k}$ para as quais: $\boldsymbol{Q}(\mathbf{R_j} \cup \mathbf{R_k}) = \mathbf{VERDADEIRO}$.
3. Parar o procedimento quando não houver mais alguma região que possa ser fundida ou dividida.

2.2 – Técnicas baseadas em descontinuidades

A segmentação baseada na detecção de descontinuidades leva em consideração, a necessidade de existir uma fronteira entre duas regiões de uma imagem. As descontinuidades são representadas pelas mudanças relativamente abruptas nos níveis de tons de cinza, e elas podem ser detectadas na forma de pontos isolados, linhas e bordas ou contornos dos objetos presentes na imagem (BALAN, 2003).

O método mais comum para detectar descontinuidades é através da aplicação de um filtro espacial com uma máscara em toda imagem em cada um dos seus pixels. Como exemplo, poderia ser usada uma máscara 3 × 3, como apresentado na Figura 2.3, onde deve ser calculado o somatório dos produtos resultantes da multiplicação de cada coeficiente da máscara pelo nível de cinza de seu pixel correspondente da imagem (GONZALES, 2011). Este procedimento também pode ser representado pela expressão a seguir:

$$\boldsymbol{R = w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9}$$

$$\boldsymbol{= \sum_{i=1}^{9} w_i z_i}$$

Equação 2.1

Onde $\boldsymbol{z_i}$ é o nível de cinza do pixel associado ao coeficiente $\boldsymbol{w_i}$ da máscara, e $\boldsymbol{R}$ é o resultado obtido através da máscara utilizada.

Desta forma, o cálculo é realizado em relação à localização central da máscara, portanto, a cada nova posição da máscara sobre a imagem, o pixel central da sub imagem abrangida pela máscara será alterado.

$\boldsymbol{w_1}$	$\boldsymbol{w_2}$	$\boldsymbol{w_3}$
$\boldsymbol{w_4}$	$\boldsymbol{w_5}$	$\boldsymbol{w_6}$
$\boldsymbol{w_7}$	$\boldsymbol{w_8}$	$\boldsymbol{w_9}$

Figura 2.3 – Máscara geral de filtro espacial 3 × 3. Fonte: Elaborada pelo autor.

2.2.1 – Detecção de pontos isolados

A detecção de pontos isolados em uma imagem pode ser feita através do uso da máscara mostrada na Figura 2.4 (GONZALES, 2011).

-1	-1	-1
-1	8	-1
-1	-1	-1

Figura 2.4 – Máscara 3 × 3 para detecção de pontos isolados. Fonte: Elaborada pelo autor.

Um ponto isolado é identificado na posição em que a máscara está centralizada se $|\boldsymbol{R}| > \mathbf{T}$, onde **T** é um limiar de valor positivo, e $\boldsymbol{R}$ é dado pela Equação 2.1.

Esta técnica parte do princípio de que um ponto isolado terá o nível de cinza muito diferente dos níveis de cinza de seus vizinhos. Assim sendo, neste processo, o pixel que se encontra no centro da máscara, é detectado como um ponto isolado se houver uma determinada diferença de nível de cinza entre ele e os demais pixels vizinhos.

2.2.2 – Detecção de linhas

Esta técnica apresenta um processo similar ao da detecção de pontos isolados, no entanto, para a segmentação de linhas, geralmente é utilizada a aplicação das máscaras mostradas na Figura 2.5 (GONZALES, 2011). Estas máscaras são úteis na detecção de linhas com um pixel de espessura, sendo a primeira máscara usada na detecção de linhas horizontais, pois o valor de $\boldsymbol{R}$ na Equação 2.1 será maior quando uma linha horizontal estiver localizada na linha central da primeira máscara.

O mesmo princípio vale para as outras máscaras da Figura 2.5, onde a segunda máscara apresentará um valor de $\boldsymbol{R}$ maior quando uma linha inclinada a $45°$ estiver situada na diagonal secundária, a terceira, quando uma linha vertical estiver situada na coluna central, e a quarta para linhas inclinadas a $-45°$ quando estiverem situadas na diagonal principal.

Desta forma, para detectar todas as linhas em uma imagem no sentido definido por uma máscara específica, é necessário simplesmente executar a máscara por meio da imagem e estabelecer um limiar comparando-o ao módulo do valor resultante de $\boldsymbol{R}$ definido pela Equação 2.1, para que então possa ser feita a separação dos pixels pertencentes à linha detectada.

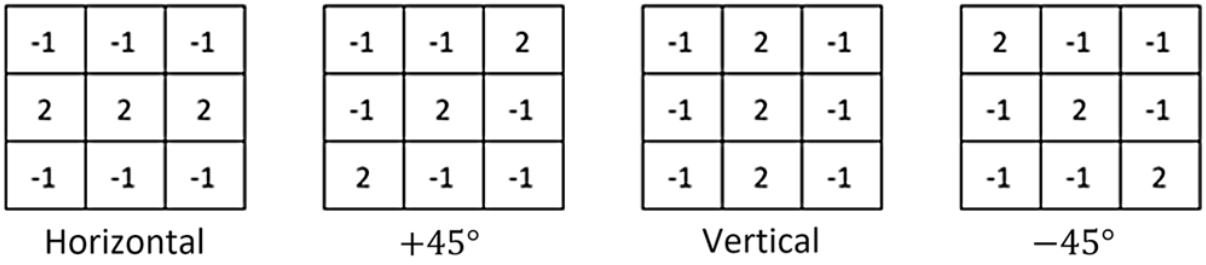

Figura 2.5 – Máscaras 3 × 3 para detecção de linhas. Fonte: Elaborada pelo autor.

2.2.3 – Detecção de bordas

Um contorno ou borda pode ser definido como o limite entre duas regiões com certa diferença de tons de cinza. A detecção de borda é a abordagem mais utilizada dentro da detecção de descontinuidades, pois pontos e linhas com apenas um pixel de espessura raramente estão presentes na maioria das aplicações reais (OHNISHI, 2005). As bordas encontradas nas imagens podem ser basicamente de três tipos: bordas em degrau, borda em rampa e borda em forma de telhado, assim como é mostrado na Figura 2.6.

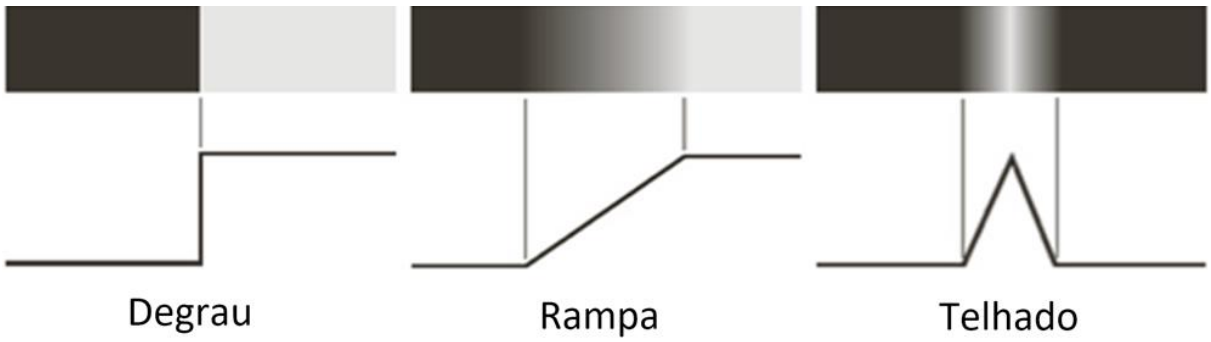

Figura 2.6 – Tipos de borda. Adaptado de: (GONZALES, 2011).

Para realizar a detecção de bordas, geralmente é utilizado o gradiente $\boldsymbol{G(x,y)}$, que é um operador de derivação local.

O gradiente $\boldsymbol{G(x,y)}$, denotado por $\boldsymbol{\nabla f}$, é dado por:

$$\boldsymbol{\nabla f = [G_x\ G_y\ ] = \left[\frac{\partial f}{\partial x}\ \frac{\partial f}{\partial y}\right]}$$

De acordo com Gonzales (2011), uma das formas mais simples de computar o gradiente é fazer uma aproximação digital das derivadas de primeira ordem, usando máscaras de tamanho 3 × 3, que são dadas pelas equações:

$$G_x = \frac{\partial f}{\partial x} = (w_7 + w_8 + w_9) - (w_1 + w_2 + w_3)$$

Equação 2.2

$$G_y = \frac{\partial f}{\partial y} = (w_3 + w_6 + w_9) - (w_1 + w_4 + w_7)$$

Equação 2.3

Onde w_i é um coeficiente de uma máscara geral 3 × 3, ilustrada na Figura 2.3.

Estas equações podem ser executadas ao longo de toda a imagem, para fazer a sua filtragem com as máscaras da Figura 2.7, as quais recebem o nome de operadores de Prewitt (GONZALES, 2011).

Prewitt

-1	-1	-1
0	0	0
1	1	1

Borda Horizontal

-1	0	1
-1	0	1
-1	0	1

Borda Vertical

0	1	1
-1	0	1
-1	-1	0

Borda em −45°

-1	-1	0
-1	0	1
0	1	1

Borda em 45°

Figura 2.7 – Operadores de Prewitt. Fonte: Elaborada pelo autor.

Uma pequena variação das duas equações anteriores utiliza o valor 2 como peso no coeficiente central:

$$G_x = \frac{\partial f}{\partial x} = (w_7 + 2w_8 + w_9) - (w_1 + 2w_2 + w_3)$$

Equação 2.4

$$G_y = \frac{\partial f}{\partial y} = (w_3 + 2w_6 + w_9) - (w_1 + 2w_4 + w_7)$$

Equação 2.5

A utilização do valor 2 na posição central produz a suavização da imagem, fazendo com que haja uma supressão de ruído. A Figura 2.8 mostra as máscaras utilizadas para implementar as equações. Essas máscaras são chamadas de operadores de Sobel (GONZALES, 2011).

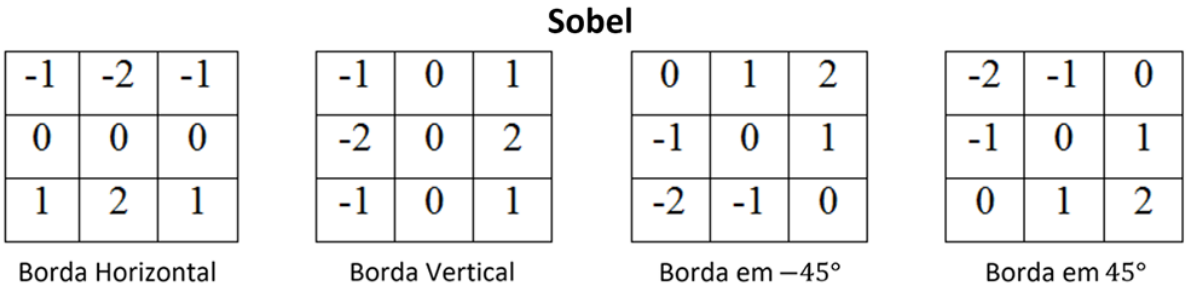

Figura 2.8 – Operadores de Sobel. Fonte: Elaborada pelo autor.

Assim, estas máscaras que foram abordadas (Figuras 2.7 e 2.8), são utilizadas na obtenção dos componentes de gradiente $\boldsymbol{G_x}$ e $\boldsymbol{G_y}$ para cada pixel da imagem. Esses componentes, que são duas derivadas parciais, serão utilizados para estimar a intensidade e a direção da borda (GONZALES, 2011).

2.3 – Técnicas baseadas em textura

Existem algumas técnicas de segmentação que são baseadas no uso de informações sobre as texturas de uma imagem. Textura é uma característica que está presente em diferentes áreas ou objetos de uma imagem. Embora seja bem reconhecida e até mesmo implicitamente intuitiva, esta característica ainda não possui uma definição formal estabelecida, sendo muitas vezes considerada como um conceito extenso e subjetivo, geralmente associado a padrões de luminosidade e cores (BALAN, 2003) (GOMES, 2009).

Segundo Gonzales (2011), a textura pode ser definida como um descritor que fornece medidas de propriedades como suavidade, rugosidade e regularidade. Em (IEEE, 1990), a textura é definida como sendo um atributo que representa o arranjo espacial dos tons de cinza dos pixels em uma região da imagem.

Para que possa ser feita a segmentação da imagem por texturas, o primeiro passo necessário é a extração de características, para descrever as diferentes texturas presentes na imagem. Este primeiro passo é crucial para a segmentação, pois é através dele que os pixels da imagem poderão ser classificados de acordo com as texturas. Várias abordagens foram desenvolvidas para realizar a extração de características e possibilitar a segmentação de texturas, nas próximas seções serão apresentadas quatro delas, a abordagem estatística, a abordagem estrutural, a abordagem espectral e abordagem baseada em modelos.

2.3.1 – Abordagem estatística

Na abordagem estatística é feita a representação quantitativa de padrões visuais da textura, para isso, é realizada a extração de características da textura. Dependendo do cenário também pode ser feita a seleção ou redução da quantidade de características extraídas, para que por fim seja aplicado um algoritmo de segmentação (BALAN, 2003).

Um dos métodos desta abordagem, é o uso dos momentos estatísticos (GONZALES, 2011), que são calculados com base no histograma de intensidade da imagem ou de uma região dela. Estes momentos podem ser de ordem $\boldsymbol{n}$, e estão associados com representações estatísticas relacionadas à distribuição de intensidades na imagem.

Considerando $\boldsymbol{z}$ uma variável aleatória que representa o conceito de intensidade, a fórmula geral para o cálculo de um momento $\boldsymbol{\mu}$ de ordem $\boldsymbol{n}$ é mostrada na Equação 2.6.

$$\boldsymbol{\mu_n(z) = \sum_{i=0}^{L-1} (z_i - m)^n p(z_i)}$$

Equação 2.6

Onde, $\boldsymbol{i}$ é um dos $\boldsymbol{L}$ níveis distintos de intensidade do histograma, $\boldsymbol{p(z_i)}$ é a quantificação do número de ocorrências de $\boldsymbol{z_i}$, e $\boldsymbol{m}$ é o valor médio de $\boldsymbol{z}$, sendo, portanto, a intensidade média, que é dada por:

$$\boldsymbol{m = \sum_{i=0}^{L-1} z_i\, p(z_i)}$$

Equação 2.7

O segundo momento ou momento de ordem 2 é particularmente importante para a descrição de texturas, pois fornece uma medida de contraste de intensidade, podendo ser usada para a descrição de suavidade relativa ou rugosidade de uma textura (GONZALES, 2011).

O terceiro momento, é uma medida da assimetria do histograma, e ele mostra se existem mais valores do lado esquerdo ou do lado direito, indicando se os níveis de intensidade têm tendência para o lado escuro ou para o lado claro. O quarto momento é uma medida que informa o quão plano é o histograma. Para a descrição das texturas, as informações obtidas a partir do terceiro e do quarto momento, são úteis apenas quando existem grandes variações entre as medidas (GONZALES, 2011).

O quinto momento e os momentos mais elevados já não estão mais fortemente relacionados com o formato do histograma, e fornecem informações quantitativas adicionais sobre o conteúdo da textura (GONZALES, 2011).

Por não levar em consideração as posições relativas entre os pontos da região da textura, mas apenas a relação entre as suas intensidades, os momentos estatísticos são limitados para descrever as texturas de forma mais consistente (GONZALES, 2011).

Outro método muito utilizado nesta abordagem estatística, consiste no uso das matrizes de co-ocorrência, que são capazes de representar a distribuição espacial de padrões de níveis de cinza de uma textura, levando em consideração as posições relativas dos pixels da imagem (GONZALES, 2011).

As matrizes de co-ocorrência, também são conhecidas como matrizes SGLD (*Spatial Gray Level Dependence*). O seu funcionamento se baseia no uso de um determinado parâmetro $\boldsymbol{Q}$, o qual é um operador que define a posição relativa entre dois pixels, considerando a distância e o ângulo entre eles, através dos parâmetros $\boldsymbol{dx}$ e $\boldsymbol{dy}$, que são a distância em $\boldsymbol{x}$ e a distância em $\boldsymbol{y}$ respectivamente. Assim sendo, considerando uma imagem $\boldsymbol{f}$, com $\boldsymbol{L}$ níveis de intensidade possíveis, a matriz de co-ocorrência $\boldsymbol{G}$ é uma matriz cujo elemento $\boldsymbol{G_{ij}}$ é a quantidade de vezes que os pares de pixels com intensidades $\boldsymbol{z_i}$ e $\boldsymbol{z_j}$ ocorrem em $\boldsymbol{f}$ na posição definida pelo operador $\boldsymbol{Q}$, sendo que $1 \leq \boldsymbol{i}$ e $\boldsymbol{j} \leq \boldsymbol{L}$. O número de níveis de intensidade possíveis na imagem determina as dimensões da matriz $\boldsymbol{G}$, portanto se uma imagem possuir, por exemplo, 255 tons de cinza, a matriz $\boldsymbol{G}$ será de 255 × 255 (GONZALES, 2011).

Os valores presentes na matriz de co-ocorrência $\boldsymbol{G}$ dependem diretamente do operador $\boldsymbol{Q}$, portanto, os padrões de textura de intensidade de uma imagem podem ser detectados através da escolha de um operador de posição adequado de acordo com a textura, para que depois os elementos de $\boldsymbol{G}$ possam ser analisados. Para imagens com texturas refinadas, é necessário que sejam criadas matrizes através de um operador $\boldsymbol{Q}$ com baixos valores para $\boldsymbol{dx}$ e $\boldsymbol{dy}$, geralmente 1 ou 2, enquanto que para texturas mais grosseiras costuma-se utilizar valores maiores (BALAN, 2003). Na Figura 2.11 são mostrados três exemplos de matrizes de co-ocorrência.

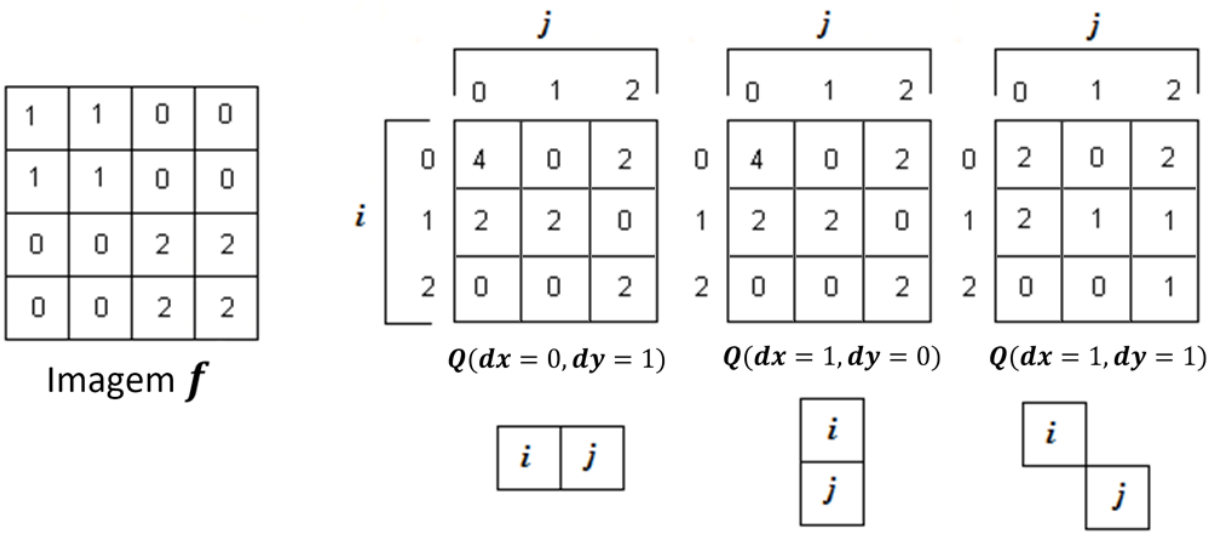

Figura 2.11 – Exemplos de matrizes de co-ocorrência. Fonte: Elaborada pelo autor.

Para fazer a caracterização do conteúdo da matriz $\boldsymbol{G}$, diversos são os descritores úteis que podem ser extraídos de uma matriz de co-ocorrência. Neste trabalho serão mostrados os 5 mais comuns e utilizados na literatura (BALAN, 2003). As fórmulas de cada descritor são válidas para matrizes de co-ocorrência de tamanho $\boldsymbol{K} \times \boldsymbol{K}$, onde o termo $\boldsymbol{p_{ij}}$ é o termo na posição $\boldsymbol{ij}$ da matriz de co-ocorrência $\boldsymbol{G}$ divido pela soma dos elementos de $\boldsymbol{G}$.

1. **Energia**

O descritor energia é uma medida de suavidade ou uniformidade de uma imagem, e seu resultado se encontra no intervalo $[0, 1]$. O valor do descritor energia é 1 para uma imagem constante, assim, este descritor indica o grau de uniformidade dos elementos da matriz de co-ocorrência (GONZALES, 2011) (BALAN, 2003). A fórmula da energia é dada por:

$$\sum_{i=1}^{K} \sum_{j=1}^{K} p_{ij}^2$$

Equação 2.8

2. **Entropia**

O descritor entropia é a medida da aleatoriedade dos elementos da matriz $\boldsymbol{G}$. O seu valor é 0 quando todos os elementos $\boldsymbol{p_{ij}}$ equivalem a 0, e é máxima quando todos os elementos $\boldsymbol{p_{ij}}$ são iguais. O valor máximo deste descritor é $2\boldsymbol{log_2 k}$, e ele mostra características do comportamento periódico da textura (GONZALES, 2011) (BALAN, 2003). Sua fórmula de cálculo é dada por:

$$-\sum_{\boldsymbol{i}=1}^{\boldsymbol{K}}\sum_{\boldsymbol{j}=1}^{\boldsymbol{K}}\boldsymbol{p_{ij}log_2 p_{ij}}$$

Equação 2.9

3. **Contraste**

O descritor contraste, é a medida da variação de intensidade entre um pixel e seu vizinho em toda a imagem. O valor do contraste é 0, quando os valores da matriz $\boldsymbol{G}$ forem constantes, e no máximo $(\boldsymbol{K}-1)^2$, que representará a máxima variação entre os valores da matriz. O contraste age como um indicador do nível de contraste da textura (GONZALES, 2011) (BALAN, 2003). Sua fórmula é dada por:

$$\sum_{\boldsymbol{i}=1}^{\boldsymbol{K}}\sum_{\boldsymbol{j}=1}^{\boldsymbol{K}}(\boldsymbol{i}-\boldsymbol{j})^2\boldsymbol{p_{ij}}$$

Equação 2.10

4. **Homogeneidade**

O descritor homogeneidade é a medida do grau de semelhança da intensidade entre um pixel e seu vizinho em toda a imagem, medindo a proximidade espacial da distribuição de elementos de $\boldsymbol{G}$ na diagonal. O intervalo de valores é $[0, 1]$, alcançando o seu máximo quando $\boldsymbol{G}$ for uma matriz diagonal (GONZALES, 2011) (BALAN, 2003). Sua fórmula é dada por:

$$\sum_{\boldsymbol{i}=1}^{\boldsymbol{K}}\sum_{\boldsymbol{j}=1}^{\boldsymbol{K}}\boldsymbol{p_{ij}}\ /\ (1+|\boldsymbol{i}-\boldsymbol{j}|)$$

Equação 2.11

5. Probabilidade Máxima

O descritor probabilidade máxima indica qual é o elemento que apresenta a resposta mais forte da matriz $\boldsymbol{G}$ (GONZALES, 2011). Sua fórmula é dada por:

$$\boldsymbol{max_{i,j}(p_{ij})}$$

Equação 2.12

Assim sendo, o objetivo é caracterizar o conteúdo das matrizes de co-ocorrência a partir destes descritores. A grande desvantagem da utilização das matrizes de co-ocorrência na extração de características de texturas se deve ao alto custo computacional, causado pelas dimensões geralmente grandes das matrizes. Além disso, para definir o operador $\boldsymbol{Q}$ muitas vezes é necessário ter um conhecimento prévio sobre a imagem ou textura, o que nem sempre é possível (BALAN, 2003).

2.3.2 – Abordagem estrutural

A abordagem estrutural é utilizada na segmentação de imagens que possuem texturas com comportamento estrutural definido ou determinístico. Na maioria dos casos elas são compostas por um arranjo repetitivo de elementos estruturais chamados de *textels* (BALAN, 2003).

Por exemplo, suponha que exista uma regra na forma $\boldsymbol{S \rightarrow aS}$, indicando que o símbolo $\boldsymbol{S}$ poderá gerar a cadeia $\boldsymbol{aS}$, com três aplicações desta regra, seria produzida a cadeia $\boldsymbol{aaaS}$. Se levarmos em consideração que $\boldsymbol{a}$ representa um textel na forma de um triângulo (Figura 2.12 (a)) e atribuir o significado de "triângulos para a direita" a uma string de caracteres do tipo $\boldsymbol{aaa}$ ..., então a regra $\boldsymbol{S \rightarrow aS}$ será capaz de gerar um padrão de textura como o que é mostrado na Figura 2.12 (b) (GONZALES, 2011).

Caso em seguida sejam adicionadas algumas regras novas a esse esquema, como: $\boldsymbol{S \rightarrow bA, A \rightarrow cA, A \rightarrow c, A \rightarrow bS, S \rightarrow a}$, onde a presença de um $\boldsymbol{b}$ tenha o significado "triângulo abaixo", e o significado de $\boldsymbol{c}$, seja "triângulo à esquerda", torna-se possível gerar uma string de caracteres da forma $\boldsymbol{aaabccbaa}$ que corresponde a uma matriz 3 × 3 de triângulos. Desta forma, padrões de textura maiores podem ser facilmente formados, como os da Figura 2.12 (c) (GONZALES, 2011).

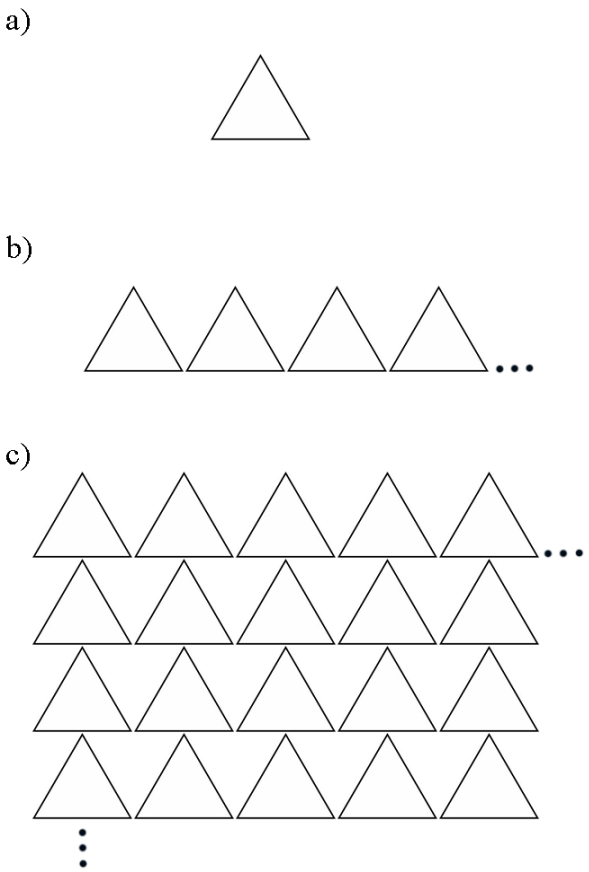

Figura 2.12 – Padrões de textura gerados pela abordagem estrutural. Fonte: Elaborada pelo autor.

Como a grande maioria das texturas não apresentam este tipo de comportamento (Figura 2.12), o uso da abordagem estrutural, na prática abrange um conjunto limitado de problemas (BALAN, 2003).

2.3.3 – Abordagem espectral

Na abordagem espectral, um método muito utilizado é a análise de energia do espectro de Fourier, que é gerado pela transformada de Fourier. O espectro de Fourier descreve uma imagem através de suas frequências espaciais, podendo ser útil na descrição de texturas (BALAN, 2003).

Neste método, é feita uma análise sistemática do espectro de Fourier, tornando possível extrair características das texturas, como o seu período espacial e a sua direção. Para isso, é necessário analisar a concentração de energia do espectro de Fourier, em regiões específicas, como ao longo de picos ou anéis, assim como é ilustrado na Figura 2.13, para que seja possível a caracterização da textura (BALAN, 2003).

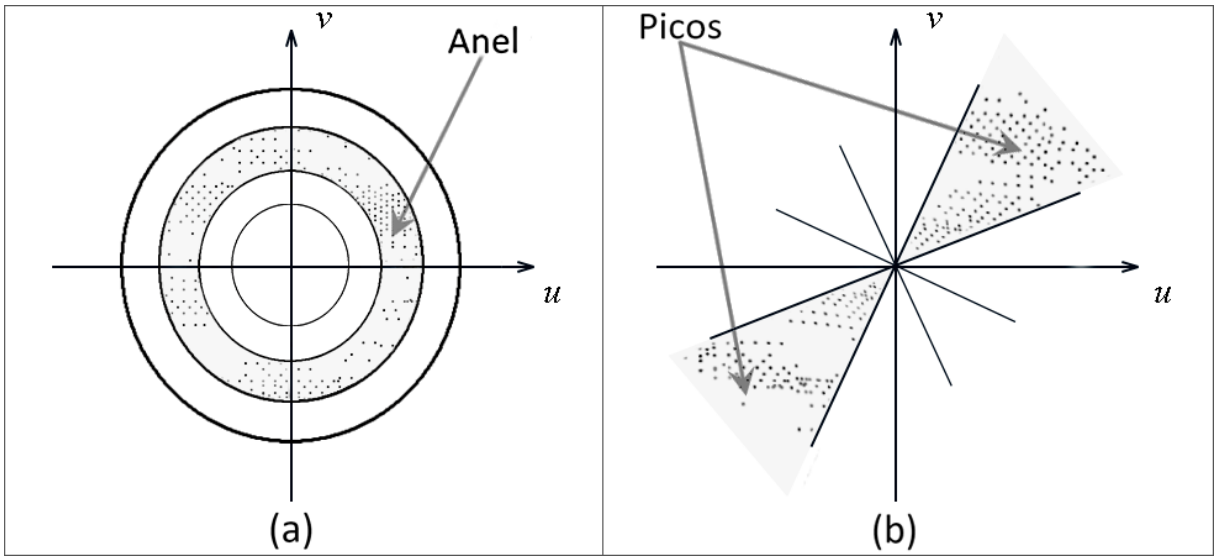

Figura 2.13 – Extração de características a partir do espectro de Fourier. Adaptado de: (BALAN, 2003).

As características extraídas a partir das regiões dos anéis (Figura 2.13 (a)) refletem os períodos espaciais da textura. Quando há uma alta concentração de energia em anéis com raios maiores, isso geralmente caracteriza texturas finas, ou altas frequências no padrão de textura. Já quando há uma alta concentração de energia em anéis com raios menores, isso geralmente é característico de texturas mais grosseiras e esparsas ou baixas frequências no padrão de textura (BALAN, 2003).

As características extraídas com base na análise de regiões de picos proeminentes (Figura 2.13 (b)) do espectro de Fourier apontam informações sobre a direção principal da textura. Desta forma, caso o padrão de textura apresente um comportamento direcional em um ângulo $\boldsymbol{\phi}$, no espectro de Fourier será observado, uma alta concentração de energia em uma região de pico com direção $\boldsymbol{\phi} + \boldsymbol{\pi}/2$ (BALAN, 2003).

A principal desvantagem em utilizar o espectro de Fourier na descrição de texturas, é o seu poder de descrição. Julesz (1983), mostrou que é possível que texturas diferentes gerem concentrações de energia idênticas no espectro de Fourier.

2.3.4 – Abordagem baseada em modelos

Nesta abordagem, a imagem que será segmentada é tratada como um modelo de probabilidade ou como uma combinação linear de um conjunto de funções, onde os parâmetros destes modelos são usados para caracterizar as diferentes texturas da imagem, e consequentemente possibilitar a sua segmentação. O problema principal a ser resolvido nestes métodos, é conseguir estimar os parâmetros destes modelos e também escolher o modelo mais adequado para uma determinada textura (ZHANG et al, 2002).

Os métodos desta abordagem baseada na utilização de modelos, podem ser utilizados para uma ampla variedade de texturas, sendo capazes geralmente, de segmentar imagens contendo texturas mais complexas e com comportamentos aleatórios. Comparado com as abordagens anteriormente discutidas, estes métodos são os que oferecem os resultados mais precisos na tarefa de segmentação por textura (BALAN, 2003).

Na abordagem baseada em modelos, os métodos podem ser supervisionados ou não supervisionados (BALAN, 2003). Nos métodos supervisionados, os parâmetros dos modelos são calculados a partir de um conjunto de treinamento obtido a partir de imagens previamente segmentadas e similares a imagem que se deseja segmentar. Já nos métodos não-supervisionados os parâmetros precisam ser estimados durante a execução do processo de segmentação, apenas através da imagem de entrada. O esquema não supervisionado é frequentemente necessário em muitas aplicações onde não existe um conjunto de treinamento disponível, como em sistemas autônomos.

As técnicas da abordagem não supervisionada são as que mais se aproximam da proposta deste trabalho, e geralmente o número de classes para a segmentação deve ser conhecido *a priori*, pois ele é necessário no processo de segmentação.

Cada um dos métodos desta abordagem baseada em modelos utiliza um determinado tipo de modelo para caracterizar as texturas. Alguns dos modelos mais utilizados incluem: o modelo de Markov, modelo de Gibs, modelo de Isiug, modelo de Piekard, modelo de Potts, modelo Wold, modelos SAR e RISAR, filtros de Gabor, modelo pirâmide orientável e a transformada *Wavelet*. Um breve estudo acerca de cada um destes modelos pode ser encontrado em (ZHANG et al, 2002) e (ROSHOLM, 1997, *apud* BALAN, 2003).

Grande parte dos métodos da abordagem baseada em modelos, utilizam além do modelo para caracterizar as texturas, alguns algoritmos para calcular os parâmetros do modelo utilizado. Um método que funciona desta maneira, é o método EM/MPM que utiliza o modelo de Markov, e é um dos métodos mais utilizados na segmentação de texturas devido ao seu desempenho. Em (BALAN, 2003) pode ser encontrado um estudo mais aprofundado sobre o método EM/MPM.

Uma das desvantagens dos métodos da abordagem baseada em modelos, é que muitos deles demandam um esforço computacional significativo para a segmentação, assim como pode ser observado por exemplo, em (BALAN, 2003). Outra desvantagem é que a grande maioria destes métodos precisam ter alguns parâmetros configurados corretamente para determinados tipos de textura, caso contrário não são obtidos bons resultados na segmentação, sendo necessário em alguns casos adicionar técnicas extras ao método para melhorar o seu desempenho, tudo isto também pode ser observado em (BALAN, 2003).

2.4 – Considerações Finais

Em cada uma das técnicas de segmentação discutidas neste capítulo, pode ser observada a presença de um fator comum entre elas, que é um meio de fazer separações entre os pixels da imagem, através da atribuição de um rótulo para caracterizar cada pixel, e consequentemente, segmentar a imagem de acordo com estes rótulos. Cada técnica usa uma forma diferente ao gerar rótulos para os pixels e fazer a sua classificação. A proposta deste trabalho é utilizar os algoritmos SIFT e SURF na rotulação dos pixels, através da classificação de sub-imagens de acordo com as diferentes texturas da imagem. Isto é possível pois uma sub-imagem nada mais é do que um conjunto de pixels, assim a classificação atribuída a uma sub-imagem, pode ser atribuída também aos pixels que a compõe. Este processo de classificação ou rotulação dos pixels é necessário para uma possível segmentação da imagem.

É possível observar que as técnicas de segmentação baseadas em texturas se aproximam bastante da proposta deste trabalho, especialmente as técnicas da abordagem baseada em modelos, pois elas lidam melhor com texturas que apresentam padrões aleatórios e complexos. As técnicas da abordagem estatística têm um aspecto bem semelhante ao deste trabalho, que está relacionado à etapa de extração de características, onde em sua metodologia podem ser utilizados os momentos estatísticos ou as matrizes de co-ocorrência, enquanto que neste trabalho a proposta é usar os algoritmos SIFT e SURF para extrair as características das diferentes texturas de uma imagem.

CAPÍTULO 3 – ALGORITMOS DETECTORES E DESCRITORES DE PONTOS CHAVE

Ao longo do tempo foram desenvolvidos vários algoritmos com a capacidade de extrair pontos chave de uma imagem para caracterizá-la. Estes pontos chave, são pontos com localização bem definida, tipicamente associados com significativas variações de propriedades como intensidade, cor e textura na imagem, possuindo por isso, grande potencial de caracterização (TUYTELAARS et al, 2008). Nestes algoritmos, a caracterização da imagem é feita através da descrição destes pontos chaves, também chamados de pontos de interesse. Existem muitas aplicações para estes algoritmos, como por exemplo, a detecção e reconhecimento de objetos em imagens, recuperação e registro de imagens, calibração de câmera, entre outras (BAY et al, 2006).

Existem alguns algoritmos que fazem apenas a detecção dos pontos de interesse na imagem, como por exemplo os algoritmos FAST (ROSTEN et al, 2006) e MSER (NISTÉR et al, 2008). Há outros que foram desenvolvidos apenas para fazer a descrição dos pontos de interesse, como os algoritmos FREAK (ALAHI et al, 2012) e BRIEF (CALONDER et al, 2010). Finalmente, há também vários algoritmos que são capazes de fazer a detecção, e também a descrição dos pontos de interesse em uma imagem, alguns exemplos são: BRISK (LEUTENEGGER et al, 2011), KAZE (ALCANTARILLA et al, 2012), ORB (RUBLEE et al, 2011), SIFT (LOWE, 2004) e SURF (BAY et al, 2006).

Na extração dos pontos de interesse da imagem, cada algoritmo utiliza uma técnica diferente, influenciando muito na eficiência e na quantidade de pontos detectados. A descrição de cada ponto de interesse é feita para que seja possível fazer o seu reconhecimento em outra imagem. Para fazer esta descrição dos pontos de interesse, é feita uma extração de características de cada ponto. Cada algoritmo faz a extração de características dos pontos chave de uma forma distinta, e isso influencia diretamente a eficiência e a velocidade do reconhecimento dos pontos de interesse.

Os algoritmos SIFT e SURF foram e ainda são bem reconhecidos por seu desempenho (MIKOLAJCZYK et al, 2005), principalmente devido ao fato de que as características dos pontos chave extraídas por eles são invariantes em relação a escala, translação e rotação, além de também possuírem um bom grau de resistência a variações de iluminação, deformações, ruídos e mudanças no ponto de visão (BAUER et al, 2007).

Devido as suas características, os algoritmos SIFT e SURF foram selecionados para o desenvolvimento deste trabalho, e a seguir será apresentado um estudo feito sobre cada um destes algoritmos.

3.1 – Algoritmo SIFT

O algoritmo SIFT (*Scale Invariant Features Transform*) foi proposto por David Lowe em 1999, e foi desenvolvido para ser capaz de detectar, descrever e reconhecer pontos de interesse em imagens, gerando um vetor descritor para cada ponto de interesse (LOWE, 1999).

Como pode ser observado em (LOWE, 2004), para fazer a detecção e a descrição dos pontos de interesse da imagem, o algoritmo SIFT segue 4 etapas:

1. Detecção de extremos no espaço de escalas.
2. Localização precisa dos pontos chave.
3. Definição de orientação.
4. Descrição dos pontos chave.

É importante salientar que o algoritmo SIFT não utiliza nenhuma informação relacionada as cores da imagem, mas usa apenas o nível de cinza dos pixels em suas etapas.

3.1.1 – Detecção de extremos no espaço de escalas

Nesta primeira etapa do algoritmo, o objetivo é encontrar os potenciais pontos chave, sendo necessário para isso, procurar por toda a imagem, e em todas as escalas possíveis, através de uma função contínua conhecida como espaço de escalas. O espaço de escalas de uma imagem é montado a partir de imagens representadas pela função $\boldsymbol{L}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\sigma})$, que são produzidas através da aplicação de um filtro Gaussiano $\boldsymbol{G}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\sigma})$ em uma imagem de entrada $\boldsymbol{I}(\boldsymbol{x}, \boldsymbol{y})$ assim como é mostrado na seguinte equação:

$$\boldsymbol{L(x, y, \sigma)} \;=\; \boldsymbol{G(x, y, \sigma) * I(x, y)}$$

Equação 3.1

Onde ***** é a operação de convolução em $\boldsymbol{x}$ e $\boldsymbol{y}$, a letra grega sigma $\boldsymbol{\sigma}$ representa a escala e:

$$\boldsymbol{G(x, y, \sigma)} \;=\; \frac{1}{2\boldsymbol{\pi\sigma^2}}\, \boldsymbol{e^{-(x^2 + y^2) \,/\, 2\sigma^2}}$$

Equação 3.2

O espaço de escalas é dividido em várias oitavas, que são conjuntos de imagens no espaço de escalas. Para cada oitava do espaço de escalas, o filtro Gaussiano é aplicado repetidamente na imagem inicial para produzir um conjunto de imagens $\boldsymbol{L(x, y, \sigma)}$ separadas por um valor constante $\boldsymbol{k}$, formando um espaço de escalas representado pela pilha de imagens na coluna esquerda da Figura 3.1.

Para fazer a detecção das localizações estáveis dos pontos chave no espaço de escalas, o algoritmo utiliza os extremos de um outro espaço de escalas, constituído de imagens denotadas por $\boldsymbol{D(x, y, \sigma)}$, onde estas imagens são geradas através do filtro DoG (*Diference of Gaussian*). O filtro DoG é uma aproximação do filtro LoG (*Laplacian of Gaussian*), que ao ser aplicado em uma imagem, facilita o encontro dos pontos chave. Uma imagem $\boldsymbol{D(x, y, \sigma)}$ pode ser computada através da seguinte equação:

$$\begin{aligned} \boldsymbol{D(x, y, \sigma)} \;&=\; \boldsymbol{(G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y)} \\ &=\; \boldsymbol{L(x, y, k\sigma) - L(x, y, \sigma)} \end{aligned}$$

Equação 3.3

Um exemplo da construção do espaço de escalas das imagens $\boldsymbol{D(x, y, \sigma)}$ através da aplicação da Equação 3.3, é mostrado na Figura 3.1. Duas imagens consecutivas com filtro Gaussiano do espaço de escalas à esquerda, são subtraídas para produzir as imagens $\boldsymbol{D(x, y, \sigma)}$ na coluna direita, conforme a Equação 3.3.

Após o processamento de cada oitava, a imagem com filtro Gaussiano que possui duas vezes o valor inicial de $\boldsymbol{\sigma}$, tem sua resolução reduzida por um fator de 2, ou seja, é reduzida pela metade, e então todo o processo é repetido a partir dela, para gerar uma nova oitava. Todo este processo gera uma pirâmide de escalas.

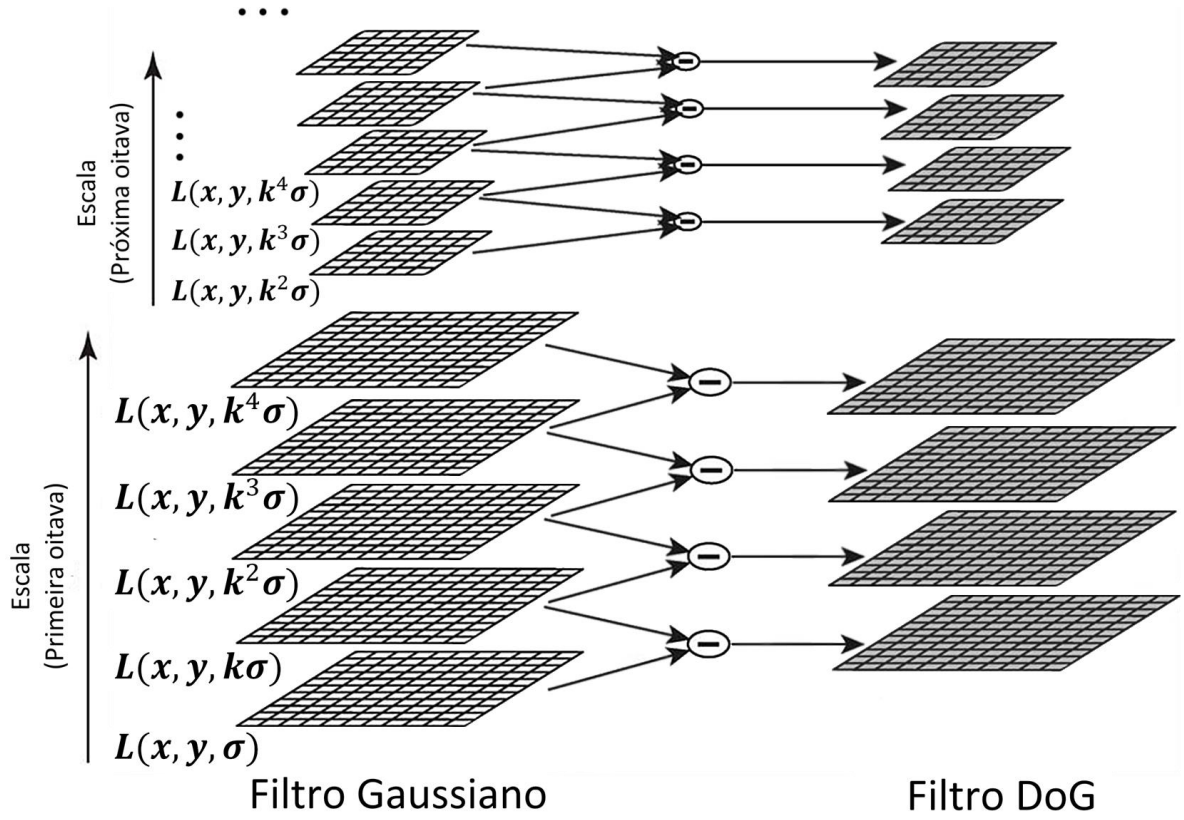

Figura 3.1 – Construção do espaço de escalas no algoritmo SIFT. Adaptado de: (LOWE, 2004).

Neste ponto, se inicializa a busca pelos pontos chave, localizando os extremos máximos e mínimos das imagens $\boldsymbol{D(x, y, \sigma)}$. Para cada pixel de uma imagem do espaço de escalas das imagens $\boldsymbol{D(x, y, \sigma)}$, é feita a comparação dele com os seus oito vizinhos na imagem atual, e também com os nove vizinhos da imagem acima e da imagem abaixo. Se o pixel examinado for o maior entre todos os seus 26 vizinhos, ele é considerado um extremo máximo, caso ele seja o menor entre todos os seus vizinhos, então ele é considerado como um extremo mínimo. Um pixel é selecionado como um potencial ponto chave, se ele for um extremo máximo ou um extremo mínimo no espaço de escalas das imagens $\boldsymbol{D(x, y, \sigma)}$. A Figura 3.2 ilustra este processo representando o pixel de amostra por um '×', e os seus vizinhos por círculos.

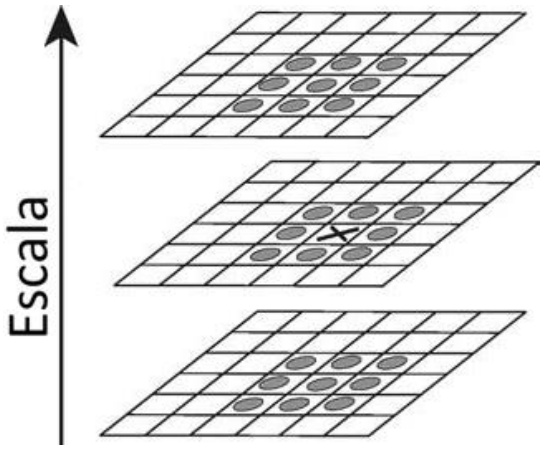

Figura 3.2 – Detecção de extremos no algoritmo SIFT. Adaptado de: (LOWE, 2004).

3.1.2 – Localização precisa dos pontos chave

Uma vez que as localizações dos potenciais pontos chave são encontradas, alguns pontos chaves são descartados para melhorar o reconhecimento e a resistência a ruídos.

Pontos chave que apresentam baixo contraste são eliminados pelo algoritmo. Para isso deve ser feita a determinação da localização interpolada dos extremos, usando a expansão de Taylor no espaço de escalas das imagens $\boldsymbol{D}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\sigma})$, sendo que este espaço de escalas sofre um deslocamento para que a origem da expansão seja definida no ponto chave de amostra (BROWN et al, 2002). Assim, a expansão de Taylor utilizada, é dada por:

$$\boldsymbol{D}(\boldsymbol{x}) \;=\; \boldsymbol{D} + \frac{\boldsymbol{\partial D^T}}{\boldsymbol{\partial x}}\boldsymbol{x} + \frac{1}{2}\boldsymbol{x^T}\frac{\boldsymbol{\partial^2 D}}{\boldsymbol{\partial x^2}}\boldsymbol{x}$$

Equação 3.4

Onde $\boldsymbol{D}$ é o valor de $\boldsymbol{D}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\sigma})$ no ponto de amostra, $\boldsymbol{x} = (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\sigma})^{\boldsymbol{T}}$ é o vetor que representa o deslocamento a partir deste ponto, e $\boldsymbol{D}(\boldsymbol{x})$ é uma aproximação do valor interpolado de $\boldsymbol{D}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\sigma})$ para um ponto transladado pelo deslocamento $\boldsymbol{x}$ (GONZÁLES, 2010). A localização do extremo, aqui denotada pelo vetor $\boldsymbol{x'}$, é determinada ao se calcular a derivada da Equação 3.4 em relação a $\boldsymbol{x}$ e igualando o seu resultado a 0, o que resulta na seguinte equação:

$$\boldsymbol{x'} = -\frac{\boldsymbol{\partial^2 D}}{\boldsymbol{\partial x^2}}^{-1}\frac{\boldsymbol{\partial D}}{\boldsymbol{\partial x}}$$

Equação 3.5

Se o valor de $\boldsymbol{x'}$ for maior que 0.5 em qualquer uma de suas dimensões, então isso significa que o extremo se encontra mais perto de outro ponto de amostra. Nesse caso, o ponto de amostra é mudado através da adição de $\boldsymbol{x'}$ a sua localização, e então, a sua localização interpolada deve ser recalculada, para que a localização estimada do extremo seja obtida.

A resposta da função $\boldsymbol{D}(\boldsymbol{x'})$ é utilizada para descartar extremos com baixos níveis de contraste, e ela é dada pela Equação 3.6, que foi obtida ao substituir a Equação 3.5 na Equação 3.4, resultando em:

$$\boldsymbol{D}(\boldsymbol{x'}) = \boldsymbol{D} + \frac{1}{2}\frac{\boldsymbol{\partial D^T}}{\boldsymbol{\partial x}}\boldsymbol{x'}$$

Equação 3.6

Assim sendo, todos os extremos que apresentem um valor de $\boldsymbol{D}(\boldsymbol{x'})$ menor que um determinado limiar $\boldsymbol{ct}$, são descartados. Em (LOWE, 2004), é utilizado para o limiar $\boldsymbol{ct}$ um valor de 0,03, considerando que os tons de cinza dos pixels da imagem sejam normalizados com valores entre 0 e 1.

Além de eliminar os pontos chave com baixo contraste, o algoritmo também descarta pontos chave mal localizados ao longo de arestas. Um ponto chave mal definido em uma imagem com filtro Diferença de Gaussiano, terá uma maior magnitude de curvatura principal ao longo da aresta, mas terá uma pequena magnitude de curvatura na direção perpendicular. A magnitude destas curvaturas principais pode ser calculada a partir de uma matriz Hessiana 2 × 2, aqui denotada por $\boldsymbol{H}$. Esta matriz é computada dentro da mesma localização e escala do ponto chave:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{D_{xx}} & \boldsymbol{D_{xy}} \\ \boldsymbol{D_{xy}} & \boldsymbol{D_{yy}} \end{bmatrix}$$

Equação 3.7

Assim, como é explicado em (GONZÁLES, 2010), as derivadas desta matriz são calculadas através de aproximações de diferenças entre pontos vizinhos do ponto chave de amostra:

$$\boldsymbol{D_{xx}} = \boldsymbol{D}(\boldsymbol{x}+1, \boldsymbol{y}, \boldsymbol{\sigma}) - 2\boldsymbol{D}(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\sigma}) + \boldsymbol{D}(\boldsymbol{x}-1, \boldsymbol{y}, \boldsymbol{\sigma})$$

Equação 3.8

$$\boldsymbol{D_{yy} = D(x, y + 1, \sigma) - 2D(x, y, \sigma) + D(x, y - 1, \sigma)}$$

Equação 3.9

$$\boldsymbol{D_{xy}} = \frac{\boldsymbol{D(x - 1, y + 1, \sigma) - D(x + 1, y + 1, \sigma) + D(x + 1, y - 1, \sigma) - D(x - 1, y - 1, \sigma)}}{4}$$

Equação 3.10

Os autovalores de $\boldsymbol{H}$ são proporcionais as curvaturas principais, tornando possível mensurar através deles, a magnitude das curvaturas principais. Considerando $\boldsymbol{\alpha}$ o autovalor de maior magnitude e $\boldsymbol{\beta}$ o de menor, a soma dos autovalores é calculada pelo traço de $\boldsymbol{H}$ e o produto dos autovalores é calculado pelo determinante de $\boldsymbol{H}$:

$$\boldsymbol{Tr(H) = D_{xx} + D_{yy} = \alpha + \beta}$$

Equação 3.11

$$\boldsymbol{Det(H) = D_{xx}D_{yy} - \left(D_{xy}\right)^2 = \alpha\beta}$$

Equação 3.12

Quando o determinante resulta em um valor negativo, isto significa que as curvaturas possuem sinais diferentes, e por isso o ponto de amostra é descartado como não sendo um extremo.

Considerando $\boldsymbol{r}$ como sendo a relação entre os autovalores $\boldsymbol{\alpha}$ e $\boldsymbol{\beta}$, fazendo com que $\boldsymbol{\alpha = r\beta}$, então pode se concluir que:

$$\frac{\boldsymbol{Tr(H)^2}}{\boldsymbol{Det(H)}} = \frac{\boldsymbol{(\alpha + \beta)^2}}{\boldsymbol{\alpha\beta}} = \frac{\boldsymbol{(r\beta + \beta)^2}}{\boldsymbol{r\beta^2}} = \frac{\boldsymbol{(r + 1)^2}}{\boldsymbol{r}}$$

Equação 3.13

Assim, são eliminados todos os pontos chave que não satisfazerem a seguinte condição:

$$\frac{\boldsymbol{Tr(H)^2}}{\boldsymbol{Det(H)}} < \frac{\boldsymbol{(r\ +\ 1)^2}}{\boldsymbol{r}}$$

Equação 3.14

Desta forma, o algoritmo SIFT elimina pontos chave que tenham uma razão entre as curvaturas principais maior que $\boldsymbol{r}$. Em (LOWE, 2004), o algoritmo usa um valor de $\boldsymbol{r = 10}$.

3.1.3 – Definição de orientação

O próximo passo no algoritmo, é determinar uma orientação consistente para cada ponto chave baseada nas propriedades locais da imagem. Assim o descritor de cada ponto chave pode ser representado em relação a esta orientação alcançando invariância a rotação, ou seja, mesmo que a imagem seja rotacionada, os pontos chave serão identificados. A ideia é usar a escala do ponto chave para selecionar uma imagem $\boldsymbol{L}$ suavizada com o filtro Gaussiano, que esteja em uma escala mais próxima, para que todos os cálculos realizados sejam invariantes a escala. Assim, para cada ponto de amostra $\boldsymbol{L}(\boldsymbol{x}, \boldsymbol{y})$ da imagem $\boldsymbol{L}$, na escala selecionada através de um ponto chave, é pré-computada a orientação do gradiente $\boldsymbol{\theta}(\boldsymbol{x}, \boldsymbol{y})$, e sua magnitude $\boldsymbol{m}(\boldsymbol{x}, \boldsymbol{y})$, com base nas diferenças entre os pixels:

$$\boldsymbol{m}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{L}(\boldsymbol{x}+1, \boldsymbol{y}) - \boldsymbol{L}(\boldsymbol{x}-1, \boldsymbol{y}))^2 + (\boldsymbol{L}(\boldsymbol{x}, \boldsymbol{y}+1) - \boldsymbol{L}(\boldsymbol{x}, \boldsymbol{y}-1))^2}$$

Equação 3.15

$$\boldsymbol{\theta}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{tan}^{-1}((\boldsymbol{L}(\boldsymbol{x}, \boldsymbol{y}+1) - \boldsymbol{L}(\boldsymbol{x}, \boldsymbol{y}-1))/(\boldsymbol{L}(\boldsymbol{x}+1, \boldsymbol{y}) - \boldsymbol{L}(\boldsymbol{x}-1, \boldsymbol{y})))$$

Equação 3.16

Depois, um histograma de orientações é formado a partir das orientações do gradiente dos pontos de amostra dentro de uma região ao redor do ponto chave, conforme ilustrado no lado esquerdo da Figura 3.3 na próxima seção. O histograma de orientações possui 36 campos cobrindo a faixa de 360 graus de orientações. Cada ponto de amostra adicionado ao histograma é ponderado através da sua magnitude do gradiente e também através de uma janela circular com o filtro Gaussiano, onde o valor de $\boldsymbol{\sigma}$ é equivalente a 1,5 vezes a escala do ponto chave. Esta janela circular é ilustrada no lado esquerdo da Figura 3.3, e ela é usada para dar menos ênfase aos gradientes distantes do ponto chave, já que estes são os mais afetados por possíveis erros de registro.

Os picos identificados no histograma de orientações correspondem as direções dominantes dos gradientes locais. O maior pico no histograma é detectado, e outros picos locais que estejam dentro de 80% do valor do maior pico, são usados para criar outros pontos chave, onde a orientação de cada um deles é a referente ao pico do histograma utilizado. Portanto, para locais com vários picos de magnitude similar, haverá múltiplos pontos chave criados na mesma localização e escala, mas com diferentes orientações.

3.1.4 – Descrição dos pontos chave

Baseado no sistema biológico da visão, o método usado pelo algoritmo SIFT para descrever um ponto chave, pode ser ilustrado na Figura 3.3, onde é mostrado como é feito o cálculo dos valores do vetor descritor de um ponto chave.

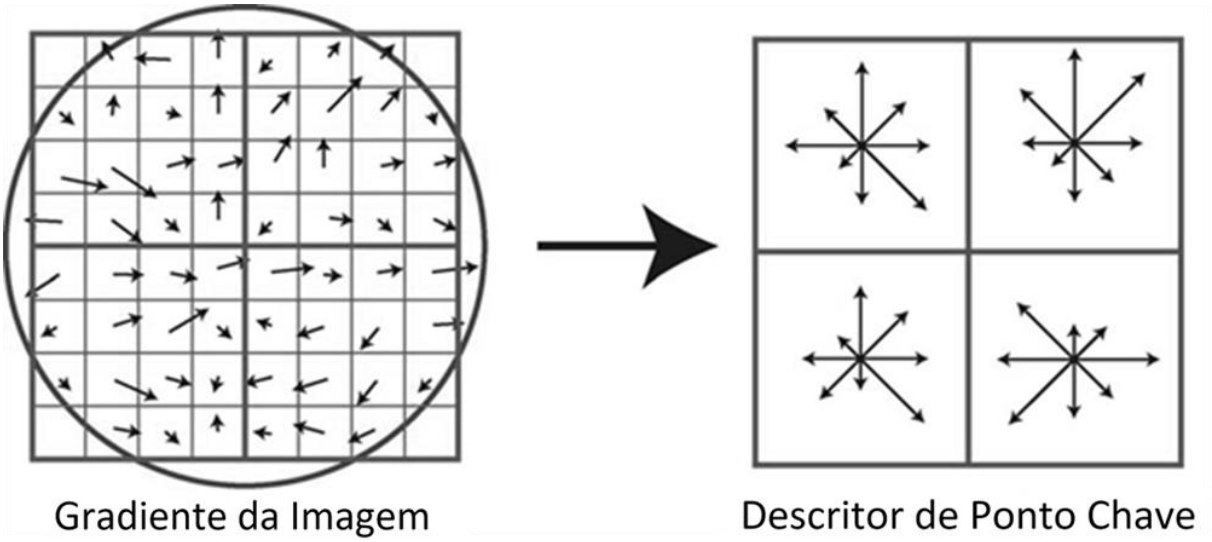

Figura 3.3 – Processo de descrição dos pontos chave no algoritmo SIFT. Adaptado de: (LOWE, 2004).

Primeiramente, as orientações e magnitudes do gradiente em uma região ao redor do ponto chave são calculadas. Para aumento da eficiência do algoritmo, os gradientes são pré-calculados para todos os níveis da pirâmide de escalas, conforme descrito na Seção 3.1.3. A fim de atingir a invariância a orientação, as coordenadas do descritor e das orientações do gradiente são rotacionadas em relação à orientação do ponto chave. As orientações e magnitudes do gradiente são ilustrados com pequenas setas em cada local de amostra no lado esquerdo da Figura 3.3.

O descritor do ponto chave é ilustrado no lado direito da Figura 3.3. As orientações e magnitudes do gradiente contidas nas 4 sub-regiões de 4 × 4, ilustradas à esquerda da Figura 3.3, são utilizadas para formar 4 histogramas de orientação. No lado direito da Figura 3.3 são mostradas 8 setas em diferentes direções para cada histograma de orientação, onde o tamanho de cada seta corresponde à uma entrada do histograma de orientação, que é equivalente à soma das magnitudes do gradiente perto daquela direção na região de amostra.

Assim, o descritor é formado por um vetor contendo os valores de todas as entradas dos histogramas de orientação. A Figura 3.3 apresenta um vetor de histogramas de orientação de 2 × 2 com 8 campos de orientação em cada um, calculado a partir de um conjunto de amostras de 8 × 8. No algoritmo SIFT, é utilizado um vetor de histogramas de 4 × 4 com 8 campos de orientação em cada um, calculado a partir de um vetor de amostras de 16 × 16. Portanto, o algoritmo gera um vetor de características com dimensões 4 × 4 × 8, o que resulta em um vetor descritor com 128 elementos para cada ponto chave.

Em um último passo, o vetor de características é modificado para reduzir os efeitos de mudanças na iluminação. Para isso, primeiramente o vetor é normalizado em uma unidade de comprimento, depois os valores do vetor são limitados a não serem maior do que 0,2, e então, o vetor de características é novamente normalizado em uma unidade de comprimento.

3.2 – Algoritmo SURF

O algoritmo SURF (*Speeded-Up Robust Features*) foi proposto por Herbert Bay (BAY et al 2006), e assim como o algoritmo SIFT, ele foi desenvolvido para ser capaz de detectar, descrever e reconhecer pontos de interesse em imagens, gerando um vetor descritor para cada ponto de interesse (BAY et al, 2006).

Como pode ser observado em (BAY et al, 2006), para detectar e a descrever os pontos de interesse da imagem, o algoritmo SURF segue 2 etapas:

1. Detecção de pontos chave.
2. Descrição dos pontos chave.

É importante salientar que assim como o algoritmo SIFT, o algoritmo SURF não utiliza nenhuma informação relacionada as cores da imagem, mas usa apenas o nível de cinza dos pixels em suas etapas.

3.2.1 – Detecção de pontos chave

Para fazer a detecção dos pontos de interesse, o algoritmo SURF utiliza uma aproximação da matriz Hessiana. O algoritmo também faz o uso de imagens integrais para reduzir drasticamente o tempo de computação, pois elas possibilitam uma rápida computação de convoluções baseadas em filtros de caixa.

Um campo qualquer de uma imagem integral $\boldsymbol{I_{\Sigma}(x')}$, localizado em $\boldsymbol{x' = (x, y)}$ representa a soma de todos os pixels da imagem de entrada $\boldsymbol{I}$ dentro de uma região retangular formada pelas coordenadas da origem da imagem $\boldsymbol{I}$ e por $\boldsymbol{x'}$.

$$\boldsymbol{I_{\Sigma}(x') = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j)}$$

Equação 3.17

Uma vez que a imagem integral é calculada, é preciso apenas três adições para calcular a soma das intensidades de qualquer área retangular como a que é mostrado na Figura 3.4. Desta forma, o tempo de cálculo é independente do tamanho da área. Isto é muito importante para o algoritmo SURF, pois ele utiliza filtros de grandes tamanhos.

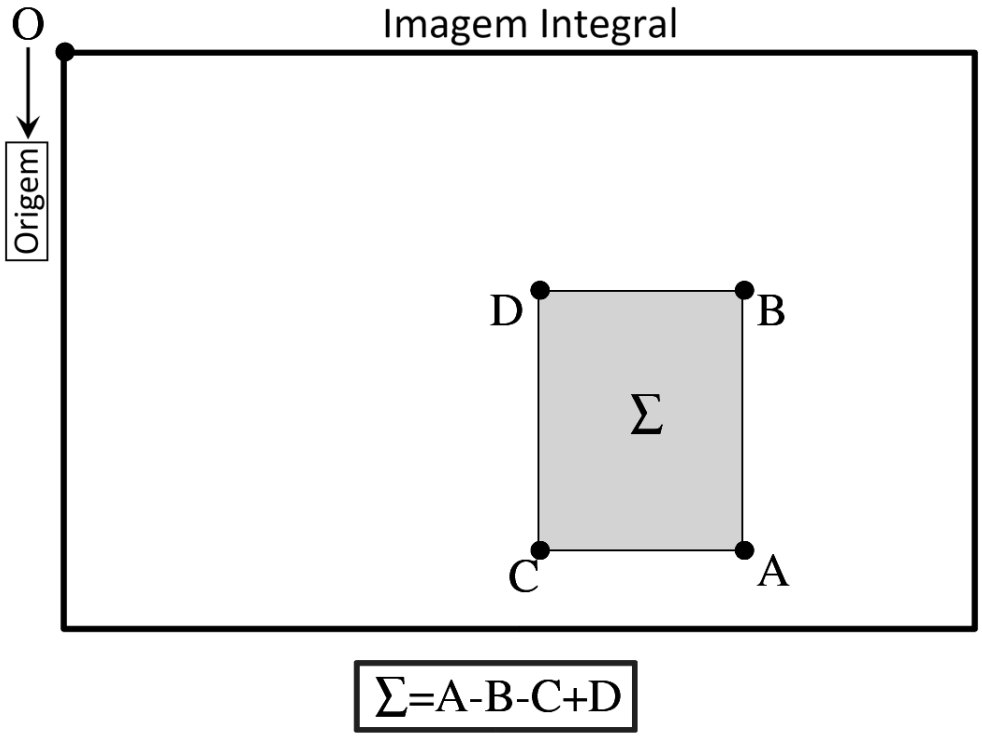

Figura 3.4 – Soma das intensidades de uma região retangular em uma imagem integral. Adaptado de: (BAY et al, 2006).

A detecção de pontos chave baseada na matriz Hessiana, funciona pela detecção de cumes em locais da imagem onde o determinante é máximo.

Assim, dado um ponto $\boldsymbol{x' = (x, y)}$ em uma imagem $\boldsymbol{I}$, a matriz Hessiana $\boldsymbol{H(x', \sigma)}$ no ponto $\boldsymbol{x'}$ e em uma escala $\boldsymbol{\sigma}$, é definida como:

$$\boldsymbol{H(x', \sigma)} = \begin{bmatrix} \boldsymbol{L_{xx}(x', \sigma)} \; \boldsymbol{L_{xy}(x', \sigma)} \\ \boldsymbol{L_{xy}(x', \sigma)} \; \boldsymbol{L_{yy}(x', \sigma)} \end{bmatrix}$$

Equação 3.18

Onde $\boldsymbol{L_{xx}(x', \sigma)}$ é a convolução da derivada de segunda ordem da função Gaussiana $\frac{\partial^2}{\partial x^2}\boldsymbol{g(\sigma)}$ com a Imagem $\boldsymbol{I}$ no ponto $\boldsymbol{x'}$, e semelhantemente o mesmo é válido para $\boldsymbol{L_{xy}(x', \sigma)}$ e $\boldsymbol{L_{yy}(x', \sigma)}$.

Para aumentar o desempenho, o algoritmo faz uso de aproximações das derivadas de segunda ordem da função Gaussiana presentes na matriz Hessiana. A Figura 3.5 mostra os filtros de caixa baseados nas derivadas de segunda ordem da função Gaussiana, enquanto que na Figura 3.6, são mostrados os filtros de caixa baseados nas aproximações das derivadas de segunda ordem da função Gaussiana, nestas figuras, as regiões de cor cinza representam o valor 0, regiões claras representam um valor positivo, e regiões escuras um valor negativo.

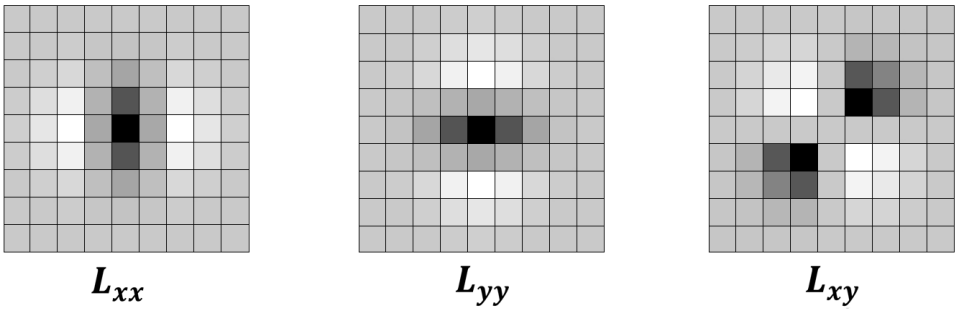

Figura 3.5 – Filtros baseados nas derivadas de segunda ordem da função Gaussiana. Adaptado de: (BAY et al, 2006).

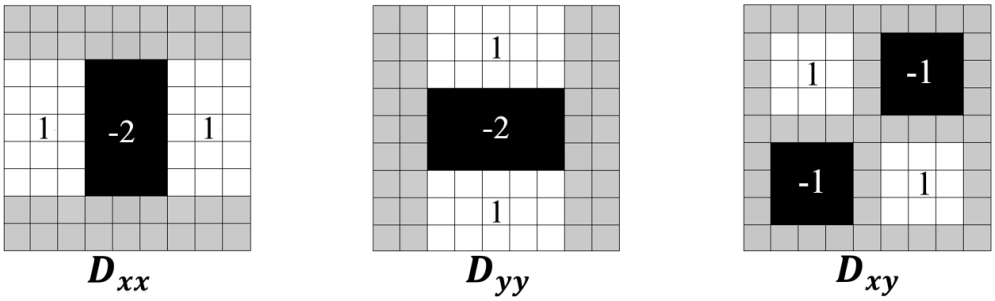

Figura 3.6 – Aproximações das derivadas de segunda ordem da função Gaussiana. Adaptado de: (BAY et al, 2006).

Os filtros de caixa 9 × 9 ilustrados na Figura 3.6 são aproximações baseadas em uma escala $\boldsymbol{\sigma} = 1{,}2$, que representa a menor escala para computar os mapas de respostas dos cumes. Estas aproximações possuem um baixo custo computacional usando imagens integrais (BAY et al, 2006), e são denotadas por $\boldsymbol{D_{xx}}$, $\boldsymbol{D_{yy}}$ e $\boldsymbol{D_{xy}}$. Assim o determinante da matriz Hessiana aproximada, é dado por:

$$\boldsymbol{det(H_{aprox}) = D_{xx}D_{yy} - (wD_{xy})^2}$$

Equação 3.19

Onde $\boldsymbol{w} = 0{,}912\ldots \simeq 0{,}9$, é um fator constante usado para balancear a expressão do determinante da matriz Hessiana.

O determinante aproximado da matriz Hessiana representa a resposta de um cume na imagem em uma localização $\boldsymbol{x'}$, que é um possível ponto chave. Estas respostas são armazenadas em um mapa de respostas sobre diferentes escalas.

Os pontos chave são procurados em diferentes escalas para alcançar a invariância a escala. Assim como o algoritmo SIFT, o algoritmo SURF utiliza um espaço de escalas, no entanto, utiliza uma forma diferente para representá-lo.

Devido ao uso de filtros de caixa e imagens integrais, o algoritmo SURF precisa apenas aplicar através de convolução, os filtros de caixa com vários tamanhos diretamente na imagem original, utilizando exatamente a mesma velocidade para todos os filtros. Portanto, o espaço de escalas é gerado através do aumento da escala do tamanho do filtro de caixa que será convolucionado com a imagem (Figura 3.7). A saída da convolução da imagem original com os filtros 9 × 9, mostrados anteriormente na Figura 3.6, é considerada como a camada inicial de escala. As próximas camadas são obtidas filtrando esta camada inicial gradualmente com filtros maiores, levando em consideração o uso das imagens integrais e a estrutura específicas dos filtros de caixa da Figura 3.6.

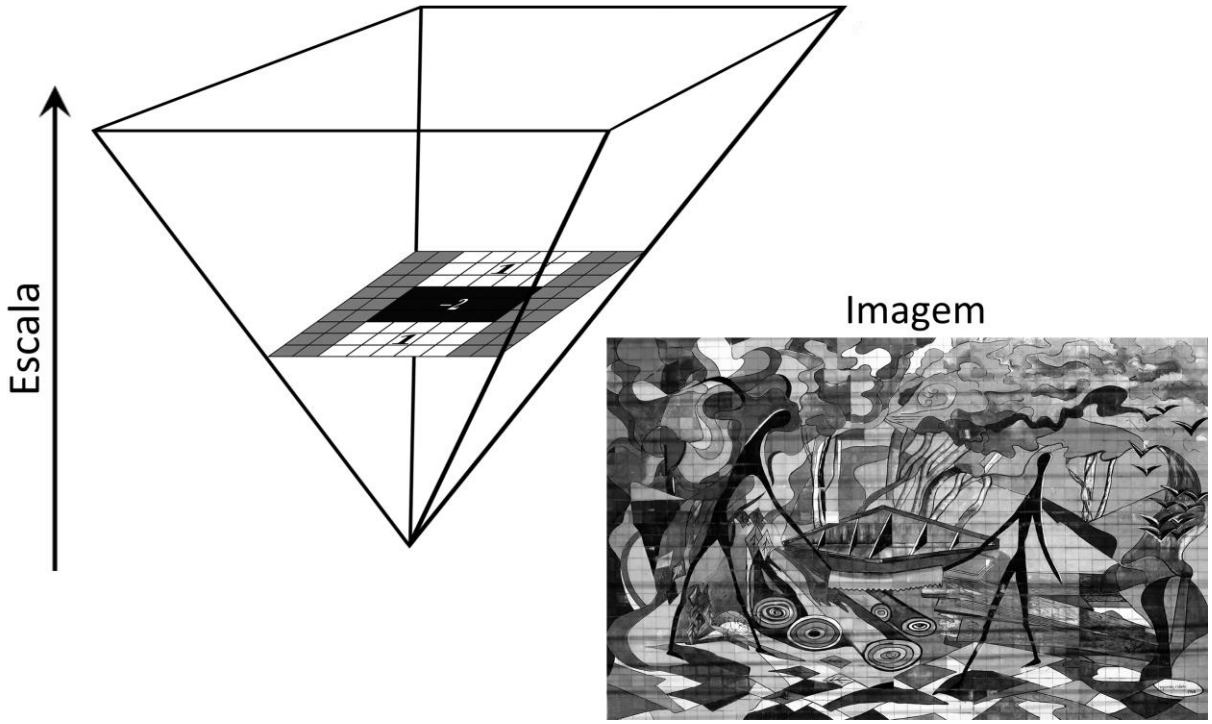

Figura 3.7 – Representação do espaço de escalas no algoritmo SURF. Adaptado de: (BAY et al, 2006).

O espaço de escalas é dividido em oitavas. Uma oitava é formada por uma série de respostas obtidas pela convolução da imagem de entrada com filtros de caixa de tamanho crescente.

A construção do espaço de escalas começa com a primeira oitava, que é formada pela aplicação dos filtros de 9 × 9, 15 × 15, 21 × 21 e 27 × 27, onde a partir do filtro 9 × 9 filtros maiores vão sendo gerados com um incremento de 6. Um processo semelhante acontece para as outras oitavas. Para cada nova oitava o valor do incremento utilizado para aumentar os filtros, é dobrado. Os tamanhos dos filtros da segunda oitava são 15, 27, 39 e 51. Uma terceira oitava é formada com o uso de filtros de tamanhos 27, 51, 75 e 99.

Caso o tamanho da imagem original seja ainda maior que os tamanhos de filtros citados, é utilizada então uma quarta oitava, usando filtros de tamanhos 51, 99, 147 e 195. A Figura 3.8 ilustra os tamanhos dos filtros para as três primeiras oitavas.

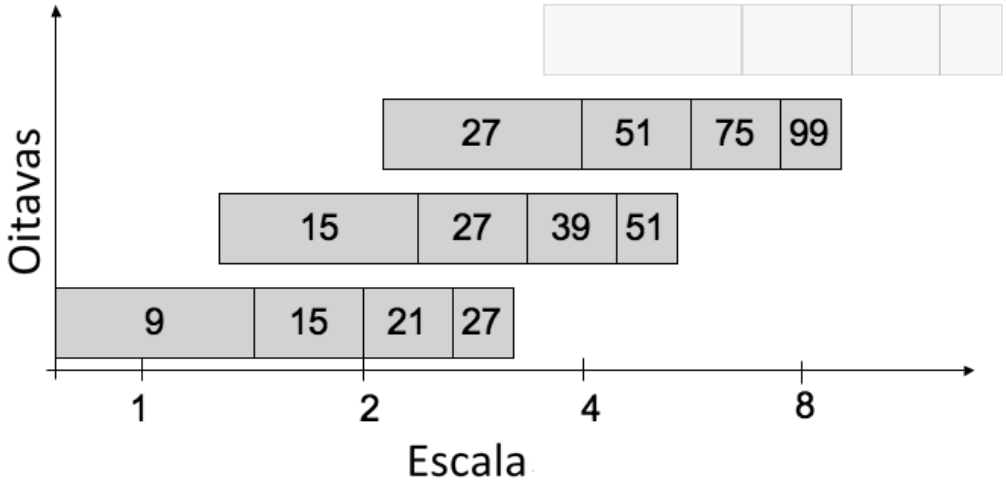

Figura 3.8 – Oitavas no espaço de escalas. Adaptado de: (BAY et al, 2006).

Para localizar os pontos chave na imagem e em todas as escalas, é aplicada uma supressão de determinantes não-máximos em uma vizinhança 3D de 3 × 3 × 3, fazendo com que a supressão seja feita na vizinhança localizada tanto espacialmente na imagem, quanto ao longo das escalas vizinhas acima e abaixo. A Figura 3.9 ilustra a vizinhança de um determinante da matriz Hessiana representado por um '×'. Esta supressão é realizada em todo o espaço de escalas, utilizando um método introduzido por Neubeck et al (2006), fazendo com que vários dos determinantes calculados da matriz Hessiana sejam eliminados, ficando apenas os determinantes máximos. Desta forma, quando um determinante máximo é detectado, ele é selecionado como um ponto de interesse naquela localização e escala, enquanto que todos os seus vizinhos são suprimidos.

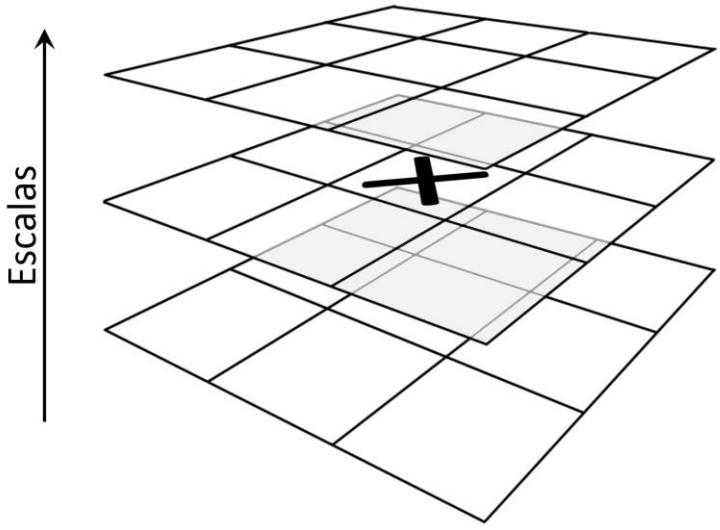

Figura 3.9 – Supressão de não-máximos. Fonte: Elaborada pelo autor.

Em um último passo, os determinantes máximos da matriz Hessiana são então interpolados com o método proposto em (BROWN et al, 2002), para melhorar a estabilidade dos pontos chave.

3.2.2 – Descrição dos pontos chave

Para fazer a descrição dos pontos chave, o primeiro passo do algoritmo, é a identificação da orientação dos pontos chave. Para isso, primeiramente são calculadas as respostas dos filtros das *wavelets* de Haar nas direções $\boldsymbol{x}$ e $\boldsymbol{y}$ dentro de uma vizinhança circular de raio igual à $6\boldsymbol{s}$ ao redor do ponto de interesse, sendo $\boldsymbol{s}$ a escala na qual o ponto de interesse foi detectado. Então, novamente o algoritmo usa as imagens integrais para uma rápida filtragem. Os filtros usados para o cálculo das respostas das *wavelets* de Haar são mostrados na Figura 3.10, sendo o filtro da esquerda correspondente a resposta na direção $\boldsymbol{x}$, e o da direita correspondente a resposta na direção $\boldsymbol{y}$. Ainda na Figura 3.10, as áreas escuras possuem peso igual à -1 e as áreas claras possuem peso igual à 1.

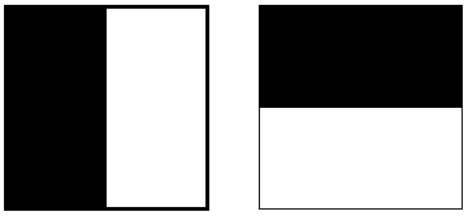

Figura 3.10 – Filtros para o cálculo das respostas das *wavelets* de Haar. Adaptado de: (BAY et al, 2006).

Uma vez que as respostas das *wavelets* são calculadas e suavizadas com um filtro Gaussiano ($\boldsymbol{\sigma = 2s}$) centrado na localização do ponto de interesse, as respostas são representadas como pontos em um espaço bidimensional, no qual o valor da resposta na direção $\boldsymbol{x}$ corresponde à abscissa do espaço, e o valor da resposta na direção $\boldsymbol{y}$ corresponde a ordenada. A orientação dominante é estimada calculando-se a soma de todas as respostas dentro de uma janela deslizante de orientação, com tamanho igual a $\boldsymbol{\pi}/3$, representada na Figura 3.11 pelo setor circular de cor cinza. As respostas na direção $\boldsymbol{x}$ são somadas, assim como as respostas na direção $\boldsymbol{y}$. Os resultados destas duas somas realizadas, produzem um vetor de orientação local. O vetor mais longo produzido entre todas as janelas deslizantes, define a orientação do ponto de interesse.

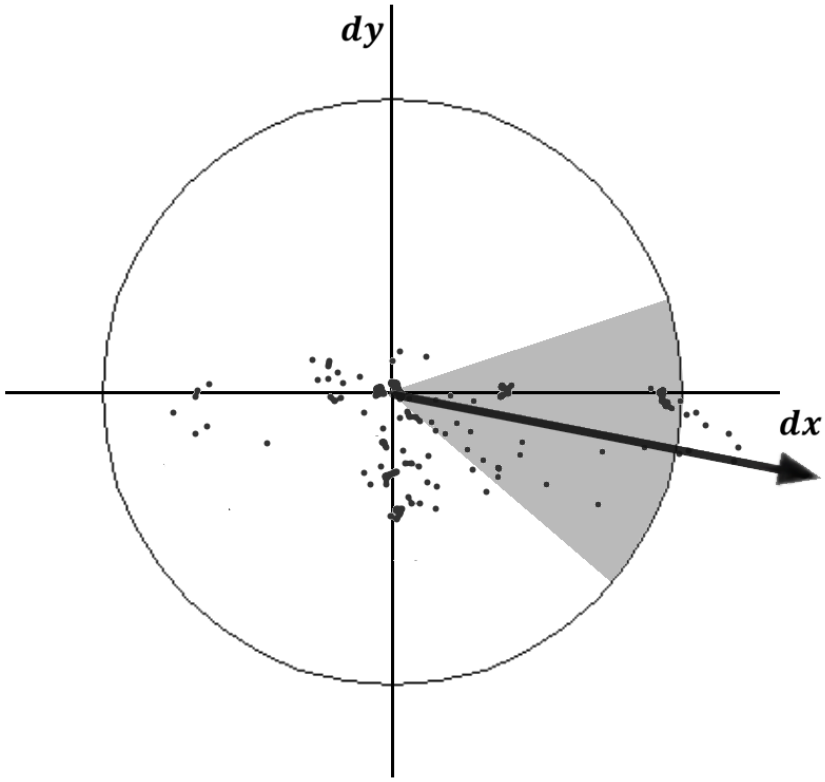

Figura 3.11 – Espaço bidimensional para a definição de orientação. Adaptado de: (BAY et al, 2006).

Para a extração do descritor, o primeiro passo consiste em construir uma região quadrada centrada ao redor do ponto de interesse e orientada ao longo da orientação selecionada para o ponto de interesse. O tamanho desta janela é de $20\boldsymbol{s}$, sendo $\boldsymbol{s}$ a escala do ponto de interesse. Exemplos destas regiões quadradas podem ser visualizados na Figura 3.12.

Figura 3.12 – Criação de janelas para a descrição dos pontos de interesse. Fonte: (BAY et al, 2006).

Depois, cada região desta é dividida em sub-regiões quadradas de mesmo tamanho, formando uma matriz de 4 × 4, totalizando 16 sub-regiões. Então para cada sub-região, as respostas das *wavelets* de Haar são calculadas em pontos regularmente espaçados por uma divisão de 5 × 5. Considere $\boldsymbol{dx}$ como sendo a resposta das *wavelets* de Haar na direção horizontal e $\boldsymbol{dy}$ a resposta na direção vertical para filtros de tamanho $2\boldsymbol{s}$. Nesta etapa, os termos "horizontal" e "vertical" são definidos de acordo com a orientação do ponto de interesse. Para aumentar a robustez a deformações geométricas e erros de localização, as respostas $\boldsymbol{dx}$ e $\boldsymbol{dy}$ são suavizadas por um filtro Gaussiano ($\boldsymbol{\sigma} = 3{,}3\boldsymbol{s}$), centrado no ponto de interesse.

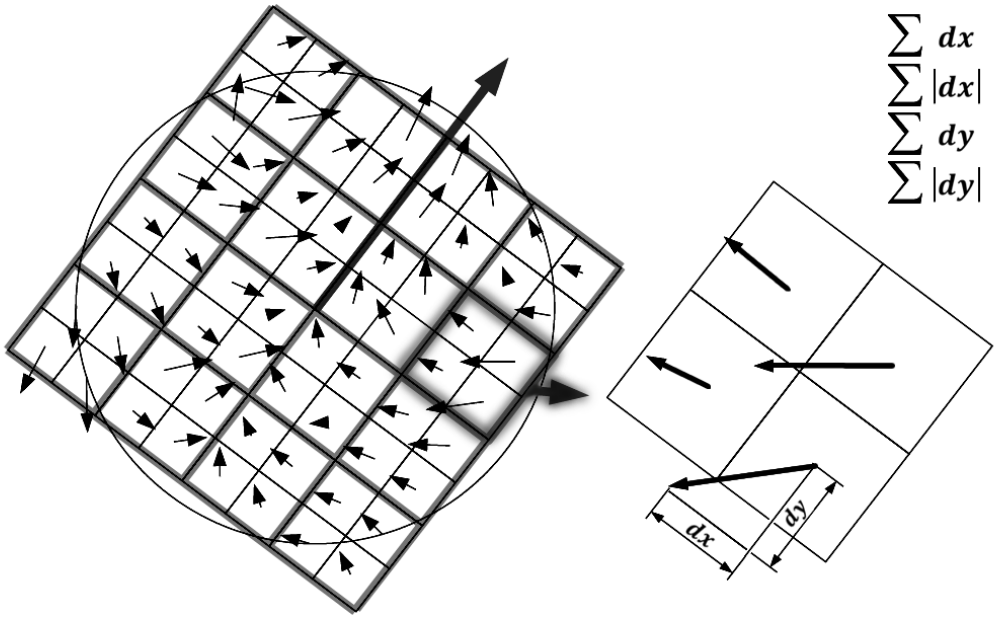

Figura 3.13 – Geração de vetor descritor no algoritmo SURF. Adaptado de: (BAY et al, 2006).

Então, as respostas das *wavelets* $\boldsymbol{dx}$ e $\boldsymbol{dy}$ são somadas em cada sub-região para formar um primeiro conjunto de entradas para o vetor descritor. Também são extraídas as somas dos valores absolutos das respostas, sendo representadas por $|\boldsymbol{dx}|$ e $|\boldsymbol{dy}|$. Por isso, cada sub-região possui um vetor descritor de tamanho 4 , denotado por $\boldsymbol{v} = (\sum \boldsymbol{dx}, \sum \boldsymbol{dy}, \sum |\boldsymbol{dx}|, \sum |\boldsymbol{dy}|)$ (Figura 3.13). Concatenando todos os vetores das 16 sub-regiões, isto resulta em um vetor descritor com 64 posições. Finalmente, para ser invariante ao contraste, o vetor descritor é transformado em um vetor unidade.

CAPÍTULO 4 – REDES NEURAIS ARTIFICIAIS

Dentro da inteligência artificial, as redes neurais artificiais (RNAs), são sistemas inspirados nas redes biológicas presentes no cérebro humano. Elas são compostas por unidades de processamento inspiradas no neurônio biológico, que computam determinadas funções matemáticas (BRAGA et al, 2007).

As redes neurais fazem parte de um dos paradigmas da inteligência artificial, que é conhecido como conexionista. Ao contrário da grande maioria dos sistemas computacionais, as redes neurais não são programadas e não utilizam algoritmos, sendo, portanto, uma forma de computação não-algorítmica. O seu funcionamento é baseado em um processo de aprendizado por meio do ajuste dos pesos associados as conexões entre suas unidades de processamento, fazendo com que seja armazenado o conhecimento adquirido (BRAGA et al, 2007).

As redes neurais possuem um grande paralelismo natural devido a distribuição de sua arquitetura, o que gera uma maior velocidade no processamento, fazendo com que muitas vezes elas apresentem um desempenho superior aos outros modelos computacionais (BRAGA et al, 2007).

As capacidades de uma rede neural dependem, principalmente, da estrutura de sua arquitetura. Algumas das principais características das redes neurais são:

- Capacidade de aprendizagem: As redes neurais são capazes de aprender a gerar determinadas saídas a partir de determinados padrões de entrada, quando submetidas a um processo de treinamento.
- Generalização: Esta característica possibilita à rede neural gerar saídas apropriadas com base na sua aprendizagem, quando os dados de entrada são desconhecidos ou diferentes do conjunto de treinamento utilizado na sua aprendizagem.
- Tolerância a falhas: As redes neurais tendem a continuar apresentando resultados aceitáveis depois que um neurônio para de operar. Isso ocorre, pois grande parte do conhecimento contido na rede ainda está disponível em suas outras unidades operantes.

Uma rede neural artificial pode se encontrar em duas fases distintas:

- Aprendizagem: Nesta fase, a rede se encontra no processo de aprendizado, onde seus parâmetros livres são ajustados, para que posteriormente ela seja capaz de desempenhar uma determinada função.
- Aplicação: Nesta fase, a rede se encontra executando a função para a qual ela foi destinada, como a classificação de padrões, otimização, aproximação, previsão, mapeamento de funções, entre outras (BRAGA et al, 2007).

4.1 – Neurônio biológico

Os neurônios biológicos são células presentes no cérebro humano e simplificadamente, eles podem ser divididos em três partes: o corpo celular, os dendritos e o axônio (BRAGA et al, 2007). A função dos dendritos é receber as informações dos impulsos nervosos, oriundas de outros neurônios e levá-las até o corpo celular, onde toda a informação é processada, e novos impulsos são gerados. Esses impulsos são transmitidos a outros neurônios através do axônio, chegando até os dendritos dos próximos neurônios.

O neurônio também possui sinapses que são os pontos de contato entre a terminação do axônio de um neurônio e o dendrito de outro. Uma sinapse funciona como uma válvula, podendo ser excitatória ou inibitória, no primeiro caso, ela estimula o impulso nervoso que passa por ela, já no segundo caso, o impulso é inibido. É através do controle da transmissão de impulsos feito pelas sinapses, que os neurônios se unem funcionalmente, formando as redes neurais biológicas (BRAGA et al, 2007).

Na Figura 4.1 são ilustrados de forma simplificada, os principais componentes de um neurônio biológico. Quando os sinais dos impulsos nervosos passam pelas sinapses, eles são transmitidos para o corpo do neurônio, onde eles são combinados para gerar um percentual de excitação. Caso o percentual de excitação do neurônio seja suficientemente alto, a célula “dispara”, produzindo um impulso nervoso que é transmitido através do axônio para os neurônios seguintes (BRAGA et al, 2007).

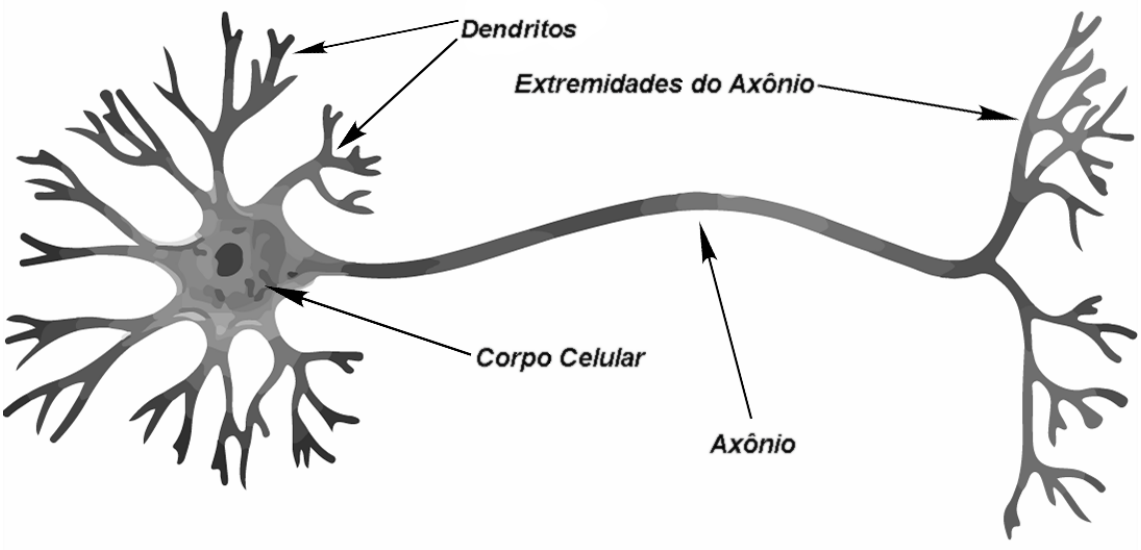

Figura 4.1 – Neurônio Biológico. Fonte: Elaborada pelo autor.

4.2 – Neurônio artificial

Na década de 1940, foi proposto um modelo de neurônio artificial com base no comportamento funcional do neurônio biológico, o modelo MCP, desenvolvido por McCulloch e Pitts (BRAGA et al, 2007). O modelo MCP foi construído a partir de uma simplificação do que se sabia naquela época a respeito do neurônio biológico. Neste modelo há $\boldsymbol{n}$ entradas, que representam os dendritos, sendo que estas entradas recebem valores $\boldsymbol{x_1, x_2, x_3, \ldots, x_n}$, que representam os sinais enviados pelos neurônios anteriores. O modelo MCP possui um único terminal de saída $\boldsymbol{y}$, que representa o axônio do neurônio. O comportamento das sinapses está presente no modelo através da associação de pesos $\boldsymbol{w_1, w_2, w_3, \ldots, w_n}$, com as entradas, onde cada peso pode atuar como uma sinapse excitatória caso o seu valor seja positivo, ou atuar como uma sinapse inibitória, se o seu valor for negativo. Uma representação gráfica desse modelo é ilustrada na Figura 4.2.

Neste modelo artificial, o comportamento de disparo do neurônio biológico é representado por um mecanismo simples, que faz a soma dos valores $\boldsymbol{x_i w_i}$, que é a soma das entradas ponderadas pelos seus pesos (BRAGA et al, 2007). Assim este mecanismo faz o neurônio disparar dependendo do resultado de uma "função de ativação", que recebe como parâmetro, a soma previamente obtida. No modelo MCP, o disparo do neurônio faz com que ele seja ativado, gerando um valor específico no terminal de saída $\boldsymbol{y}$. Como exemplo, $\boldsymbol{y} = 1$ quando o neurônio é ativado e $\boldsymbol{y} = 0$ no contrário.

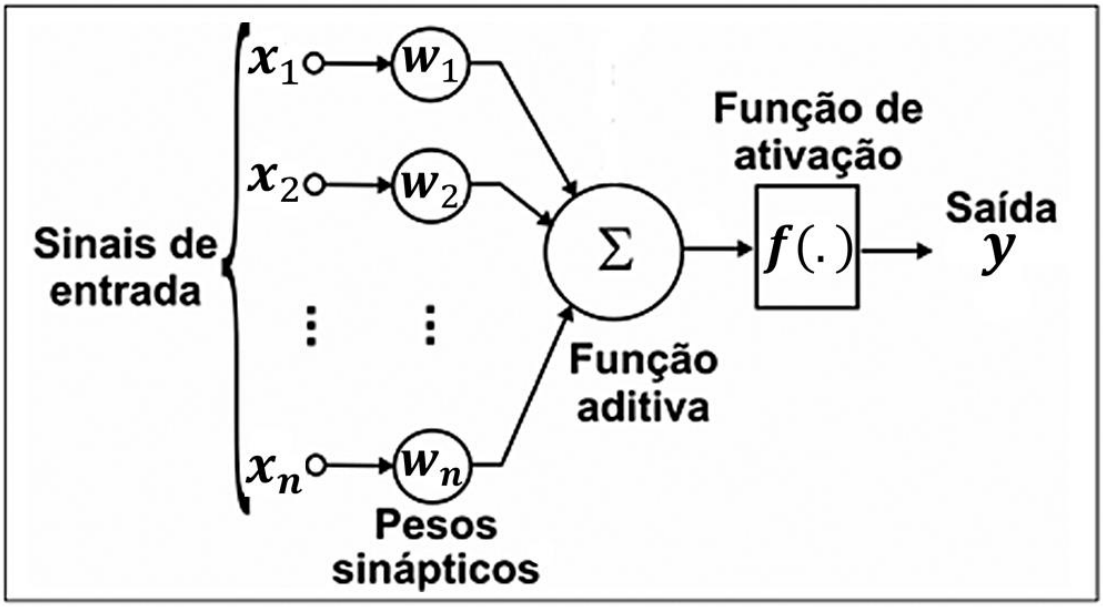

Figura 4.2 – Modelo MCP. Fonte: Elaborada pelo autor.

4.3 – Funções de ativação

Assim como foi discutido na Seção 4.2, a função de ativação é responsável por gerar a saída $\boldsymbol{y}$ do neurônio a partir da soma de suas entradas ponderadas pelos seus pesos. Cada tipo de arquitetura de rede neural utiliza um tipo específico de função de ativação.

A função de ativação do tipo degrau, é usada no modelo de neurônio artificial MCP, e ela leva em consideração um limiar de ativação $\boldsymbol{\theta}$ (BRAGA et al, 2007). Esta função é apresentada na Equação 4.1, e é ilustrada na Figura 4.3(a) para $\boldsymbol{\theta} = 3$.

$$\boldsymbol{f(u)} = \begin{cases} 1, \text{se} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{x_i w_i} \geq \boldsymbol{\theta} \\ 0, \text{se} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{x_i w_i} < \boldsymbol{\theta} \end{cases}$$

Equação 4.1

Onde $\boldsymbol{\theta}$ é o limiar da função de ativação.

Uma outra função de ativação é a função sigmoidal, que é uma aproximação contínua da função degrau. Esta função é ilustrada na Figura 4.3(b) e é representada pela Equação 4.2. Além de ser diferenciável, esta função possui uma região semilinear, o que a torna útil na aproximação de funções contínuas (BRAGA et al, 2007).

$$\boldsymbol{f(u) = \frac{1}{1 + e^{-\beta u}}}$$

Equação 4.2

Onde $\boldsymbol{\beta}$ é a inclinação da função.

Outra função de ativação que é utilizada em neurônios dependendo do tipo de problema a ser resolvido, é a função de ativação linear, que é apresentada na Figura 4.3(c) e é dada pela Equação 4.3 (BRAGA et al, 2007).

$$\boldsymbol{f(u) = u}$$

Equação 4.3

A função de ativação radial é utilizada em redes neurais do tipo RBF (Radial Basis Function). Uma função de ativação radial se caracteriza por apresentar uma resposta que decresce ou cresce com o aumento da distância de um ponto central, como a função gaussiana apresentada na Figura 4.3(d) (BRAGA et al, 2007). A função de ativação radial é representada pela Equação 4.4.

$$\boldsymbol{f(u) = e^{\frac{-(u - \mu)^2}{r^2}}}$$

Equação 4.4

Onde $\boldsymbol{\mu}$ é o centro (ponto médio) e $\boldsymbol{r}$ é o raio de abertura da função.

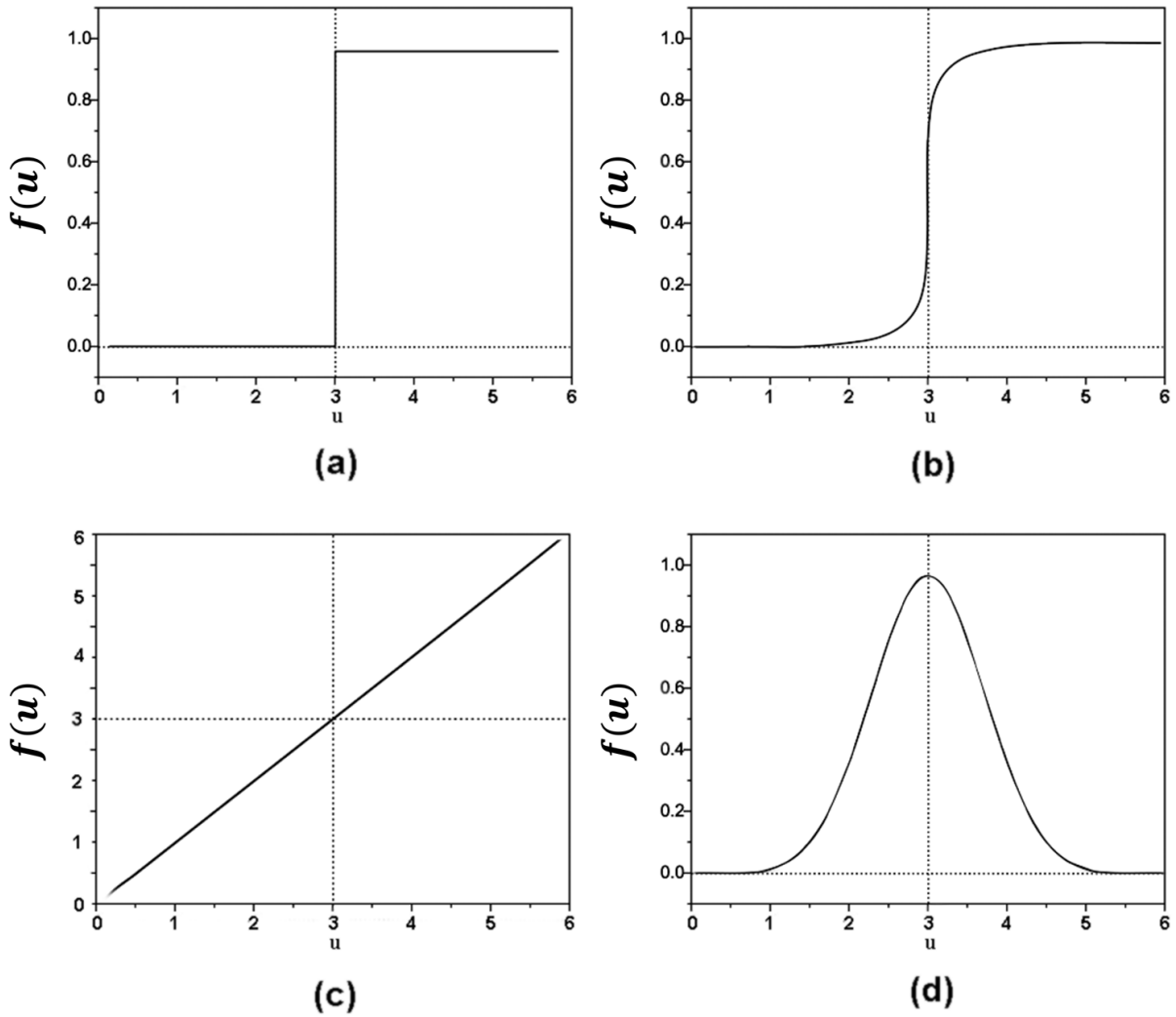

Figura 4.3 – Principais funções de ativação. Fonte: Elaborada pelo autor.

4.4 – Arquiteturas

Os neurônios individuais possuem uma capacidade de computação limitada, portanto, para resolver problemas de complexidade elevada, é necessária uma arquitetura que comporte um conjunto de neurônios artificiais conectados em rede, formando uma rede neural artificial (BRAGA et al, 2007).

As principais arquiteturas de redes neurais podem ser classificadas em duas categorias: redes estáticas e redes recorrentes.

Redes estáticas ou redes *"feedforward"*, ilustradas na Figura 4.4(a e b), geram valores de saída independentes do estado anterior da rede, dependendo apenas do padrão de entrada atual, já que não possuem recorrência em sua estrutura. São redes cujos grafos de interconexões não possuem ciclos e podem ser formadas por várias camadas de neurônios. A estrutura mais simples é apresentada na Figura 4.4(a), que corresponde a uma rede neural com uma única camada de neurônios alimentada para a frente, por uma camada de entrada, gerando uma camada de saída formada pelas saídas dos neurônios. Este tipo de arquitetura é capaz de resolver muitos problemas, no entanto apresenta algumas restrições de complexidade, por possuírem uma única camada. A estrutura apresentada na Figura 4.4(b) é semelhante à da Figura 4.4(a), com a diferença de que possui uma camada adicional de neurônios. Esta camada intermediária confere à RNA um grande aumento na sua capacidade computacional e universalidade na aproximação de funções contínuas (BRAGA et al, 2007).

Redes recorrentes ou redes *"feedback"*, são ilustradas nas Figuras 4.4(c) e 4.4(d), possuem conexões recorrentes entre neurônios de uma mesma camada ou entre neurônios de saída e de camadas anteriores. Na arquitetura ilustrada na Figura 4.4(c), a saída depende não somente dos padrões de entrada, mas também do seu valor atual. Geralmente, este tipo de arquitetura é utilizado na resolução de problemas que envolvam processamento temporal, como a previsão de eventos futuros (BRAGA et al, 2007).

A arquitetura ilustrada na Figura 4.4(d) possui uma única camada de neurônios, em que a saída de cada um deles está conectada às entradas de todos os outros neurônios. Devido a esta estrutura, a rede não possui entradas externas, e sua operação se dá com a mudança dinâmica dos estados dos neurônios, que operam de forma auto associativa.

Para definir a arquitetura de uma RNA, é preciso levar em consideração diversos fatores, como: complexidade do problema, dimensionalidade do espaço de entrada, características dinâmicas ou estáticas, conhecimento *a priori* sobre o problema e representatividade dos dados (BRAGA et al, 2007).

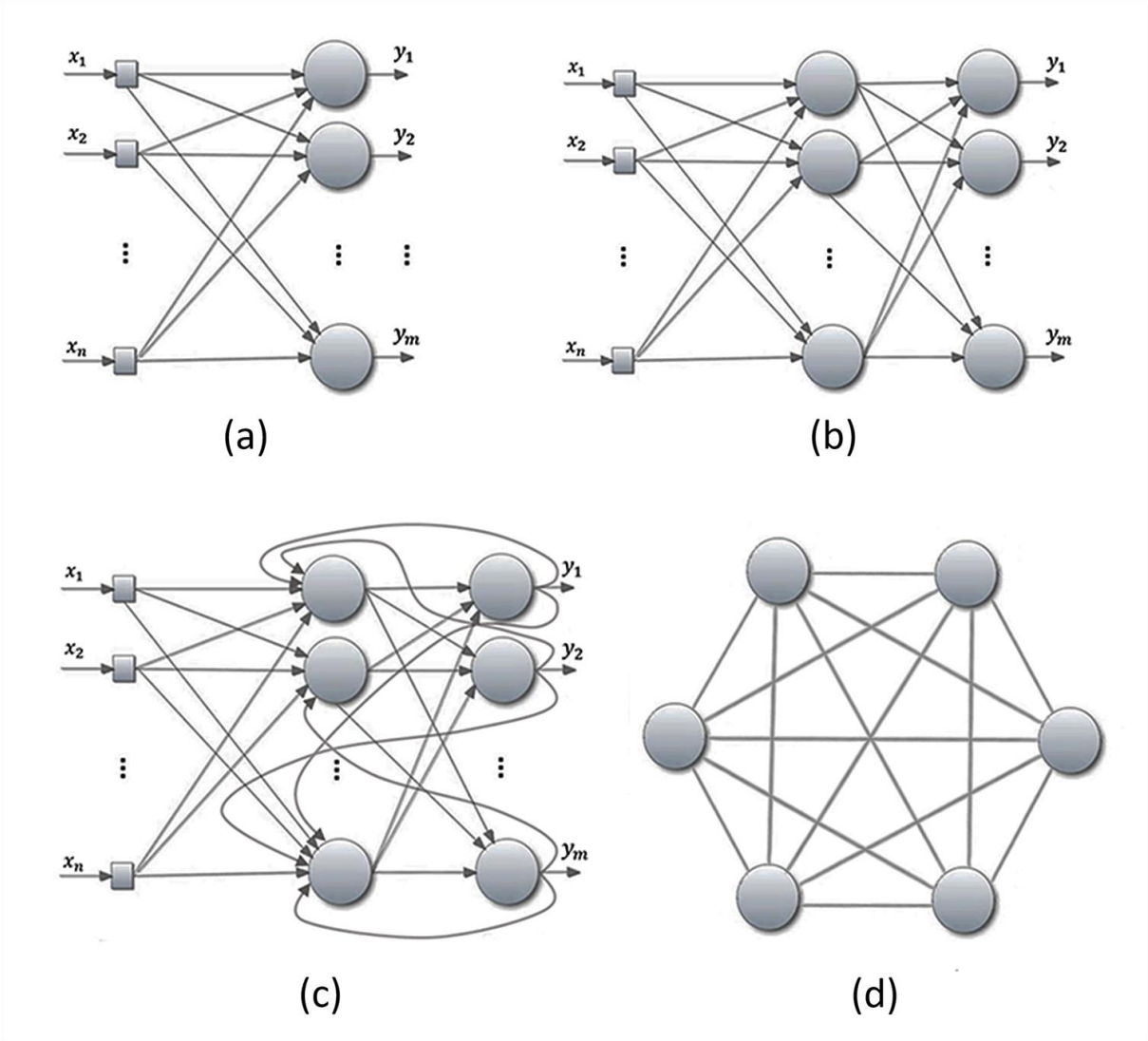

Figura 4.4 – Arquiteturas de redes neurais. Fonte: Elaborada pelo autor.

4.5 – Tipos de aprendizado

Uma das principais características presentes nas redes neurais artificiais, é a aprendizagem por meio de exemplos. Na etapa de aprendizado de uma RNA, os seus parâmetros livres são ajustados iterativamente por meio de uma forma continuada de estímulo pelo ambiente externo, fazendo com que ao final do processo a rede possa ter adquirido um conhecimento sobre o ambiente externo (BRAGA et al, 2007). O conhecimento que uma rede neural adquire no aprendizado se encontra distribuído por toda a rede em seus neurônios.

O tipo específico de aprendizado é definido pela maneira particular em que ocorrem os ajustes dos parâmetros livres. O processo de aprendizado está relacionado com a melhoria do desempenho da rede segundo algum critério preestabelecido, como por exemplo o grau de erro da rede. Existem dois paradigmas principais de aprendizado: aprendizado supervisionado e aprendizado não-supervisionado (BRAGA et al, 2007).

No aprendizado supervisionado existe a figura de um supervisor, ou professor externo, o qual é responsável por apresentar padrões de entrada à rede e comparar a saída calculada pela mesma com a saída desejada. Como a saída da rede varia de acordo com os valores de seus pesos, estes valores são modificados iterativamente para aproximar a saída da rede da saída desejada. Na Figura 4.5 é ilustrada uma representação esquemática do aprendizado supervisionado. O aprendizado supervisionado se aplica a problemas em que seja possível fazer um mapeamento dos padrões de entrada com as saídas desejadas.

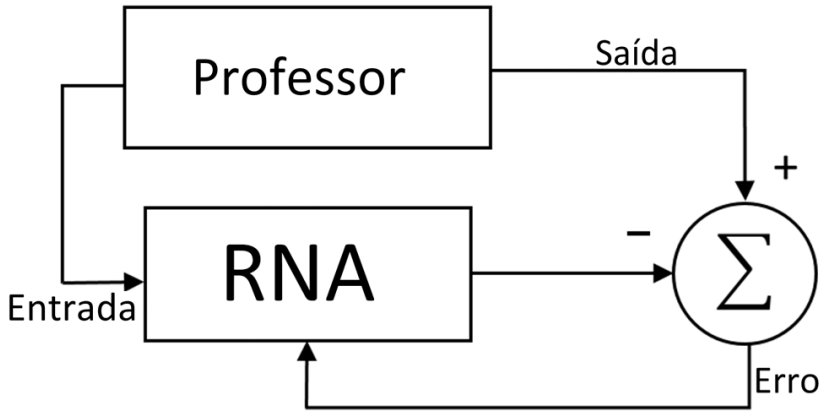

Figura 4.5 – Esquema de aprendizado supervisionado. Fonte: Elaborada pelo autor.

Os algoritmos de treinamento para aprendizado supervisionado mais conhecidos são a regra delta e o algoritmo *"backpropagation"*, que é a generalização da regra delta para redes neurais de múltiplas camadas.

No aprendizado não-supervisionado, não existe a figura de um professor ou supervisor externo para controlar o processo de aprendizado. Neste paradigma, o treinamento é feito apenas disponibilizando os padrões de entrada para a rede, não necessitando disponibilizar saídas desejadas. No processo de aprendizado os padrões de entrada são continuamente apresentados a rede, e então, as regularidades presentes nesses padrões de entrada faz com que o aprendizado seja possível. Para o sucesso do aprendizado não-supervisionado, é preciso haver regularidades e redundâncias nos padrões de entrada apresentados à rede.

Uma representação esquemática do aprendizado não-supervisionado é apresentada na Figura 4.6. O aprendizado não-supervisionado é muito útil na resolução de problemas como agrupamento e classificação.

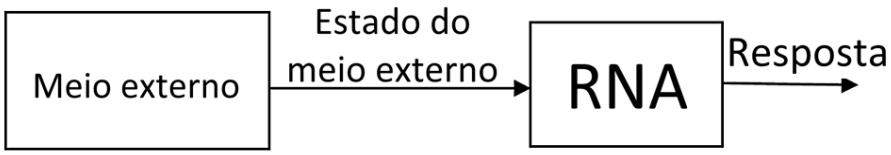

Figura 4.6 – Esquema de aprendizado não-supervisionado. Fonte: Elaborada pelo autor.

O processo de aprendizado é concretizado quando a rede gera saídas coerentes com a solução do problema em questão.

4.6 – Modelo Perceptron

O modelo Perceptron foi proposto por Frank Rosenblatt, em 1958, e se baseava em uma arquitetura de rede neural com unidades básicas de neurônios MCP, seguindo uma regra de aprendizado supervisionado (BRAGA et al, 2007).

O modelo original do Perceptron, é ilustrado na Figura 4.7, sendo composto por uma camada de entrada, uma camada intermediária com neurônios MCP para fazer a associação das entradas, e uma camada de saída composta pelas saídas dos neurônios.

Rosenblatt (1962) demostrou que um neurônio MCP treinado com o algoritmo de aprendizado do perceptron sempre converge em uma solução caso o problema em questão seja linearmente separável. Os neurônios do perceptron são similares ao modelo MCP, mas possuem uma entrada extra com valor fixo $\boldsymbol{x}_0 = 1$, onde essa entrada é associada com um peso $\boldsymbol{w}_0 = \boldsymbol{\theta}$. Assim, os neurônios do modelo perceptron possuem um vetor de entrada $\boldsymbol{x} = \{1, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ e um vetor de pesos $\boldsymbol{w} = \{\boldsymbol{\theta}, \boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n\}$.

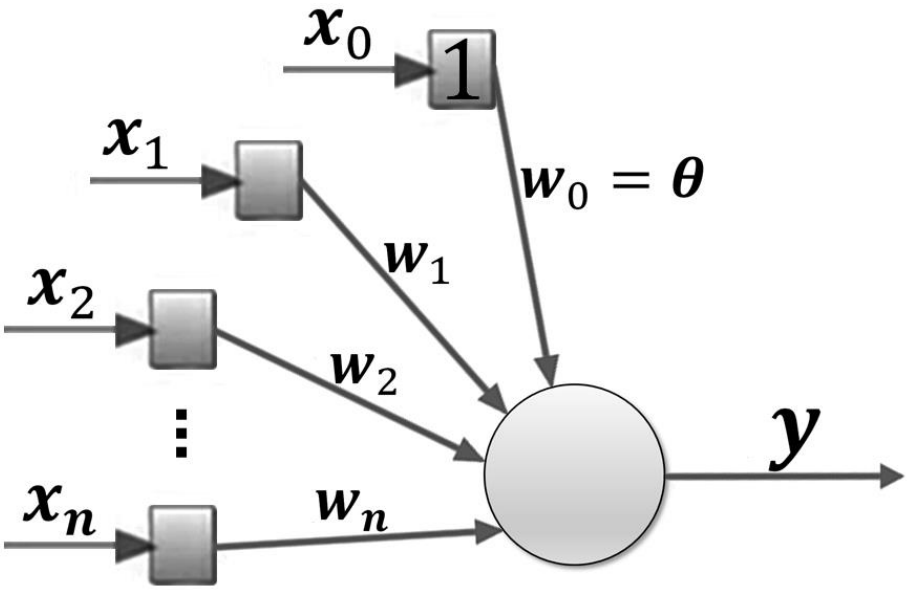

Figura 4.7 – Perceptron de uma única saída. Fonte: Elaborada pelo autor.

De acordo com Braga et al (2007), na regra de aprendizado do perceptron, o objetivo é atualizar o vetor de pesos $\boldsymbol{w}(\boldsymbol{n})$, onde $\boldsymbol{n}$ é o instante atual, fazendo com que o seu valor atualizado $\boldsymbol{w}(\boldsymbol{n}+1)$ esteja mais próximo da solução desejada em relação à $\boldsymbol{w}(\boldsymbol{n})$. O valor atualizado do vetor de pesos é dado por:

$$\boldsymbol{w}(\boldsymbol{n}+1) \;=\; \boldsymbol{w}(\boldsymbol{n}) + \boldsymbol{\eta e x}(\boldsymbol{n})$$

Onde $\boldsymbol{\eta}$ é o valor da taxa de aprendizado escolhida, $\boldsymbol{x}$ é o vetor de entrada e $\boldsymbol{e}$ é o erro dado por:

$$\boldsymbol{e} = \boldsymbol{y_d} - \boldsymbol{y}$$

Onde $\boldsymbol{y_d}$ é a saída desejada para o vetor de entrada $\boldsymbol{x}$, e $\boldsymbol{y}$ é a saída atual da rede a partir do vetor de entrada $\boldsymbol{x}$.

Desta forma, o aprendizado do perceptron pode ser resumido pelo seguinte algoritmo:

- Atribuir um valor para $\boldsymbol{\eta}$;
- Inicializar o vetor de pesos $\boldsymbol{w}$ com valores aleatórios, geralmente entre 0 e 1;
- Fazer a atualização dos pesos através da regra $\boldsymbol{w}(\boldsymbol{n}+1) = \boldsymbol{w}(\boldsymbol{n}) + \boldsymbol{\eta e x}(\boldsymbol{n})$ para todos os $\boldsymbol{p}$ pares $(\boldsymbol{x^i},\ \boldsymbol{y_d^i})$ do conjunto de treinamento $\boldsymbol{\Gamma} = \{(\boldsymbol{x^i},\ \boldsymbol{y_d^i})\}_{\boldsymbol{i=1}}^{\boldsymbol{p}}$;
- Repetir o passo 3 até que $\boldsymbol{e} = 0$ para todos os $\boldsymbol{p}$ elementos de $\boldsymbol{\Gamma}$.

4.7 – Modelo MLP

Redes neurais artificiais de uma única camada possuem a limitação de resolver apenas problemas que sejam linearmente separáveis. No entanto, a grande maioria das situações e problemas reais apresentam características não-lineares. Este problema pode ser solucionado com o uso de redes neurais Perceptron de Múltiplas Camadas (MLP – *Multilayer Perceptron*) (BRAGA et al, 2007). Teoricamente uma rede neural MLP com uma camada intermediária pode aproximar qualquer função contínua (CYBENKO, 1989), e duas camadas intermediarias já é capaz de implementar qualquer função matemática, mesmo não sendo linearmente separável (DUDA et al, 2001).

Uma rede neural MLP é formada por múltiplas camadas de neurônios, podendo ter uma ou mais camadas intermediárias. As múltiplas camadas de uma rede MLP são responsáveis por transformar, sucessivamente, o problema descrito pelo conjunto de padrões de entrada em uma representação tratável para a camada de saída da rede (BRAGA et al, 2007).

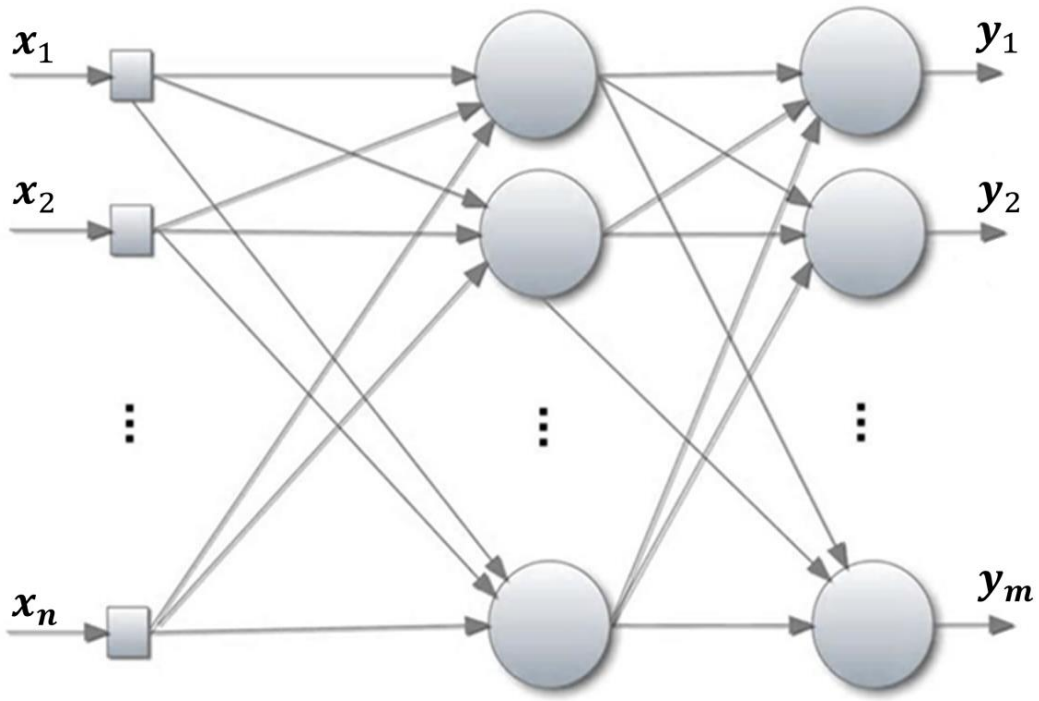

Figura 4.8 – Exemplo de rede neural MLP. Fonte: Elaborada pelo autor.

No processo de aprendizado de uma rede neural MLP, um dos algoritmos mais utilizados é o algoritmo de retro propagação de erros ou *"backpropagation"* (BRAGA et al, 2007). Este algoritmo surgiu no meio da década de 1980 e foi a primeira solução de treinamento supervisionado desenvolvida para redes MLP. Por ser supervisionado, o algoritmo *"backpropagation"* utiliza o erro da rede em seu processo de treinamento. Assim como no modelo perceptron, o erro é calculado com base na saída da rede e na saída desejada, no entanto, este processo é feito apenas para a camada de saída, pois não existem saídas desejadas definidas para as camadas intermediárias. Para estimar o erro das camadas intermediárias, o algoritmo utiliza o gradiente descendente, por meio de uma estimativa do efeito que as camadas intermediárias causam no erro da camada de saída. Para isso, o algoritmo calcula o erro da camada de saída da rede e então, faz a sua retroalimentação para as camadas intermediárias, fazendo o ajuste dos pesos de acordo com os valores das conexões entre as camadas.

Devido ao uso do gradiente descendente os neurônios da rede neural MLP não podem utilizar a função de ativação do tipo degrau presente no modelo perceptron. A função sigmoidal é geralmente utilizada para substituir a função de ativação do tipo degrau. Mais detalhes e informações sobre o algoritmo "*backpropagation"* podem ser encontradas em (BRAGA et al, 2007).

4.8 – Modelo SOM

As redes neurais artificiais SOM (do inglês, *Self-Organizing Maps*), conhecidas também como mapas auto organizáveis, fazem parte de uma classe de RNAs que apresentam a capacidade de auto-organização, sendo por isso chamadas de redes auto organizáveis (BRAGA et al, 2007). Redes neurais dessa classe são muito úteis em diversas aplicações onde é necessário que a rede seja treinada através do paradigma de aprendizado não-supervisionado. A única informação fornecida a essas redes é o conjunto de padrões de entrada, que é usado por elas para ajustar seus parâmetros por si próprias através de regras locais, sem qualquer auxílio externo.

4.8.1 – Informações

De acordo com Braga et al (2007), as redes neurais auto organizáveis são mais semelhantes as estruturas neurobiológicas em relação as redes neurais supervisionadas. Devido a essas características, as RNAs auto organizáveis podem ser usadas em muitas aplicações, como por exemplo, no reconhecimento de padrões e no agrupamento de dados em que as classes não são conhecidas previamente, possibilitando uma classificação dos padrões de entrada. Alguns dos principais modelos de redes neurais dessa classe são as redes SOM, e as redes ART (do inglês, *Adaptive Resonance Theory*). Nesse capítulo será apresentado apenas um estudo feito sobre o modelo SOM, o qual foi escolhido para a implementação deste trabalho.

As redes neurais artificiais SOM foram desenvolvidas por Teuvo Kohonen (KOHONEN, 1989). Essas redes são fortemente inspiradas no mapa topológico presente no córtex cerebral. Isso se deve ao fato de que existem áreas do cérebro que são responsáveis por funções específicas, como por exemplo, a fala, a visão, o controle motor, a sensibilidade ao toque, entre outras. Nessas áreas, como os neurônios estão espacialmente ordenados, aqueles que estão topologicamente próximos, tendem a responder a padrões ou estímulos semelhantes. Essa ordenação topológica é devido à um *"feedback"* lateral entre as células do córtex cerebral que estão localmente interconectadas (BRAGA et al, 2007).

4.8.2 – Arquitetura

Em uma rede neural SOM, os neurônios são geralmente organizados em uma grade ou reticulado bidimensional, na forma de uma superfície plana, onde os neurônios de saída estão organizados em linhas e colunas, assim como é ilustrado na Figura 4.9. Todos os neurônios da rede SOM recebem o vetor de entrada $\boldsymbol{x}$ da rede, que é usado para apresentar os padrões de entrada a rede. A saída de uma rede SOM é formada pela saída de todos os seus neurônios (BRAGA et al, 2007).

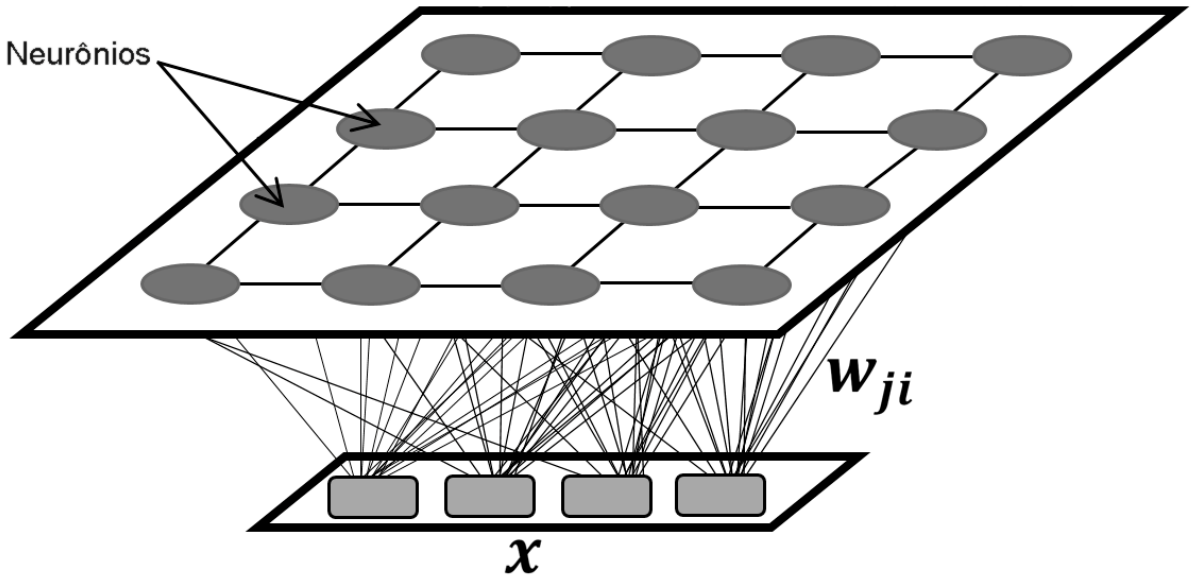

Figura 4.9 – Exemplo de rede neural SOM. Fonte: Elaborada pelo autor.

Em uma rede SOM, cada neurônio possui uma vizinhança que pode assumir vários formatos diferentes. Embora o formato quadrado seja o mais comum, há também os formatos hexagonal e circular, ilustrados na Figura 4.10. A definição do formato mais adequado depende do problema e da distribuição dos dados, sendo muitas vezes definido em um processo de tentativa e erro (BRAGA et al, 2007). A vizinhança de um neurônio nas redes SOM é usada para modelar o *"feedback"* lateral entre os neurônios biológicos.

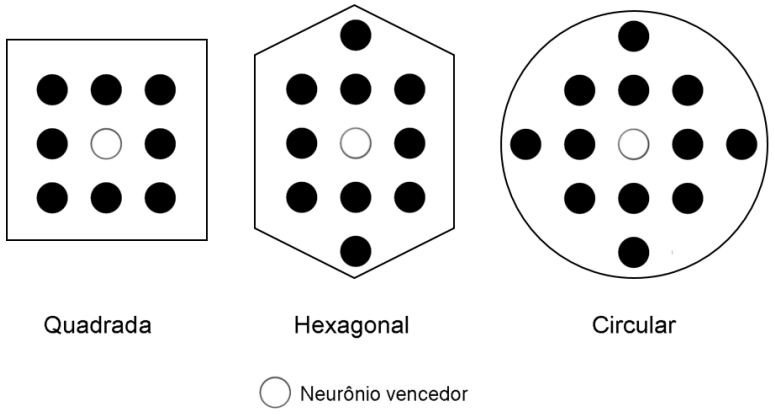

Figura 4.10 – Formatos da região de vizinhança. Fonte: Elaborada pelo autor.

Cada neurônio de uma rede neural SOM possui um vetor de pesos associado. Inicialmente todos os pesos de uma rede neural SOM são definidos por valores aleatórios, onde preferencialmente, esses valores devem ser diferentes (HAYKIN, 1999). Para um melhor funcionamento da rede os vetores de pesos e os vetores de entrada devem ser normalizados, geralmente para uma norma unitária (BRAGA et al, 2007).

4.8.3 – Aprendizado

Na rede SOM é utilizado um algoritmo de aprendizado competitivo e não-supervisionado, onde os neurônios competem entre si para se tornarem ativos. Assim para cada padrão de entrada apenas um neurônio de saída ou neurônio por grupo se torna ativo. O estado de ativação de um neurônio é determinado pela distância entre seu peso e o vetor de entrada, através da Equação 4.5, que representa uma função de ativação baseada na medida de distância euclidiana (AFFONSO, 2011). Quanto maior for a semelhança entre a entrada da rede e o vetor de pesos de um neurônio, menor será o valor de saída desta função. Um neurônio se torna ativo quando gerar a menor saída, dentre os outros neurônios da rede.

$$\boldsymbol{y_j} = \sum_{\boldsymbol{i}=1}^{\boldsymbol{n}} \|\boldsymbol{x_i} - \boldsymbol{w_{ji}}\|$$

Equação 4.5

Onde $\boldsymbol{w_{ji}}$ representa o peso na posição $\boldsymbol{i}$ do vetor de pesos do neurônio $\boldsymbol{j}$ e $\boldsymbol{x_i}$ é o valor da posição $\boldsymbol{i}$ do vetor de entrada $\boldsymbol{x}$.

Desta forma quando é apresentado um determinado padrão de entrada $\boldsymbol{p}$ para a rede, ela faz uma busca pelo neurônio mais parecido com $\boldsymbol{p}$, ou seja, aquele que apresentar a menor saída para a Equação 4.5. O neurônio mais parecido é o vencedor da competição entre os neurônios. Depois inicia-se o processo de atualização de pesos, onde apenas o neurônio vencedor e seus vizinhos dentro de um certo raio ou área de vizinhança atualizam seus pesos.

Este processo faz com que o neurônio vencedor e seus vizinhos, se tornem mais semelhantes ao padrão de entrada $\boldsymbol{p}$, fazendo com que seja construído um mapa topológico na rede, onde neurônios que estão topologicamente próximos tendem a responder de maneira similar a padrões de entrada semelhantes. De acordo com Braga et al (2007), a atualização dos pesos do neurônio vencedor e daqueles situados em sua vizinhança, é dada por:

$$\boldsymbol{w_{ji}(t+1)} = \begin{cases} \boldsymbol{w_{ji}(t) + \eta(t)\big(x_i(t) - w_{ji}(t)\big)}, \text{se } \boldsymbol{j \in \Lambda(t)} \\ \boldsymbol{w_{ji}(t)}, \text{caso contrário} \end{cases}$$

Equação 4.6

Onde w_{ji} é o peso na posição i do vetor de pesos do neurônio j, x_i é o valor na posição i do vetor de entrada x, η é a taxa de aprendizado, Λ é a vizinhança do neurônio vencedor, e t é o instante de tempo atual. Durante os ciclos de treinamento, a taxa de aprendizado e o raio de vizinhança vão sendo continuamente decrementados. Após várias apresentações do conjunto de treinamento, os vetores de pesos dos neurônios tendem a seguir as distribuições dos vetores de entrada.

Todo este processo de treinamento da rede SOM pode ser dividido em duas fases: a fase de ordenação e a fase de convergência (BRAGA et al, 2007).

Durante a fase de ordenação, os vetores de pesos são topologicamente ordenados. Assim, os neurônios do mapa topológico são reunidos em agrupamentos que refletem a distribuição dos padrões de entrada. Com isso, a rede pode descobrir quantos grupos devem ser identificados e quais as suas localizações relativas no mapa. Esta etapa do treinamento é geralmente caracterizada por realizar um mapeamento grosseiro dos padrões de entrada, onde a taxa de aprendizado é inicialmente alta, próxima de 1, sendo gradualmente reduzida até um valor próximo de 0,1. Por isso, nesta fase também ocorrem grandes mudanças nos pesos e a vizinhança dos neurônios é reduzida até atingir um raio de cerca de um ou dois vizinhos.

Na fase de convergência é feito um ajuste mais fino do mapa na rede, gerando cerca de 100 a 1000 vezes mais ciclos que a fase anterior. Nesta segunda fase é utilizada uma taxa de aprendizado baixa, cerca de 0,01 ou menos, e o raio da vizinhança geralmente é de um ou nenhum vizinho. Com isso, é feito um aprimoramento do mapeamento realizado no estágio anterior, melhorando o agrupamento realizado. Na Figura 4.11 é mostrado um exemplo de redução da região de vizinhança durante o treinamento de uma rede neural SOM.

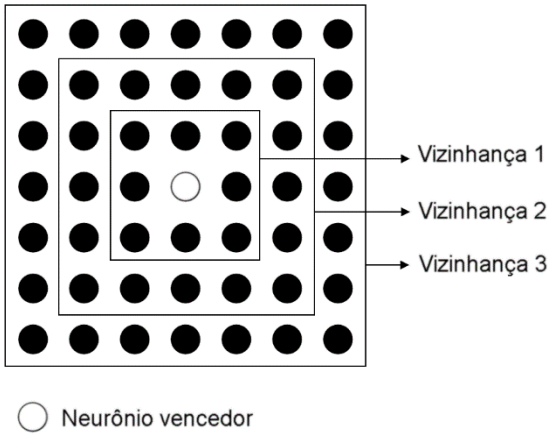

Figura 4.11 – Redução da região de vizinhança. Fonte: Elaborada pelo autor.

Como antes do treinamento os vetores de pesos dos neurônios são inicializados com valores aleatórios, muitos neurônios podem apresentar vetores de pesos muito diferentes dos padrões de entrada, dificultando a definição dos agrupamentos na rede durante o processo de aprendizado. Devido a este problema, a rede pode não convergir ou apresentar ciclos muito lentos no aprendizado. Uma alternativa para resolver ou minimizar esse problema, é usar o algoritmo de consciência. Este algoritmo implementa um limiar para cada neurônio, onde este limiar seria a consciência do neurônio (DESIENO, 1988), por isso, esta técnica também é chamada de consciência. De acordo com essa técnica, neurônios frequentemente selecionados nas competições teriam seu limiar aumentado, gerando um registro de quantas vezes ele ganhou a competição. Se um neurônio vence a competição com muita frequência, ele "se sente culpado", ou seja, sua chance de ser selecionado novamente é reduzida podendo até mesmo ser retirado da competição, possibilitando a utilização de outros neurônios.

Depois do treinamento, os neurônios se organizam topologicamente, fazendo com que os padrões detectados por um dado neurônio estejam relacionados com as coordenadas da sua posição dentro do reticulado. Assim, é formado um mapa de características auto organizado com mapas topológicos dos padrões de entrada, onde padrões semelhantes são detectados por neurônios próximos dentro do reticulado (BRAGA et al, 2007).

Após o treinamento, com os agrupamentos gerados pela rede SOM baseados nos padrões de entrada, em algumas aplicações, pode ser necessário rotular os neurônios de saída para indicar os grupos ou classes que eles representam. Esta fase de rotulação ocorre de forma supervisionada, e é útil para permitir posteriormente a classificação de padrões desconhecidos (BRAGA et al, 2007).

CAPÍTULO 5 – DESENVOLVIMENTO

O desenvolvimento deste trabalho envolveu a criação de duas aplicações para se alcançar os objetivos descritos na Seção 1.1, a seguir serão apresentadas em detalhes todas as etapas de cada uma destas aplicações.

5.1 – Extrator de características

Foi feita a implementação de uma aplicação chamada neste trabalho de **Extrator de características**, que é responsável por gerar um vetor descritor para uma dada imagem de entrada, utilizando para isso o algoritmo SIFT ou o algoritmo SURF. Para desenvolver esta aplicação, foi utilizada a biblioteca de visão computacional OpenCV 3.0 juntamente com o seu repositório de módulos extras, onde as implementações dos algoritmos SIFT e SURF estão disponíveis. A linguagem utilizada na codificação foi a C++.

A implementação da aplicação **Extrator de características** se desenvolveu em basicamente duas etapas:

1. Descrição de pontos de interesse;
2. Geração de vetor descritor.

5.1.1 – Descrição de pontos de interesse

Nesta etapa, deve ser selecionado na aplicação, o algoritmo SIFT ou o algoritmo SURF. Então, com base no algoritmo selecionado é realizada a detecção e a descrição dos pontos de interesse em uma dada imagem de entrada conforme explicado nas seções 3.1 e 3.2.

Os parâmetros utilizados nesta tarefa para o algoritmo SIFT foram:

- **NOctaveLayers**: Este parâmetro se refere ao número de camadas $\boldsymbol{L(x, y, \sigma)}$ de cada oitava do espaço de escalas, assim como é explicado na Seção 3.1.1. Para este parâmetro foi usado o valor 3, que é o padrão da biblioteca OpenCV 3.0.
- **ContrastThreshold**: Este parâmetro se refere ao limiar $\boldsymbol{ct}$ usado na eliminação de pontos chave com baixo contraste, assim como é descrito na Seção 3.1.2. Quanto maior o valor deste limiar, mais pontos chave serão produzidos pelo algoritmo. Para este parâmetro foi usado o valor $0{,}04$, que é o padrão da biblioteca OpenCV 3.0.

- **EdgeThreshold**: Este parâmetro se refere ao limiar $\boldsymbol{r}$ usado na eliminação de arestas, assim como é descrito na Seção 3.1.2. Quanto maior o valor deste limiar, menos pontos chave serão produzidos pelo algoritmo. Para este parâmetro foi usado o valor 10, que é o mesmo utilizado em (LOWE, 2004).
- **Sigma**: Este parâmetro se refere ao valor de sigma do filtro Gaussiano aplicado na imagem de entrada da primeira oitava, assim como é descrito na Seção 3.1.1. Para este parâmetro foi usado o valor 1,6, que é o mesmo utilizado em (LOWE, 2004).

Para o algoritmo SURF, os parâmetros utilizados foram os seguintes:

- **HessianThreshold**: Este parâmetro é também chamado de limiar hessiano pois ele é um limiar usado para avaliar o determinante da matriz Hessiana, que é utilizada na fase de detecção dos pontos de interesse, descrita na Seção 3.2.1. Caso o determinante da matriz Hessiana em um determinado ponto da imagem for maior que este parâmetro, então este ponto é considerado um ponto chave. Quanto maior o valor deste parâmetro, menos pontos de interesse serão detectados, no entanto serão pontos mais salientes. Quanto menor o valor deste parâmetro, mais pontos chave serão detectados, no entanto serão pontos menos salientes. Para este parâmetro foi utilizado um valor de 800, pois foi o valor que possibilitou o maior desempenho do algoritmo.
- **NOctaves**: Este parâmetro se refere ao número de oitavas do espaço de escalas descrito na Seção 3.2.1. Para este parâmetro foi utilizado o valor 4, que é o padrão da biblioteca OpenCV 3.0.
- **NOctaveLayers**: Este parâmetro se refere ao número de imagens geradas em cada oitava do espaço de escalas, através da aplicação dos diferentes filtros, assim como é descrito na Seção 3.2.1. Para este parâmetro foi utilizado o valor 3, que é o padrão da biblioteca OpenCV 3.0.

Após esta etapa é gerado um conjunto $\boldsymbol{C}$ de vetores descritores, sendo um vetor descritor para cada ponto de interesse detectado na imagem de entrada.

5.1.2 – Geração de vetor descritor

Na segunda etapa desta aplicação, é calculado um vetor descritor **F** de $\boldsymbol{X}$ elementos, onde $\boldsymbol{X} = 128$ caso na aplicação tenha sido selecionado o algoritmo SIFT, e $\boldsymbol{X} = 64$ caso tenha sido selecionado o algoritmo SURF.

O cálculo do vetor **F** é dado pela média aritmética feita entre os vetores descritores gerados na primeira etapa descrita na seção anterior. Assim, o cálculo de um determinado elemento na posição $\boldsymbol{v}$ do vetor **F** é dado por:

$$\mathbf{F}_{\boldsymbol{v}} = \frac{\sum_{\boldsymbol{i}=0}^{\boldsymbol{n}-1} \boldsymbol{C}_{\boldsymbol{i}_{\boldsymbol{v}}}}{\boldsymbol{n}}$$

Equação 5.1

Onde $\boldsymbol{C}$ é o vetor ou conjunto de $\boldsymbol{n}$ vetores descritores de $\boldsymbol{X}$ elementos gerados na primeira etapa.

Caso o algoritmo selecionado tenha sido o SIFT, os elementos do vetor **F** são ainda divididos por 100, para que seus valores sejam normalizados entre 0 e 1. Essa etapa não é necessária caso o algoritmo SURF tenha sido selecionado, pois os valores calculados para os seus vetores descritores já são normalizados entre -1 e 1.

Como o vetor **F** será utilizado no treinamento e na aplicação da rede neural SOM, esta normalização dos seus elementos melhora o funcionamento da rede, assim como é explicado no final da Seção 4.8.2.

Assim, a partir de uma dada imagem de entrada, a aplicação **Extrator de características** gera como saída o vetor descritor **F**, o qual representa a imagem de entrada.

5.2 – Classificador

Foi realizada a implementação de uma outra aplicação chamada neste trabalho de **Classificador**. Para isso foi utilizada a linguagem de programação Java juntamente com o Framework Encog 3.3.0 para a codificação.

A aplicação **Classificador** tem como principal função, implementar um processo de classificação de sub-imagens, para alcançar o objetivo descrito na Seção 1.1, utilizando para isso uma rede neural SOM. Assim, para compreender as seções seguintes, considere a imagem **I** como sendo a imagem escolhida para se extrair sub-imagens e classificá-las de acordo com as diferentes texturas da imagem **I**.

5.2.1 – Inicialização

A primeira etapa da aplicação **Classificador** é a inicialização da rede neural SOM que será utilizada. Para isso, a aplicação é capaz de instanciar uma rede neural SOM adaptada para o algoritmo SIFT ou adaptada para o algoritmo SURF.

A adaptação da rede neural SOM é feita de acordo com a dimensão dos vetores descritores gerados por cada um destes algoritmos. Assim, quando a rede neural SOM é adaptada para o algoritmo SIFT, cada neurônio possui 128 entradas, e quando for adaptada para o algoritmo SURF, os neurônios possuem 64 entradas.

As entradas dos neurônios recebem os valores das posições de um vetor de entrada **x** durante o treinamento ou aplicação da rede neural, assim como é ilustrado na Figura 4.9 da Seção 4.8.2. A aplicação desenvolvida pode instanciar a rede neural SOM com um reticulado de neurônios de dimensões $\boldsymbol{W} \times \boldsymbol{H}$, onde $\boldsymbol{W}$ é a largura e $\boldsymbol{H}$ a altura. Assim, a rede neural instanciada possui $\boldsymbol{W} * \boldsymbol{H}$ neurônios. Depois que a rede neural é instanciada, os pesos de todos os seus neurônios, são inicializados com valores aleatórios no intervalo $[-1, 1]$.

A aplicação **Classificador** pode instanciar múltiplas redes neurais com as características citadas acima, podendo realizar o processamento descrito nas próximas seções em cada uma das redes neurais instanciadas.

5.2.2 – Treinamento

A segunda etapa da aplicação, consiste em fazer o treinamento da rede neural SOM instanciada conforme explicado na seção anterior, com um conjunto **T** de padrões de entrada. O processo de formação do conjunto **T** pode ser descrito da seguinte forma:

Primeiramente, são selecionadas algumas sub-imagens a partir das diferentes texturas da imagem **I**. Estas sub-imagens não possuem dimensões definidas, mas devem fazer parte de apenas uma das texturas da imagem **I**. Cada uma destas sub-imagens são usadas como entrada da aplicação **Extrator de características**, fazendo com que seja gerado um vetor **F** de saída para cada sub-imagem, assim como é ilustrado na Figura 5.1. Cada vetor **F** gerado é adicionado ao conjunto **T**. Os vetores do conjunto **T** devem possuir todos a mesma dimensão, que é definida pela seleção do algoritmo na aplicação **Extrator de características**, assim como é explicado na Seção 5.1.1.

O algoritmo selecionado na aplicação **Extrator de características** deve ser o mesmo para o qual a rede neural instanciada foi adaptada conforme explicado na Seção 5.2.1.

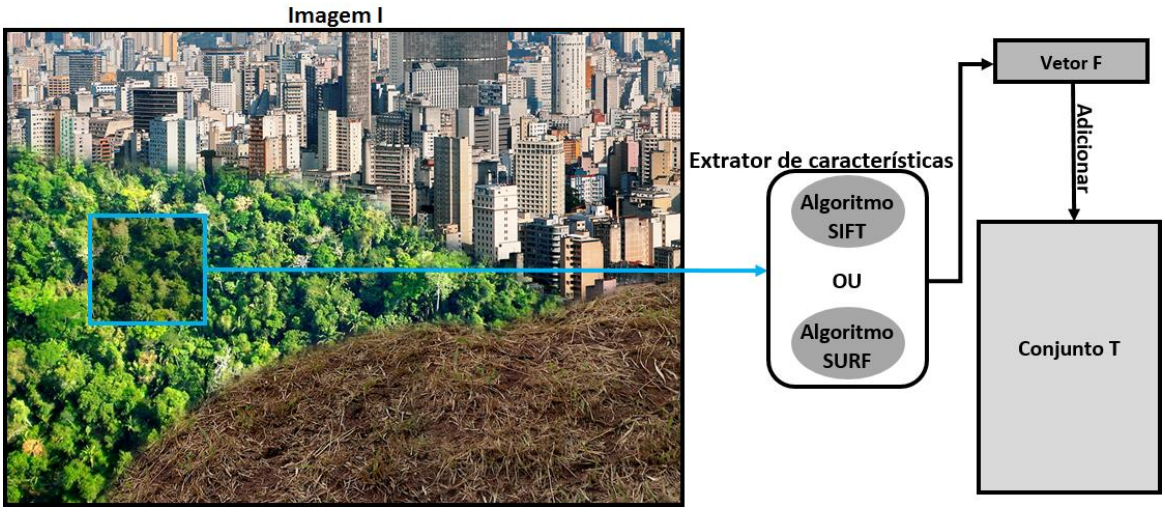

Figura 5.1 – Processo de formação do conjunto de treinamento. Fonte: Elaborada pelo autor.

A implementação do treinamento da rede neural SOM foi feita conforme é explicado na Seção 4.8.3, no entanto, houve uma modificação na Equação 4.6, que é responsável pela atualização dos pesos do neurônio vencedor e daqueles situados em sua vizinhança. Esta modificação foi feita com a adição da função de vizinhança $\boldsymbol{\theta(j,t)}$, que será abordada na próxima seção. Assim, nesta aplicação, a atualização dos pesos do neurônio vencedor e daqueles situados em sua vizinhança é feita através da seguinte equação:

$$\boldsymbol{w_{ji}(t+1)} = \begin{cases} \boldsymbol{w_{ji}(t) + \theta(j,t)\eta(t)\left(x_i(t) - w_{ji}(t)\right)}, \text{se } \boldsymbol{j \in \Lambda(t)} \\ \boldsymbol{w_{ji}(t)}, \text{caso contrário} \end{cases}$$

Equação 5.2

Onde $\boldsymbol{w_{ji}}$ é o peso na posição $\boldsymbol{i}$ do vetor de pesos do neurônio $\boldsymbol{j}$, $\boldsymbol{x_i}$ é o valor na posição $\boldsymbol{i}$ do vetor de entrada $\boldsymbol{x}$, $\boldsymbol{\eta}$ é a taxa de aprendizado, $\boldsymbol{\Lambda}$ é a vizinhança do neurônio vencedor, e $\boldsymbol{t}$ é o instante de tempo atual.

O treinamento da rede neural é feito em apenas uma única fase, onde a taxa de aprendizado e o raio de vizinhança do neurônio vencedor vão sendo constantemente decrementados durante o ciclo de iterações. Assim, alguns parâmetros devem ser definidos para o treinamento da rede neural:

- **Número de iterações**: É um número fixo de iterações que serão utilizadas para o treinamento da rede neural.

- **Taxa de aprendizado inicial**: O valor da taxa de aprendizado para a atualização dos pesos do neurônio vencedor e seus vizinhos na primeira iteração.
- **Taxa de aprendizado final**: O valor da taxa de aprendizado para a atualização dos pesos do neurônio vencedor e seus vizinhos na última iteração.
- **Raio de vizinhança inicial**: Define o raio de vizinhança do neurônio vencedor na primeira iteração do treinamento.
- **Raio de vizinhança final**: Define o raio de vizinhança do neurônio vencedor na última iteração do treinamento.
- **Função de vizinhança**: Define o tipo de função de vizinhança. Mais detalhes são apresentados na próxima seção.
- **Tipo de seleção**: A partir do conjunto **T** descrito anteriormente, é possível formar $\boldsymbol{N}$ grupos de $\boldsymbol{X}$ vetores gerados a partir das sub-imagens extraídas de apenas uma determinada textura da imagem **I**, sendo que $\boldsymbol{X}$ não precisa ser um valor fixo para todos os $\boldsymbol{N}$ grupos. Assim, são possíveis dois tipos de seleção para o treinamento:
    - Seleção randômica → Os vetores do conjunto **T** são selecionados aleatoriamente para serem apresentados a rede neural durante o treinamento.
    - Seleção randômica por grupo → Um vetor é selecionado aleatoriamente a partir de um dos $\boldsymbol{N}$ grupos do conjunto **T** para ser apresentado a rede neural durante o treinamento. A seleção do grupo é feita sequencialmente, começando pelo primeiro e indo até ao último grupo, recomeçando o ciclo até que o treinamento termine.

5.2.3 – Função de vizinhança

A função de vizinhança determina a porcentagem da taxa de aprendizado que os neurônios dentro do raio de vizinhança receberão durante o treinamento, considerando para isso a sua distância até o neurônio vencedor (BMU – *Best Matching Unit*). Com isso, é possível controlar como os neurônios vizinhos mais próximos e mais distantes deverão ser treinados (JEFF, 2011).

Para este trabalho, a função Gaussiana foi escolhida como a função de vizinhança. Como a rede neural SOM utilizada nesta aplicação forma um reticulado de saída de duas dimensões, foi utilizada a função Gaussiana de duas dimensões, que é dada pela seguinte equação:

$$f(x,y) = Ae^{-\left(\frac{(x-x_o)^2}{2\sigma_x^2} + \frac{(y-y_o)^2}{2\sigma_y^2}\right)}$$

Equação 5.3

Onde o parâmetro A é a altura do pico da curva, x_o e y_o é a posição do centro do pico no eixo x e no eixo y respectivamente, e σ é o desvio padrão, responsável por controlar as extensões do relevo gerado pela função. O gráfico formado pela função gaussiana de duas dimensões é mostrado na Figura 5.2.

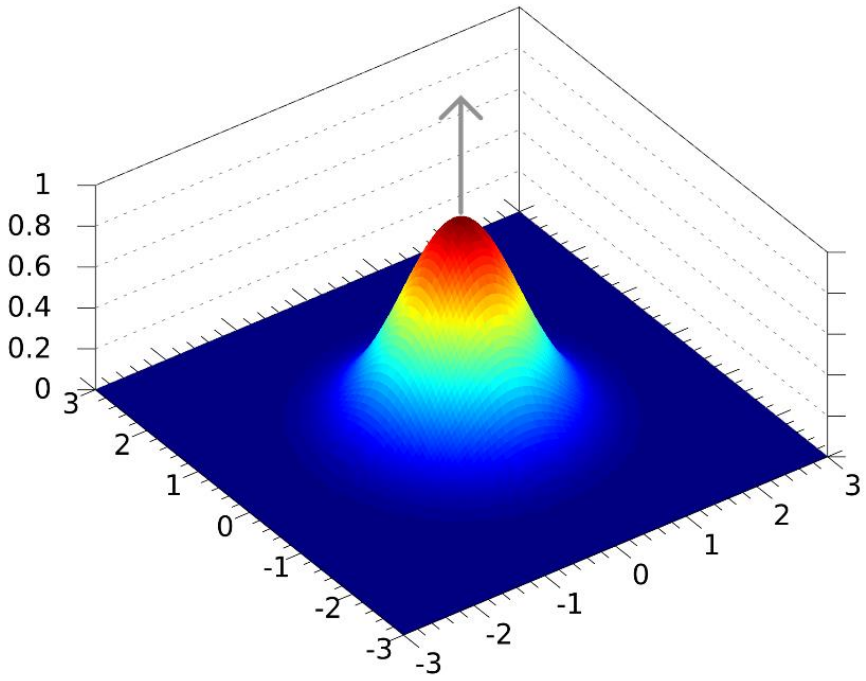

Figura 5.2 – Forma do gráfico de uma função Gaussiana 2D.
Adaptado de: (https://en.wikipedia.org/wiki/Gaussian_function).

A função Gaussiana de duas dimensões possui um único pico variável em sua curva, mas permite que o usuário especifique separadamente valores para a posição e extensões da curva, sendo que a equação não precisa ser simétrica.

Nesta aplicação, o valor do pico utilizado é igual a um, a as coordenadas do centro da curva valem zero para centrá-la na origem. Assim, para os neurônios mais próximos do BMU, a função de vizinhança escolhida retornará um valor próximo de um. Para neurônios mais distantes a função de vizinhança vai retornar valores próximos de zero. A função de vizinhança irá retornar o valor um para o BMU, indicando que ele irá receber a maior parte do treinamento dentre todos os outros neurônios (JEFF, 2011).

5.2.4 – Avaliação

Depois de realizado o treinamento da rede neural SOM, conforme explicado na Seção 5.2.2, a terceira etapa da aplicação consiste em fazer uma avaliação desta rede neural, esta avaliação é feita basicamente em duas fases:

- Experimentação;
- Verificação.

Considerando a imagem **I** com $\boldsymbol{N}$ texturas distintas, na fase **Experimentação**, primeiramente, a partir de uma determinada textura $\boldsymbol{R}$ da imagem **I**, várias sub-imagens são selecionadas e utilizadas no processo ilustrado na Figura 5.3. Estas sub-imagens não possuem dimensões definidas, mas devem fazer parte apenas da textura $\boldsymbol{R}$. O processo ilustrado na Figura 5.3 consiste em aplicar a sub-imagem como entrada da aplicação **Extrator de características**, para que então seja gerado o vetor descritor **F**. O vetor **F** é então aplicado como entrada da rede neural SOM, fazendo com que ela gere uma saída, assim como é ilustrado na Figura 5.3. O algoritmo selecionado na aplicação **Extrator de características** deve ser o mesmo para o qual a rede neural foi adaptada conforme explicado na Seção 5.2.1.

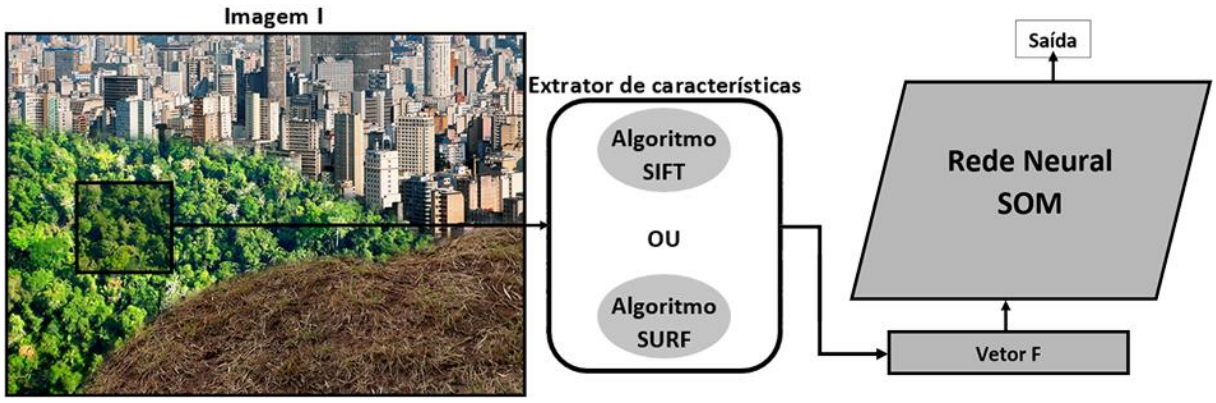

Figura 5.3 – Processo de classificação dos pixels. Fonte: Elaborada pelo autor.

A saída da rede neural neste processo, é o número do neurônio vencedor, ou seja, é o número do neurônio que possui o vetor de pesos mais semelhante ao vetor de entrada **F**, onde este número pode variar de 0 até $\boldsymbol{G} - 1$, onde $\boldsymbol{G}$ é a quantidade de neurônios da rede neural.

Ao repetir este procedimento com todas as sub-imagens, a rede neural terá gerado várias saídas, sendo uma saída para cada vetor **F** gerado a partir de uma sub-imagem de $\boldsymbol{R}$. A aplicação gera então um vetor $\boldsymbol{W}$ contendo todas as saídas geradas, este vetor tem seus elementos ordenados de forma crescente e então é associado a textura $\boldsymbol{R}$.

Todo este processo é repetido para cada textura da imagem **I**, fazendo com que cada textura da imagem **I** tenha um vetor $\boldsymbol{W}$ relacionado a ela. Além disso é gerado um conjunto $\boldsymbol{K}$ de $\boldsymbol{N}$ grupos de padrões de entrada, onde cada padrão de entrada é um vetor **F** gerado a partir de uma sub-imagem extraída de uma das $\boldsymbol{N}$ texturas distintas da imagem **I**, pelo processo ilustrado na Figura 5.3.

Na fase **Verificação**, é calculado o erro da rede neural, que antes da primeira execução desta fase, é igual a 0. Um vetor $\boldsymbol{W}$ pode representar um intervalo caso o valor de sua primeira posição for diferente do valor da última posição, ou pode representar apenas um valor caso contrário. Quando um vetor $\boldsymbol{W}$ representa um intervalo, os elementos da sua primeira e última posição são o extremo mínimo e o extremo máximo do intervalo representado, respectivamente. Considerando isto, nesta fase, primeiramente a aplicação faz uma inspeção para detectar se o valor ou intervalo de valores representado por um vetor $\boldsymbol{W}$ faz intersecção com um valor ou intervalo de valores representado por outro vetor $\boldsymbol{W}$.

Caso não haja qualquer intersecção e seja a primeira vez que a fase **Verificação** é executada, o erro desta rede neural é tido como 0, e isto significa que a rede neural tornou possível a classificação correta das sub-imagens de acordo com as diferentes texturas da imagem **I**, onde as sub-imagens são aquelas selecionadas inicialmente na fase **Experimentação**. Neste caso, a fase **Verificação** não é executada novamente.

Caso tenha ocorrido alguma intersecção, isto significa que a rede neural não conseguiu possibilitar a classificação correta de todas as sub-imagens inicialmente selecionadas. Quando isto acontece, a aplicação atualiza o erro da rede neural. O erro pode ser no máximo 100 e ele é atualizado da seguinte forma:

Ao ser detectada uma intersecção, é considerado o valor da primeira e da última posição de cada vetor $\boldsymbol{W}$ envolvido. Assim para cada vetor $\boldsymbol{W}$ envolvido, é calculada uma diferença $\mathbf{D1}$ e outra diferença $\mathbf{D2}$, sendo que:

$$\mathbf{D1} = \boldsymbol{M} - \boldsymbol{W}_0 \qquad\qquad \mathbf{D2} = \boldsymbol{W}_{\boldsymbol{s}} - \boldsymbol{M}$$

Onde $\boldsymbol{M}$ é a média dos valores dos elementos do vetor $\boldsymbol{W}$, $\boldsymbol{W}_0$ é o valor da primeira posição do vetor $\boldsymbol{W}$ e $\boldsymbol{W}_{\boldsymbol{s}}$ é o valor da última posição do vetor $\boldsymbol{W}$.

Assim, dentre as quatro diferenças calculadas, a aplicação faz uma busca pela maior. Caso a maior diferença seja uma diferença **D1**, o elemento da primeira posição do vetor $\boldsymbol{W}$ usado no cálculo desta diferença é removido. Caso a maior diferença seja uma diferença **D2**, o elemento da última posição do vetor $\boldsymbol{W}$ usado no cálculo desta diferença é removido.

Depois que o padrão é removido, o erro da rede tem o seu valor adicionado com um erro parcial $\boldsymbol{Z}$ dado por:

$$\boldsymbol{Z} = 100/(\boldsymbol{tm} - 3)$$

Equação 5.4

Onde $\boldsymbol{tm}$ é o número de padrões do menor grupo do conjunto $\boldsymbol{K}$.

Caso o erro da rede neural seja ainda menor que 100, a aplicação realiza a fase **Verificação** novamente, considerando as alterações realizadas em cada vetor $\boldsymbol{W}$. Caso o erro da rede neural tenha atingido o valor 100 a fase **Verificação** não é executada novamente.

Esta etapa de avaliação, é necessária para que seja possível alcançar o objetivo deste trabalho, citado na Seção 1.1. Quanto mais sub-imagens forem selecionadas a partir de cada textura da imagem **I** para serem utilizadas nesta etapa, mais preciso e confiável será o seu resultado.

CAPÍTULO 6 – RESULTADOS

Neste capítulo serão relatados os resultados obtidos através de experimentos e testes realizados a partir das aplicações desenvolvidas, e como estes testes e experimentos foram executados, incluindo os dados e parâmetros utilizados.

Para compreender como os resultados que serão mostrados neste capítulo, foram obtidos, considere a imagem **I** como sendo a imagem escolhida para se extrair sub-imagens e classificá-las de acordo com as diferentes texturas da imagem **I**.

Para alcançar os objetivos deste trabalho, foram executados alguns testes, para possibilitar a avaliação e a comparação entre o desempenho dos algoritmos SIFT e SURF na classificação das sub-imagens da imagem **I**.

6.1 – Parâmetros

A seguir, será explicado como foram definidos os valores para alguns parâmetros, durante a execução dos testes deste trabalho.

O parâmetro **número de iterações**, é utilizado no treinamento da rede neural, conforme explicado na Seção 5.2.2. O desempenho dos algoritmos SIFT e SURF não é bom caso seja escolhido um alto valor para este parâmetro. Aumentar muito o seu valor, faz com que o desempenho dos algoritmos vá se reduzindo cada vez mais. Os testes realizados mostraram que para possibilitar um alto senão o melhor desempenho dos algoritmos SIFT e SURF, o valor deste parâmetro deve ser dado por $40 * \boldsymbol{n}$, onde $\boldsymbol{n}$ é o número de texturas distintas da imagem **I**.

Considerando que nos testes realizados para obter os resultados deste trabalho, foram utilizadas apenas redes neurais quadradas, o parâmetro **dimensão da rede neural**, aqui denotado por $\boldsymbol{D}$, se refere ao tamanho dos lados da rede neural. Assim, dado este parâmetro $\boldsymbol{D}$, a rede neural instanciada na etapa de inicialização (Seção 5.2.1), terá dimensões $\boldsymbol{D} \times \boldsymbol{D}$. Aumentar o valor deste parâmetro implica no aumento do tempo na etapa de treinamento (Seção 5.2.2) e na etapa de avaliação (Seção 5.2.4). Os testes realizados mostraram que para alcançar um bom desempenho dos algoritmos SIFT e SURF, o valor deste parâmetro $\boldsymbol{D}$ deve ser dado por:

$$\boldsymbol{D = c * n}$$

Equação 5.5

Onde $\boldsymbol{c} = 5$, e $\boldsymbol{n}$ é o número de texturas distintas da imagem **I**. De acordo com os testes realizados, aumentar o valor de $\boldsymbol{c}$, resultará em um pequeno aumento no desempenho dos algoritmos, no entanto, pode ser inviável devido ao grande aumento no tempo de processamento das etapas de treinamento (Seção 5.2.2) e avaliação (Seção 5.2.4) da rede neural.

O parâmetro **raio de vizinhança inicial**, é utilizado no treinamento da rede neural, conforme explicado na Seção 5.2.2. Para possibilitar o melhor desempenho dos algoritmos SIFT e SURF, o valor ideal deste parâmetro vai depender do parâmetro **dimensão da rede neural**, que foi abordado anteriormente, e também da imagem **I**. Ao contrário da dimensão da rede neural, este parâmetro influencia fortemente o desempenho dos algoritmos SIFT e SURF.

Os parâmetros **taxa de aprendizado inicial**, **taxa de aprendizado final** e **raio de vizinhança final** são utilizados no treinamento da rede neural, conforme explicado na Seção 5.2.2. Os valores fixos definidos para estes parâmetros foram respectivamente 1, 0,005 e 1.

O parâmetro **tipo de seleção** é utilizado no treinamento da rede neural, conforme explicado na Seção 5.2.2. Para este parâmetro, foi escolhida a opção de seleção randômica por grupo, pois os algoritmos SIFT e SURF apresentaram um desempenho significativamente maior que o desempenho obtido com a opção de seleção randômica.

6.2 – Fatores

Devido a inicialização aleatória dos pesos dos neurônios da rede neural, conforme descrito na Seção 5.2.1, caso sejam instanciadas várias redes neurais com dimensões idênticas, adaptadas para o mesmo algoritmo e treinadas com o mesmo conjunto **T** e os mesmos parâmetros, os quais são abordados na Seção 5.2.2, cada uma destas redes neurais instanciadas poderá apresentar um erro diferente quando passarem pela etapa de avaliação explicada na Seção 5.2.4. Considerando isto, para alcançar os objetivos citados na Seção 1.1, foi necessário instanciar várias redes neurais, a fim de calcular dois fatores, que serão utilizados para descrever o desempenho de cada algoritmo.

O primeiro fator é o **acerto total**, e ele indica a média de redes neurais que tiveram erro igual a zero ao passar pela etapa de avaliação (Seção 5.2.4), sendo que quanto maior for o valor deste fator, maior será o desempenho do algoritmo.

O segundo fator é o **erro geral** e ele indica o erro médio gerado pelas redes neurais ao passar pela etapa de avaliação (Seção 5.2.4), sendo que quanto menor for o seu valor, maior será o desempenho do algoritmo.

Assim, considerando um determinado algoritmo **X**, onde **X** pode ser o algoritmo SIFT ou o algoritmo SURF, o cálculo dos fatores **acerto total** e **erro geral** para o algoritmo **X**, é feito da seguinte forma:

Primeiramente é executado um ciclo **P** de dez iterações, onde em cada iteração, o procedimento **K** é executado.

No procedimento **K**, primeiramente é executado um ciclo **O** de cinquenta iterações, onde em cada iteração, os seguintes passos são executados:

1. Uma rede neural é instanciada através da etapa de inicialização, conforme explicado na Seção 5.2.1. Nesta etapa, os parâmetros utilizados são definidos conforme explicado na Seção 6.1, sendo que a rede neural deve ser adaptada para o algoritmo **X**.
2. A rede neural passa então pela etapa de treinamento, que é abordada na Seção 5.2.2.
3. Depois a rede neural passa pela etapa de avaliação, que é abordada na Seção 5.2.4, fazendo com que seja calculado o erro da rede neural SOM instanciada.

Ao final de todas as iterações do ciclo **O**, o último passo do procedimento **K**, consiste em calcular os valores $\boldsymbol{A}$ e $\boldsymbol{E}$, onde $\boldsymbol{A}$ é o número de iterações de **O** em que o erro calculado para a rede neural foi igual a zero, e $\boldsymbol{E}$ é a média de todos os erros calculados ao longo de todas as iterações de **O**.

Assim, em cada iteração do ciclo **P**, é gerado um valor $\boldsymbol{A}$ e um valor $\boldsymbol{E}$, pelo procedimento **K**. Ao final de todas as iterações do ciclo **P**, o fator **acerto total** é calculado como sendo a média de todos os valores $\boldsymbol{A}$ gerados ao longo de todas as iterações de **P**, e o fator **erro geral** é calculado como sendo a média de todos os valores $\boldsymbol{E}$ gerados ao longo de todas as iterações de **P**.

6.3 – Método

Nesta seção será explicado o método utilizado na execução de cada um dos testes realizados, os quais serão abordados nas próximas seções.

Este método é executado para um determinado algoritmo **X**, onde **X** pode ser o algoritmo SIFT ou o algoritmo SURF. Assim, este método pode ser resumido nos seguintes passos:

1. É selecionada uma determinada imagem **I** com diferentes texturas;
2. São extraídas cinco sub-imagens a partir de cada textura da imagem **I**, para que elas possam ser utilizadas na etapa de treinamento, assim como é explicado na Seção 5.2.2;
3. São extraídas dez sub-imagens a partir de cada textura da imagem **I**, para que elas possam ser utilizadas na etapa de avaliação, assim como é explicado na Seção 5.2.4;
4. São definidos todos os parâmetros conforme explicado na Seção 6.1, com base na imagem **I** selecionada no passo 1, sendo que o parâmetro **raio de vizinhança inicial**, recebe um valor inicial de 2.
5. Depois, é iniciado um ciclo **U** de ***N*** iterações, onde em cada iteração são executados os seguintes procedimentos.
    - Os fatores **acerto total** e **erro geral** são calculados para o algoritmo **X**, conforme explicado na Seção 6.2, utilizando os parâmetros e as sub-imagens definidas nos passos 2, 3 e 4;
    - O parâmetro **raio de vizinhança inicial**, abordado na Seção 6.1, tem seu valor incrementado em uma unidade.

    Assim, neste passo, o único parâmetro que é alterado ao longo das iterações do ciclo **U**, é o **raio de vizinhança inicial**, os outros parâmetros são os mesmos em todas as iterações do ciclo **U**. A imagem **I** e as sub-imagens definidas nos passos 2 e 3, também permanecem inalteradas durante as iterações do ciclo **U**. Portanto, em cada iteração de **U**, são calculados os fatores **acerto total** e **erro geral** para um determinado **raio de vizinhança inicial**.

6.4 – Teste 1

Considere o método **T1** como sendo o método abordado na Seção 6.3, onde os seus passos foram executados da seguinte forma:

1. No primeiro passo foi selecionada como a imagem **I**, uma imagem sintética formada por duas texturas distintas, assim como pode ser observado na Figura 6.1.

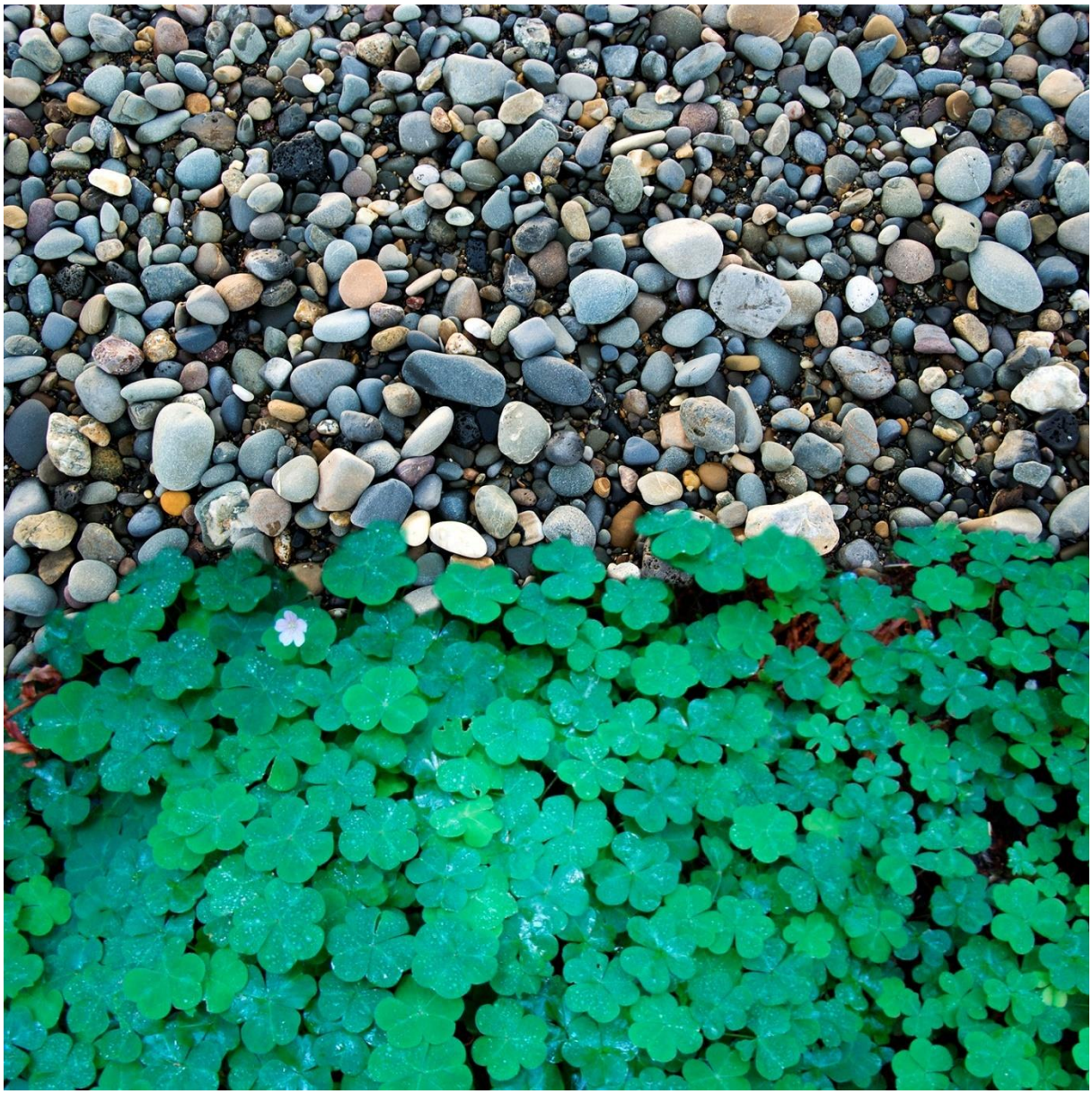

Figura 6.1 – Primeira imagem sintética utilizada na experimentação. Fonte: Elaborada pelo autor.

2. No segundo passo, as sub-imagens extraídas foram aquelas mostradas nas figuras 6.2 e 6.3.

Figura 6.2 – Sub-imagens da textura superior utilizadas no treinamento. Fonte: Elaborada pelo autor.

Figura 6.3 – Sub-imagens da textura inferior utilizadas no treinamento. Fonte: Elaborada pelo autor.

3. No terceiro passo, as sub-imagens extraídas foram aquelas mostradas nas figuras 6.4 e 6.5.

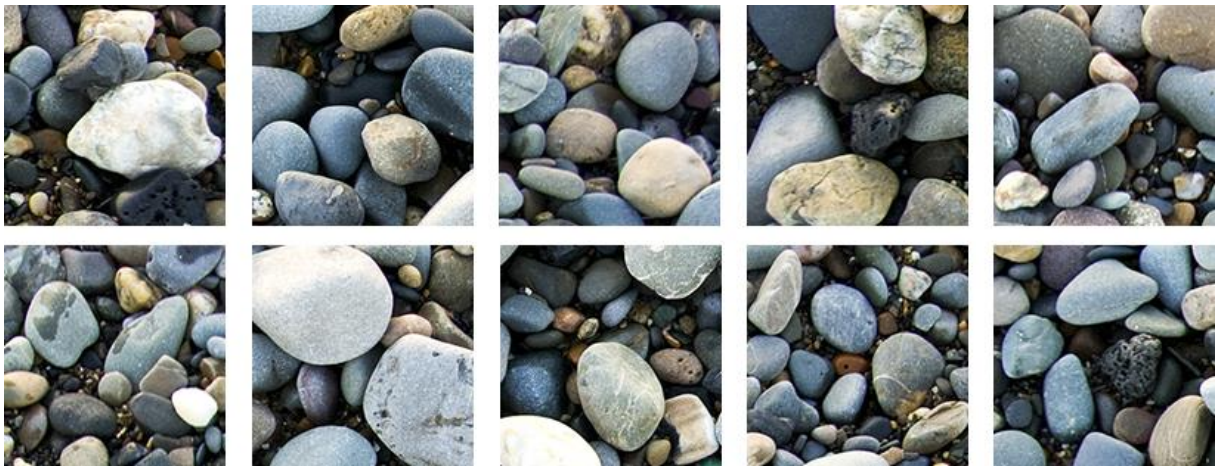

Figura 6.4 – Sub-imagens da textura superior utilizadas na avaliação. Fonte: Elaborada pelo autor.

Figura 6.5 – Sub-imagens da textura inferior utilizadas na avaliação. Fonte: Elaborada pelo autor.

4. O quarto passo é executado.

5. No quinto passo foi executado o ciclo **U** com 98 iterações.

Ao executar o método **T1** para o algoritmo SIFT e também para o algoritmo SURF, foram obtidos para cada um destes algoritmos, 98 resultados para os fatores **acerto total** e **erro geral**, onde cada um destes resultados está relacionado com um determinado **raio de vizinhança inicial**, conforme explicado na Seção 6.3. Os resultados obtidos para o algoritmo SIFT estão representados no gráfico da Figura 6.6, enquanto que os resultados obtidos para o algoritmo SURF estão representados no gráfico da Figura 6.7.

Assim, para avaliar o desempenho de cada algoritmo neste teste, foi selecionado dentre os 98 resultados para o fator **acerto total**, o maior valor, e dentre os 98 resultados para o fator **erro geral**, o menor valor. Cada um destes valores selecionados, é mostrado em destaque nos gráficos ilustrados nas figuras 6.6 e 6.7.

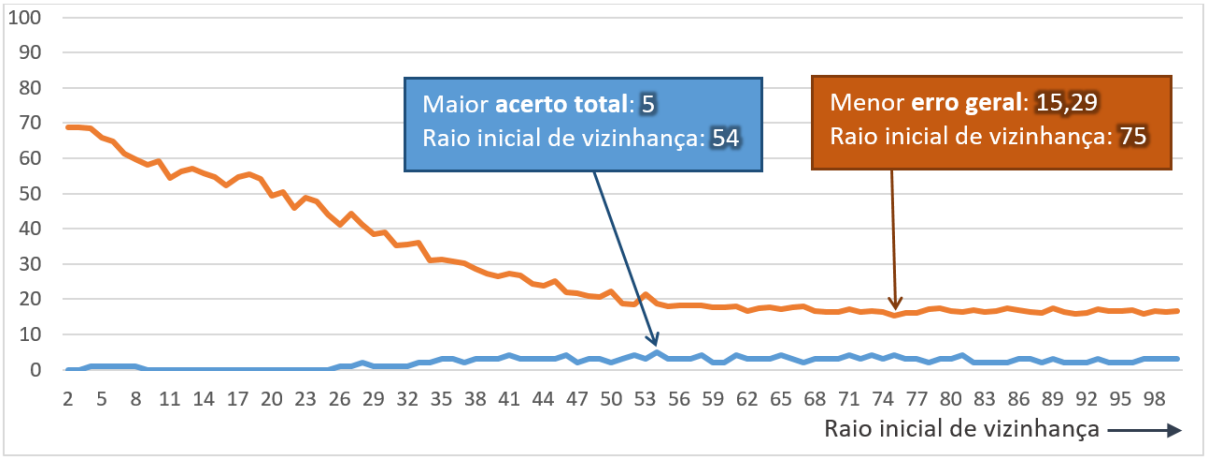

Figura 6.6 – Desempenho do algoritmo SIFT no Teste 1. Fonte: Elaborada pelo autor.

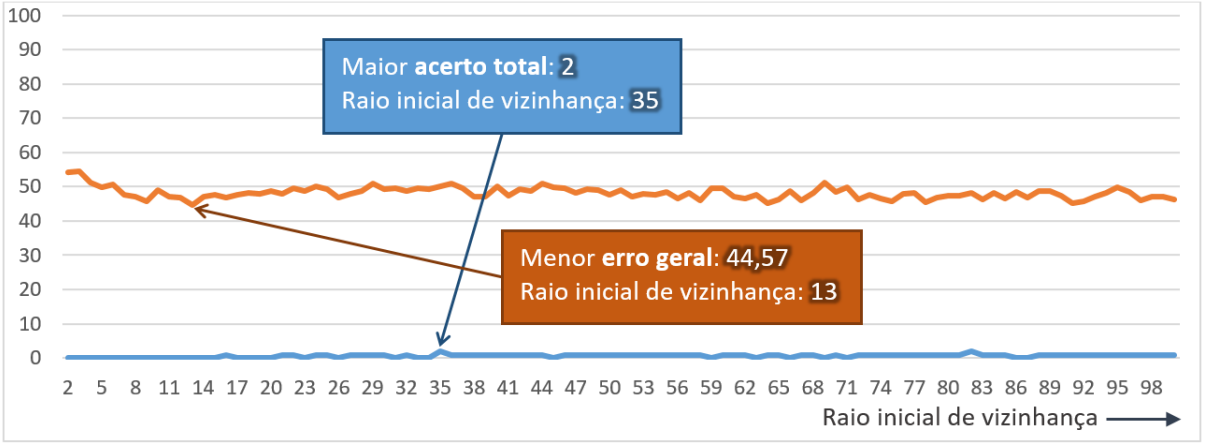

Figura 6.7 – Desempenho do algoritmo SURF no Teste 1. Fonte: Elaborada pelo autor.

6.5 – Teste 2

Considere o método **T2** como sendo o método abordado na Seção 6.3, onde os seus passos foram executados da seguinte forma:

1. No primeiro passo foi selecionada como a imagem **I**, uma imagem sintética formada por três texturas distintas, assim como pode ser observado na Figura 6.8.

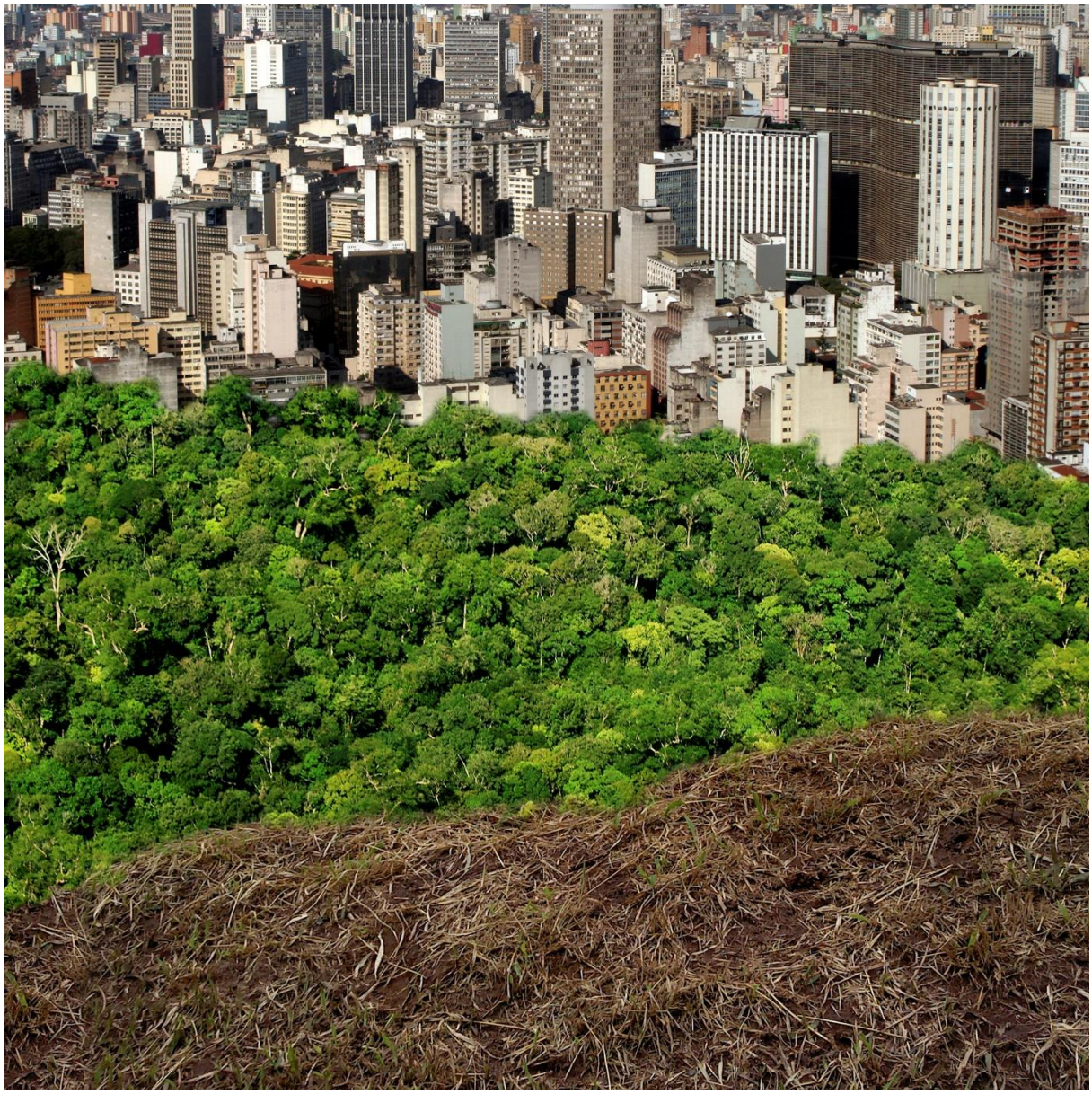

Figura 6.8 – Segunda imagem sintética utilizada na experimentação. Fonte: Elaborada pelo autor.

2. No segundo passo, as sub-imagens extraídas foram aquelas mostradas nas figuras 6.9, 6.10 e 6.11.

Figura 6.9 – Sub-imagens da primeira textura utilizadas no treinamento. Fonte: Elaborada pelo autor.

Figura 6.10 – Sub-imagens da segunda textura utilizadas no treinamento. Fonte: Elaborada pelo autor.

Figura 6.11 – Sub-imagens da terceira textura utilizadas no treinamento. Fonte: Elaborada pelo autor.

3. No terceiro passo, as sub-imagens extraídas foram aquelas mostradas nas figuras 6.12, 6.13 e 6.14.

Figura 6.12 – Sub-imagens da primeira textura utilizadas na avaliação. Fonte: Elaborada pelo autor.

Figura 6.13 – Sub-imagens da segunda textura utilizadas na avaliação. Fonte: Elaborada pelo autor.

Figura 6.14 – Sub-imagens da terceira textura utilizadas na avaliação. Fonte: Elaborada pelo autor.

4. O quarto passo é executado.

5. No quinto passo foi executado o ciclo **U** com 98 iterações.

Ao executar o método **T2** para o algoritmo SIFT e também para o algoritmo SURF, foram obtidos para cada um destes algoritmos, 98 resultados para os fatores **acerto total** e **erro geral**, onde cada um destes resultados está relacionado com um determinado **raio de vizinhança inicial**, conforme explicado na Seção 6.3. Os resultados obtidos para o algoritmo SIFT estão representados no gráfico da Figura 6.15, enquanto que os resultados obtidos para o algoritmo SURF estão representados no gráfico da Figura 6.16.

Assim, para avaliar o desempenho de cada algoritmo neste teste, foi selecionado dentre os 98 resultados para o fator **acerto total**, o maior valor, e dentre os 98 resultados para o fator **erro geral**, o menor valor. Cada um destes valores selecionados, é mostrado em destaque nos gráficos ilustrados nas figuras 6.15 e 6.16.

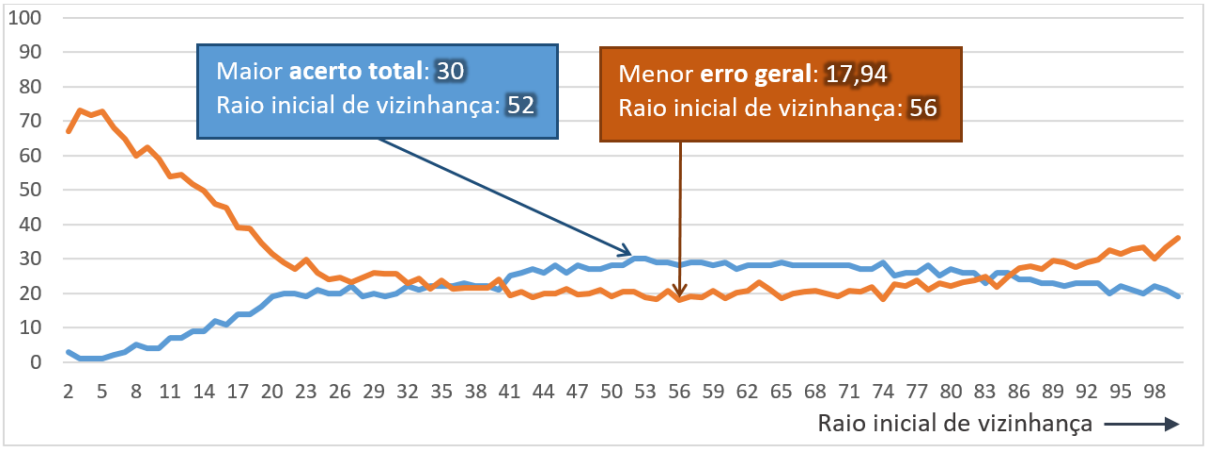

Figura 6.15 – Desempenho do algoritmo SIFT no Teste 2. Fonte: Elaborada pelo autor.

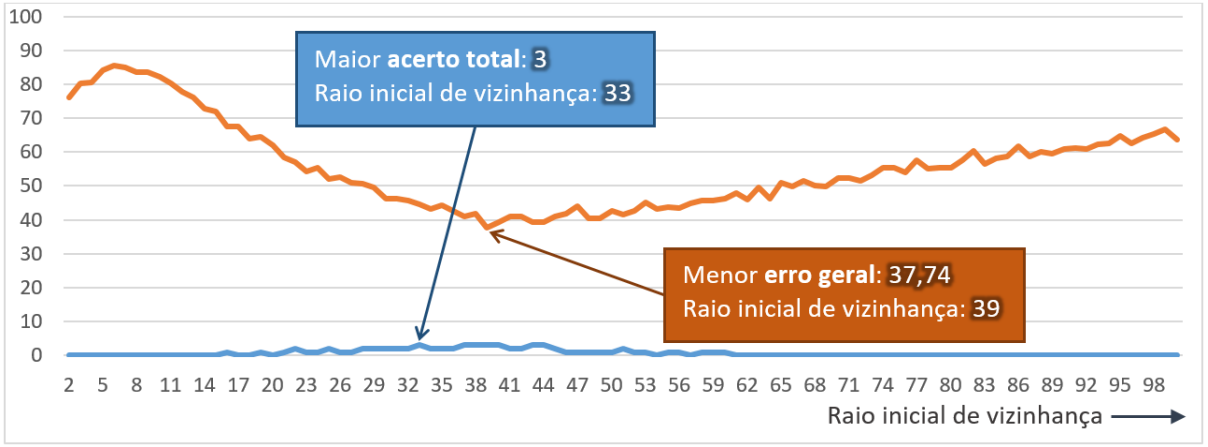

Figura 6.16 – Desempenho do algoritmo SURF no Teste 2. Fonte: Elaborada pelo autor.

6.6 – Teste 3

Considere o método **T3** como sendo o método abordado na Seção 6.3, onde os seus passos foram executados da seguinte forma:

1. No primeiro passo foi selecionada como a imagem **I**, uma imagem sintética formada por quatro texturas distintas, assim como pode ser observado na Figura 6.17.

Figura 6.17 – Terceira imagem sintética utilizada na experimentação. Fonte: Elaborada pelo autor.

2. No segundo passo, as sub-imagens extraídas foram aquelas mostradas nas figuras 6.18, 6.19, 6.20 e 6.21.

Figura 6.18 – Sub-imagens da primeira textura superior utilizadas no treinamento. Fonte: Elaborada pelo autor.

Figura 6.19 – Sub-imagens da segunda textura superior utilizadas no treinamento. Fonte: Elaborada pelo autor.

Figura 6.20 – Sub-imagens da primeira textura inferior utilizadas no treinamento. Fonte: Elaborada pelo autor.

Figura 6.21 – Sub-imagens da segunda textura inferior utilizadas no treinamento. Fonte: Elaborada pelo autor.

3. No terceiro passo, as sub-imagens extraídas foram aquelas mostradas nas figuras 6.22, 6.23, 6.24 e 6.25.

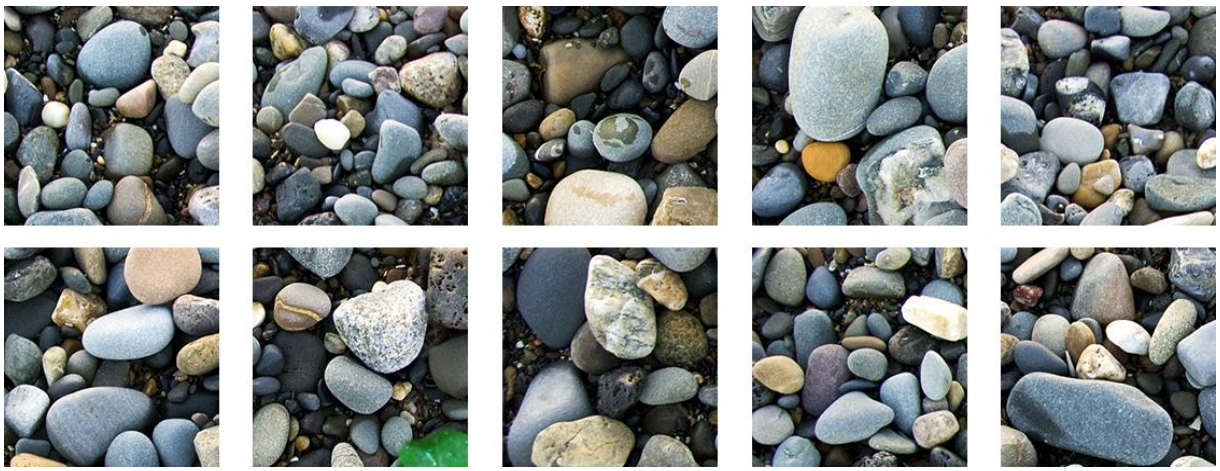

Figura 6.22 – Sub-imagens da primeira textura superior utilizadas na avaliação. Fonte: Elaborada pelo autor.

Figura 6.23 – Sub-imagens da segunda textura superior utilizadas na avaliação. Fonte: Elaborada pelo autor.

Figura 6.24 – Sub-imagens da primeira textura inferior utilizadas na avaliação. Fonte: Elaborada pelo autor.

Figura 6.25 – Sub-imagens da segunda textura inferior utilizadas na avaliação. Fonte: Elaborada pelo autor.

4. O quarto passo é executado.

5. No quinto passo foi executado o ciclo **U** com 98 iterações.

Ao executar o método **T3** para o algoritmo SIFT e também para o algoritmo SURF, foram obtidos para cada um destes algoritmos, 98 resultados para os fatores **acerto total** e **erro geral**, onde cada um destes resultados está relacionado com um determinado **raio de vizinhança inicial**, conforme explicado na Seção 6.3. Os resultados obtidos para o algoritmo SIFT estão representados no gráfico da Figura 6.26, enquanto que os resultados obtidos para o algoritmo SURF estão representados no gráfico da Figura 6.27.

Assim, para avaliar o desempenho de cada algoritmo neste teste, foi selecionado dentre os 98 resultados para o fator **acerto total**, o maior valor, e dentre os 98 resultados para o fator **erro geral**, o menor valor. Cada um destes valores selecionados, é mostrado em destaque nos gráficos ilustrados nas figuras 6.26 e 6.27.

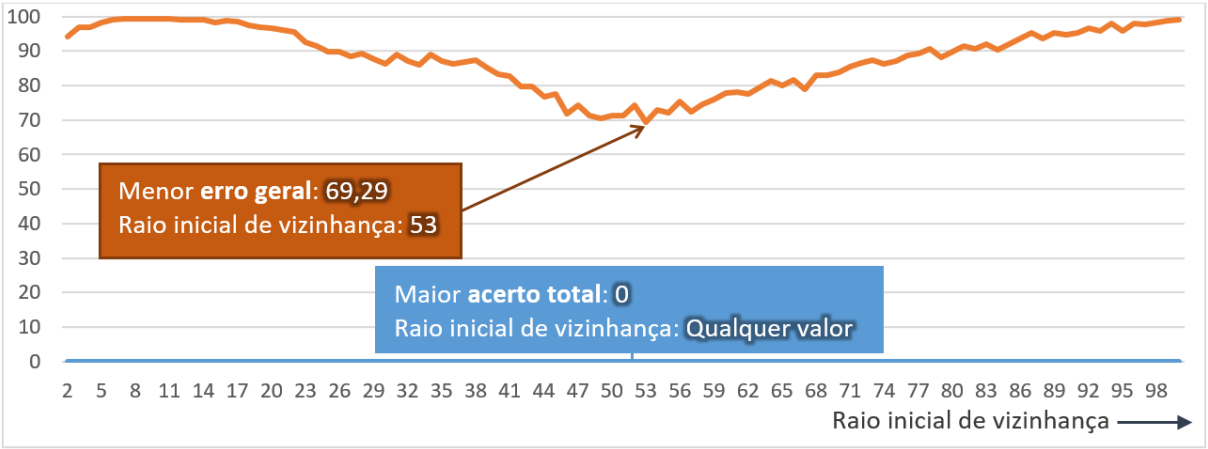

Figura 6.26 – Desempenho do algoritmo SIFT no Teste 3. Fonte: Elaborada pelo autor.

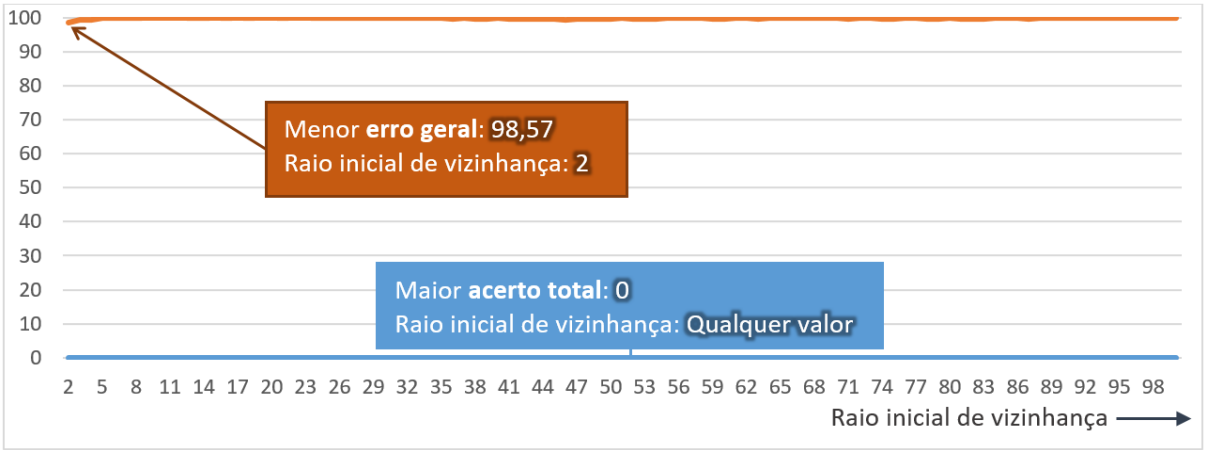

Figura 6.27 – Desempenho do algoritmo SURF no Teste 3. Fonte: Elaborada pelo autor.

CAPÍTULO 7 – CONCLUSÕES

Muitos métodos vêm sendo desenvolvidos ao longo do tempo para descrever texturas (Seções 1.3 e 2.3). Estes métodos são geralmente utilizados para possibilitar a classificação e a segmentação de imagens por discriminação de textura. A grande dificuldade encontrada por muitos destes métodos está ao encontrar texturas com comportamentos aleatórios e complexos, devido muitas vezes a natureza complexa dos objetos similares que a forma. Muitos destes métodos demandam um alto custo computacional e muitas vezes não são muito flexíveis com algumas texturas mais complexas, necessitando geralmente de uma configuração de parâmetros específica para determinadas texturas, que em muitas das vezes é difícil de se descobrir.

Foi apresentado neste trabalho a proposta de um processo de classificação de sub-imagens por discriminação de textura, com o objetivo de auxiliar na segmentação de imagens, conforme é explicado na Seção 2.4.

O objetivo da pesquisa realizada foi avaliar a capacidade da utilização dos algoritmos SIFT e SURF neste processo, para fazer a descrição de texturas com padrões aleatórios de objetos similares e complexos, como as texturas que podem ser observadas nas imagens utilizadas nos testes realizados (Seções 6.4, 6.5 e 6.6). Para fazer a classificação das sub-imagens foi utilizada uma rede neural artificial SOM, sendo que seus parâmetros foram configurados de acordo com os valores ideais encontrados durante a realização de experimentos.

Os resultados mostraram que dependendo do cenário, é possível obter uma classificação satisfatória das sub-imagens, utilizando o processo descrito neste trabalho. Embora os resultados obtidos tenham mostrado que o processo pode apresentar erros, eles também mostraram o seu grande potencial que pode ser explorado.

De acordo com os testes e experimentos realizados, para obter o maior desempenho dos algoritmos neste processo de classificação, os parâmetros utilizados, devem ser configurados conforme explicado na Seção 6.1. A escolha errada de valores para estes parâmetros pode tornar o processo de classificação apresentado neste trabalho, totalmente ineficiente e inviável.

A invariância a rotação e a escala apresentada pelos algoritmos SIFT e SURF (LOWE, 2004) (BAY et al, 2006), permite que a classificação das sub-imagens seja feita mesmo quando as texturas possuam variações de escala e rotação. Além disso, os algoritmos SIFT e SURF apresentam um certo grau de resistência à ruídos e variações de iluminação (LOWE, 2004) (BAY et al, 2006) (IŞIK et al, 2015), fazendo com que o processo proposto neste trabalho possibilite a classificação correta das sub-imagens, mesmo em um cenário em que as texturas apresentem variações de iluminação e/ou ruídos.

Uma desvantagem do método proposto neste trabalho, é que não é possível saber qual é o valor do raio inicial de vizinhança que possibilita o maior desempenho dos algoritmos na classificação das sub-imagens, assim como é explicado na Seção 6.1. Isso pode levar a necessidade da realização de testes para descobrir qual é o valor ideal para o raio inicial de vizinhança, conforme foi realizado na Seção 6.3.

Embora neste trabalho não tenha sido definido as dimensões das sub-imagens utilizadas nas etapas de treinamento (Seção 5.2.2) e avaliação (Seção 5.2.4), estas dimensões podem influenciar o desempenho dos algoritmos, pois quanto menores forem as dimensões, menor o número de características que os algoritmos poderão extrair, o que poderia diminuir drasticamente o desempenho dos algoritmos. Desta forma, quanto maiores as dimensões das sub-imagens, maior o número de características que os algoritmos poderão extrair, o que poderia aumentar consideravelmente o desempenho dos algoritmos.

Foi possível observar nos resultados que o algoritmo SIFT apresentou um maior desempenho em relação ao algoritmo SURF, apresentando uma vantagem significativa em todos os testes realizados (Seção 6.4, 6.5 e 6.6). Algo similar pode ser observado em (LEE et al, 2014), onde em alguns testes, o algoritmo SURF apresentou um desempenho inferior ao desempenho do algoritmo SIFT.

Considerando a imagem **I** como a imagem escolhida para se extrair sub-imagens e classificá-las com o processo descrito neste trabalho, o número de texturas distintas da imagem **I** influenciou significativamente o desempenho dos algoritmos. Assim como pode ser observado nos testes realizados, quanto maior o número de texturas distintas da imagem **I**, menor tende a ser o desempenho dos algoritmos.

Para trabalhos futuros, poderia ser feita uma busca pelo aumento do desempenho dos algoritmos de uma forma geral e principalmente em situações onde a imagem **I** apresente uma grande quantidade de texturas diferentes. Algumas opções para pesquisa seriam:

- Explorar mais os parâmetros abordados nas seções 5.1.1 e 6.1, e as relações que possam existir entre eles e/ou com a imagem **I**;
- Adicionar a informação de cor ou outras informações ao vetor **F** durante a sua formação, conforme explicado na Seção 5.1.

Em uma outra opção de trabalho futuro, poderiam ser utilizados no processo de classificação proposto neste trabalho, outros algoritmos detectores e descritores de pontos chave, como por exemplo, alguns dos algoritmos citados na Seção 3.1.

Outro possível trabalho futuro, é a implementação do algoritmo de consciência, conforme explicado na Seção 4.8.3, já que este algoritmo de consciência busca minimizar o impacto da inicialização aleatória dos pesos dos neurônios, nos agrupamentos feitos na rede neural SOM durante a fase de treinamento. Este impacto se mostrou como uma desvantagem do método de classificação apresentado neste trabalho, pois devido a ele, a primeira rede neural instanciada nem sempre será aquela que possibilitará o melhor desempenho dos algoritmos. Assim, o impacto desta inicialização aleatória causou a necessidade de se instanciar várias redes neurais, para que os objetivos deste trabalho pudessem ser alcançados, assim como é explicado na Seção 6.2.

Seria possível também em um trabalho futuro, fazer uma comparação do processo de classificação proposto neste trabalho com outros métodos de classificação de texturas.

Assim, com o desenvolvimento deste trabalho, foi possível perceber que melhorias podem ser feitas em trabalhos futuros, para que o processo de classificação apresentado neste trabalho, possa ser melhor aproveitado no desenvolvimento de métodos e técnicas de segmentação de imagens que possuem diferentes texturas de comportamento aleatório e complexo, possibilitando a implementação de sistemas e aplicações autônomas.

REFERÊNCIAS BIBLIOGRÁFICAS

AFFONSO, Gustavo Sousa. **Mapas Auto-organizáveis de Kohonen (SOM) aplicados na avaliação dos parâmetros da qualidade da água**. Tese de Doutorado. UNIVERSIDADE DE SÃO PAULO, São Paulo. 2011.

ALAHI, Alexandre; ORTIZ, Raphael; VANDERGHEYNST, Pierre. **Freak: Fast retina keypoint**. In: Computer vision and pattern recognition (CVPR), 2012 IEEE conference on. Ieee, p. 510-517. 2012.

ALCANTARILLA, Pablo Fernández; BARTOLI, Adrien; DAVISON, Andrew J. **KAZE features**. In: European Conference on Computer Vision. Springer Berlin Heidelberg, p. 214-227. 2012.

BALAN, André Guilherme Ribeiro. **Técnicas de segmentação de imagens aéreas para contagem de população de aves**. Tese de Doutorado. Instituto de Ciências Matemáticas e de Computação. 2003.

BAUER, Johannes; SÜNDERHAUF, Niko; PROTZEL, Peter. **Comparing several implementations of two recently published feature detectors**. IFAC Proceedings Volumes, v. 40, n. 15, p. 143-148. 2007.

BAY, Herbert et al. **Speeded-up robust features (SURF)**. Computer vision and image understanding, v. 110, n. 3, p. 346-359. 2008.

BEJA, Pedro Filipe Hortelão. **Segmentação de imagens de elastografia mamária**. Tese de Doutorado. 2013.

BELUCO, Adriano. **Classificação de imagens de sensoriamento remoto baseada em textura por redes neurais**. 2002.

BRAGA, A. de P.; CARVALHO, ACPLF; LUDERMIR, Teresa Bernarda. **Redes neurais artificiais: teoria e aplicações**. 2ª Edição, Livros Técnicos e Científicos. 2007.

BROWN, Matthew; LOWE, David G. **Invariant Features from Interest Point Groups**. In: BMVC. 2002.

CALONDER, Michael et al. **Brief: Binary robust independent elementary features**. In: European conference on computer vision. Springer Berlin Heidelberg, p. 778-792. 2010.

COMANICIU, Dorin; MEER, Peter. **Mean shift: A robust approach toward feature space analysis**. Pattern Analysis and Machine Intelligence, IEEE Transactions on, v. 24, n. 5, p. 603-619. 2002.

COUTO, Leandro Nogueira. **Sistema para localização robótica de veículos autônomos baseado em visão computacional por pontos de referência**. Tese de Doutorado. Universidade de São Paulo. 2012.

CYBENKO, George. **Approximation by superpositions of a sigmoidal function**. Mathematics of control, signals and systems, v. 2, n. 4, p. 303-314. 1989.

DE FREITAS, Emannuel Diego G.; CORREIA, Suzete Elida N.; REGIS, Carlos Danilo M. **Classificação de Textura em Imagens com Redes Neurais Artificiais para Segmentação de Regiões de AVCi em Tomografias Computadorizadas**. XII Congresso Brasileiro de Inteligência Computacional. 2015.

DEMPSTER, Arthur P.; LAIRD, Nan M.; RUBIN, Donald B. **Maximum likelihood from incomplete data via the EM algorithm**. Journal of the royal statistical society. Series B (methodological), p. 1-38. 1977.

DESIENO, Duane. **Adding a conscience to competitive learning**. In: Neural Networks, IEEE International Conference on. IEEE, p. 117-124. 1988.

DINIZ, Wendell Fioravante da Silva et al. **Acionamento de dispositivos robóticos através de interface natural em realidade aumentada**. 2012.

DUDA, Richard O.; HART, Peter E.; STORK, David G. **Pattern classification**. 2nd. Edition. New York. 2001.

GOMES, Daniel de Filgueiras. **Visão computacional e segmentação de imagens por discriminação de textura**. 2009.

GONZÁLES, Giancarlo Luis Gómez. **Aplicação da Técnica SIFT para Determinação de Campos de Deformações de Materiais**. Tese de Doutorado. PUC-Rio. 2010.

GONZALEZ, Rafael C.; WOODS, Richard E. **Processamento Digital de Imagens**. 3ª Edição. São Paulo, Pearson, 624p. 2011.

HAYKIN, Simon S. **Neural networks: a comprehensive foundation**. Prentice-Hall, 2nd Edition. 1999.

HIRATA JR, Roberto. **Segmentação de imagens por morfologia matemática**. Tese de Doutorado. Instituto de Matemática e Estatística da Universidade de São Paulo. 1997.

IŞIK, Şahin; ÖZKAN, Kemal. **A comparative evaluation of well-known feature detectors and descriptors**. International Journal of Applied Mathematics, Electronics and Computers, v. 3, n. 1, p. 1-6. 2015.

IEEE. **IEEE Standards Glossary of Image Processing and Pattern Recognition Terminology**. INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. 1990.

JEFF, Heaton. **Programming Neural Networks with Encog3 in Java**. 2011.

JULESZ, Bela; BERGEN, James R. **Human factors and behavioral science: Textons, the fundamental elements in preattentive vision and perception of textures**. The Bell system technical journal, v. 62, n. 6, p. 1619-1645. 1983.

KOHONEN, Teuvo. **Self-organising and associative memory**. Springer Series on Information Sciences (Springer-Verlag). 1989.

LEE, Jia-Hong; WU, Mei-Yi; KUO, Hsien-Tsung. **Evaluation of Robust Feature Descriptors for Texture Classification**. World Academy of Science, Engineering and Technology,

International Journal of Computer, Electrical, Automation, Control and Information Engineering, v. 8, n. 7, p. 1276-1280. 2014.

LEUTENEGGER, Stefan; CHLI, Margarita; SIEGWART, Roland Y. **BRISK: Binary robust invariant scalable keypoints**. In: 2011 International conference on computer vision. IEEE, p. 2548-2555. 2011.

LIBERMAN, Felipe. **Classificação de Imagens Digitais por Textura usando Redes Neurais**. Tese de Doutorado. UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL. 1997.

LOWE, David G. **Distinctive image features from scale-invariant keypoints**. International journal of computer vision, v. 60, n. 2, p. 91-110. 2004.

LOWE, David G. **Object recognition from local scale-invariant features**. In: Computer vision. The proceedings of the seventh IEEE international conference on. Ieee, p. 1150-1157. 1999.

MACQUEEN, James et al. **Some methods for classification and analysis of multivariate observations**. In: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, p. 281-297. 1967.

MAIA, J. G. R. **Detecção e Reconhecimento de Objetos usando Descritores Locais**. PhD in Bioinformatics, Universidade Federal do Ceara, Biomathematics Group, R. Qta. Grande, v. 6, p. 2780-156. 2010.

MEYER, Fernand. **Topographic distance and watershed lines**. Signal processing, v. 38, n. 1, p. 113-125. 1994.

MIKOLAJCZYK, Krystian; SCHMID, Cordelia. **A performance evaluation of local descriptors**. IEEE transactions on pattern analysis and machine intelligence, v. 27, n. 10, p. 1615-1630. 2005.

NEUBECK, Alexander; VAN GOOL, Luc. **Efficient non-maximum suppression**. In: 18th International Conference on Pattern Recognition (ICPR'06). Vol. 3. IEEE, p. 850-855. 2006.

NISTÉR, David; STEWÉNIUS, Henrik. **Linear time maximally stable extremal regions**. In: European Conference on Computer Vision. Springer Berlin Heidelberg, p. 183-196. 2008.

OHNISHI, Yuji de Oliveira. **Método baseado em processamento digital de imagens para diagnóstico precoce de micro-estruturas dentárias**. Tese de Doutorado. UNIVERSIDADE FEDERAL DE SÃO CARLOS. 2005.

PARKER, Jim R. **Algorithms for image processing and computer vision**. John Wiley & Sons. 2010.

ROSENBLATT, Frank. **Principles of neurodynamics**. 1962.

ROSHOLM, Anders. **Statistical methods for segmentation and classification of images**. 1997.

ROSTEN, Edward; DRUMMOND, Tom. **Machine learning for high-speed corner detection**. In: European conference on computer vision. Springer Berlin Heidelberg, p. 430-443. 2006.

RUBLEE, Ethan et al. **ORB: An efficient alternative to SIFT or SURF**. In: 2011 International conference on computer vision. IEEE, 2011. p. 2564-2571.

SANTOS, Daniel Teixeira dos. **Compressão de Imagens Usando a Função de Peano e a Transformada Wavelet 1D**. 2012.

SHI, Jianbo; MALIK, Jitendra. **Normalized cuts and image segmentation**. Pattern Analysis and Machine Intelligence, IEEE Transactions on, v. 22, n. 8, p. 888-905. 2000.

SOOTTITANTAWAT, Somkid; AUWATANAMONGKOL, Surapong. **Texture classification using an invariant texture representation and a tree matching kernel**. International Journal of Computer Science Issues, v. 8, p. 99-106. 2011.

TUYTELAARS, Tinne; MIKOLAJCZYK, Krystian. **Local invariant feature detectors: a survey**. Foundations and trends® in computer graphics and vision, v. 3, n. 3, p. 177-280. 2008.

ZHANG, Jianguo; TAN, Tieniu. **Brief review of invariant texture analysis methods**. Pattern recognition, v. 35, n. 3, p. 735-747. 2002.

APÊNDICE A – CÓDIGOS E IMAGENS

Os códigos desenvolvidos para implementar a aplicação **Extrator de características** (Seção 5.1) e a aplicação **Classificador** (Seção 5.2), estão disponíveis em: (https://github.com/jozielpn/ASCIT).

As imagens e as sub-imagens utilizadas nos testes realizados (Seções 6.4, 6.5 e 6.6), também estão disponíveis em: (https://github.com/jozielpn/ASCIT).