UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

INSPEÇÃO INDUSTRIAL ATRAVÉS DE VISÃO COMPUTACIONAL

MAURÍCIO EDGAR STIVANELLO

BLUMENAU
2004

2004/2-38

MAURÍCIO EDGAR STIVANELLO

INSPEÇÃO INDUSTRIAL ATRAVÉS DE VISÃO COMPUTACIONAL

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Paulo César Rodacki Gomes - Orientador

BLUMENAU
2004

2004/2-38

INSPEÇÃO INDUSTRIAL ATRAVÉS DE VISÃO COMPUTACIONAL

Por

MAURÍCIO EDGAR STIVANELLO

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Paulo César Rodacki Gomes, Dr. – Orientador, FURB
Membro:	Prof. Marcel Hugo, FURB
Membro:	Prof. Antônio Carlos Tavares, FURB

Blumenau, 17 de novembro de 2004

Dedico este trabalho a minha família, pelo apoio recebido durante os anos da graduação.

Não basta conquistar a sabedoria, é preciso usá-la.

Cícero

AGRADECIMENTOS

À minha noiva Leila, pelo carinho e incentivo.

Aos meus amigos, pela compreensão quanto às horas ausente.

Ao professor Paulo César Rodacki Gomes, pela orientação e apoio no desenvolvimento deste trabalho, assim como em outros projetos.

Ao professor Cláudio Loesch, pelas dúvidas esclarecidas.

Aos colegas e professores do curso de Ciências da computação pela troca de conhecimentos e experiências.

RESUMO

Este trabalho apresenta um protótipo de inspeção automatizada, utilizando visão computacional. A visão computacional tem assumido um importante papel na indústria, auxiliando o homem em tarefas repetitivas e insalubres, como a inspeção de produtos. Os sistemas de visão computacional fundamentam-se em diferentes áreas da ciência da computação. No desenvolvimento do protótipo são utilizados, dentre outras técnicas de processamento de imagens, os descritores de Fourier, importante abordagem para a descrição de formas a partir da borda. Para a interpretação das representações dos produtos, foi implementada uma rede neural artificial do tipo Perceptron Multicamadas, sendo esta muito empregada no reconhecimento de padrões. A combinação destas técnicas possibilitou o desenvolvimento de um protótipo de inspeção automatizada independente de produto.

Palavras chaves: Segmentação; Descritores de Fourier; Redes Neurais Artificiais; Perceptron Multicamadas; Visão Computacional; Inspeção Automatizada.

ABSTRACT

This work presents a software prototype of automatized inspection using machine vision. The machine vision has assumed an important role in the industry, assisting the man in repetitive and unhealthy tasks, as the inspection of products. The systems for machine vision are based on different areas of the computer science. The software prototype uses Fourier descriptors in order to describe the edge of products shapes, amonget other image processing techniques such as segmentation. For the interpretation of the representations of the products, a neural network of the type Multilayer Perceptron was implement. The combination of these techniques made possible the development of a software prototype of automatized inspection independent of product.

Key-Words: Segmentation; Fourier Descriptors; Artificial Neural Networks; Multilayer Perceptron; Machine Vision.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O mercado consumidor de produtos industrializados está cada vez mais exigente no que diz respeito à qualidade. Pequenos defeitos e imperfeições antes ignorados agora devem ser observados pelos fabricantes, sob o risco de não satisfazerem os consumidores de seus produtos. Este fato tem motivado as indústrias a investirem cada vez mais nos processos de inspeção de qualidade, já que a mesma se tornou um diferencial competitivo.

Apesar deste aumento dos investimentos na área, na maioria das indústrias brasileiras, principalmente nas de pequeno e médio porte, a inspeção e a seleção dos produtos nas fábricas ainda é realizada por inspetores humanos. Na figura 1 é exemplificada uma situação como esta, para o caso específico de inspeção de azulejos numa indústria de cerâmica. Seguindo o modelo convencional de inspeção, o funcionário encontra-se inserido em determinado ponto de uma linha de produção de azulejos para avaliar cada uma das peças produzidas. O funcionário deve avaliar a integridade, a impressão e as dimensões da peça. Caso alguma destas características encontre-se fora dos padrões de qualidade aceitáveis, a peça é retirada da linha de produção para ser descartada ou adicionada a um lote de qualidade inferior.

Figura 1 – Exemplo de inspeção manual

Assim como ocorre na indústria de cerâmica, indústrias de outros segmentos também se da utilizam inspeção manual. No caso da indústria madeireira, os inspetores avaliam a qualidade das peças de madeira com base na quantidade e tamanho dos nós presentes nas mesmas. Já na indústria gráfica, a inspeção manual visa identificar defeitos de impressão presentes nos impressos, como falha no registro de impressão, presença de manchas

indesejadas e problemas relacionados à integridade do impresso. Também na indústria de bebidas são inspecionadas as garrafas, visando identificar defeitos como tampas e rótulos mal posicionados ou não presentes, volume do líquido incorreto e falhas na marcação de códigos. Na figura 2 podem ser observados dois dos exemplos de defeito citados, a serem identificados pelo inspetor no momento da inspeção manual. Dentre estes encontram se a falta de integridade da tampa em uma garrafa, assim como a falha no registro de impressão de um impresso.

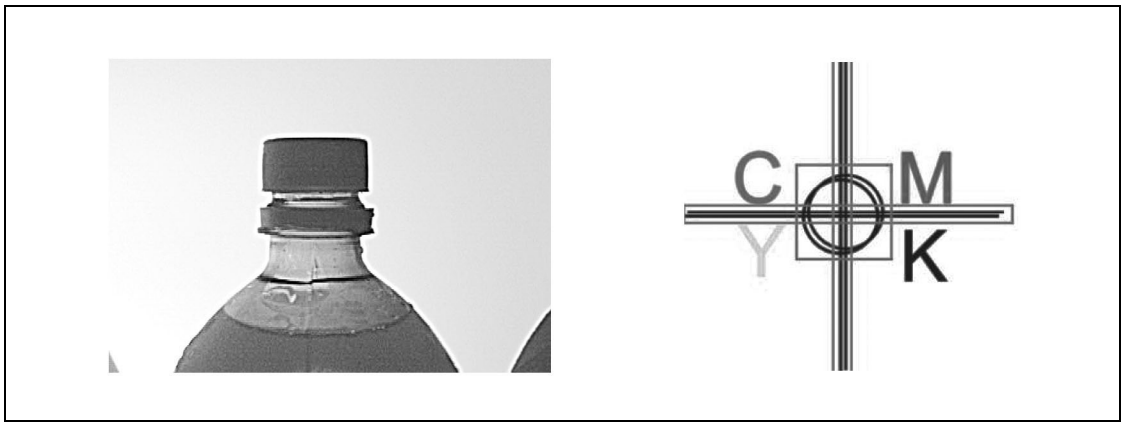

Figura 2 – Exemplo de defeitos em produtos

Por se tratar de uma tarefa extremamente repetitiva e que exige um excessivo esforço físico pelo funcionário que a realiza, a inspeção manual acarreta problemas tais como a falta de inspeção em todos os produtos, a falta de precisão nas inspeções e a alta rotatividade desses trabalhadores. Além disso, a efetividade da inspeção humana declina significativamente no decorrer da jornada de trabalho.

Os sistemas de inspeção por computador surgem para auxiliar o homem nestas tarefas. Disponíveis em diversas configurações e aplicáveis em diferentes situações industriais, estes sistemas tem como principal objetivo o auxilio ou até mesmo a substituição da visão humana no ambiente industrial. As vantagens de sistemas como estes, quando comparados à inspeção convencional, evidenciam-se ainda mais quando a linha de produção deve ser monitorada em alta velocidade e com precisão. A figura 3 exibe um sistema de inspeção automatizada utilizado na indústria de bebidas.

Figura 3 – Exemplo sistema de inspeção automatizada

Apesar de proporcionarem benefícios como aumento de produtividade, melhoria da qualidade e redução de desperdício, a tecnologia de inspeção automatizada ainda não conquistou o espaço merecido. O alto custo e a baixa oferta destes sistemas são alguns dos responsáveis por este quadro.

Visando difundir a tecnologia e contribuir para a valorização da mão de obra humana, que na opinião do autor deveria ser aplicada em tarefas mais intelectuais, o presente trabalho apresenta a implementação de um protótipo de software para inspeção industrial automatizada. Este protótipo processa imagens capturadas por uma câmera de vídeo, inspecionando os produtos contidos nas mesmas através da análise de determinadas propriedades, baseando-se em parâmetros previamente definidos. Para realizar tal tarefa foram utilizadas diversas técnicas de processamento de imagens. Também foi utilizada uma rede neural do tipo *perceptron* multicamada para o reconhecimento e interpretação.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é disponibilizar um sistema que processe e analise imagens de produtos de uma linha de produção.

Os objetivos específicos do trabalho são:

a) analisar características como forma, posição e orientação de embalagens plásticas (frascos) utilizadas na indústria;
b) classificar as embalagens dentre as classes aprovada e rejeitada;
c) detectar defeitos nas embalagens, como a falta ou posição incorreta dos componentes que a constituem.

1.2 ESTRUTURA DO TRABALHO

Primeiramente será realizada uma revisão bibliográfica dos diversos temas que fundamentam o desenvolvimento do protótipo de software de inspeção automatizada. Após isso é demonstrada a especificação e são abordados detalhes da implementação do protótipo. Finalmente são apresentadas as conclusões do presente trabalho, assim como sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordados os assuntos e técnicas utilizadas no desenvolvimento do protótipo e necessários à conclusão deste trabalho.

2.1 SISTEMAS DE INSPEÇÃO AUTOMATIZADA

Os sistemas de visão computacional resultam da integração de diversas tecnologias, e podem ser aplicados em diferentes situações. Dentre as aplicações mais utilizadas pode-se citar, além da indústria, a área de segurança, onde atuam como soluções de identificação biométrica no reconhecimento humano e também no reconhecimento veicular no trânsito.

A figura 4 ilustra a arquitetura simplificada de um sistema de inspeção automatizada para a indústria. Neste exemplo, o processo de inspeção acontece de forma integrada com a linha de produção, evitando assim trabalho desnecessário, já que uma peça identificada como defeituosa não estará presente nas etapas seguintes da produção. Neste processo, para cada peça que transita pela esteira, é capturado pelo sistema de aquisição um quadro contendo a imagem da mesma. Este quadro é transmitido ao sistema de processamento, onde um computador executando um software especificamente desenvolvido para a função de inspeção analisa a imagem a fim de identificar algum defeito na peça. Caso seja identificado algum defeito, o sistema de processamento comanda através de um sinal digital o sistema de descarte, que por meio de um mecanismo pneumático retira a peça defeituosa da esteira.

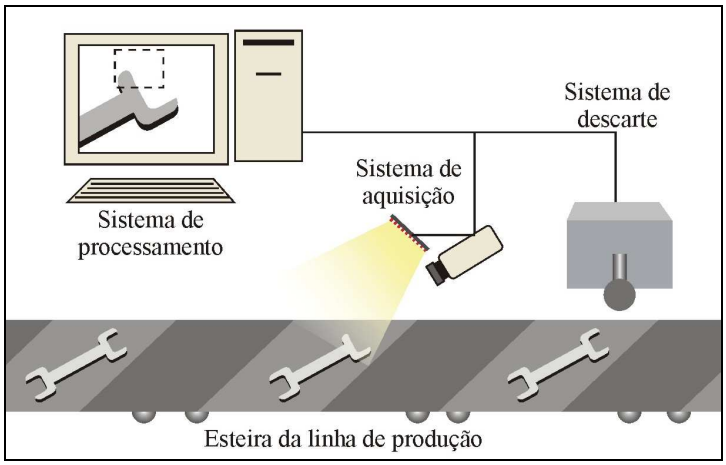

Figura 4 – Exemplo de arquitetura de sistema de inspeção automatizada

Como pode ser visto, a arquitetura de um sistema de inspeção automatizada é composta por vários subsistemas e pode assumir diferentes configurações. Para aplicações de

inspeção industrial, porém, é comum que seja composto por um sistema de aquisição de imagens, um sistema de processamento e um sistema de descarte.

A função do sistema de aquisição de imagens é obter informações da cena a ser monitorada, necessárias ao processamento e consecutiva inspeção. São utilizados diferentes equipamentos como sensores, câmeras, lentes, filtros, dispositivos de iluminação, entre outros. Exemplos destes componentes podem ser vistos na figura 5.

Figura 5 – Exemplo de câmera, lente e componente de iluminação

As câmeras são os componentes básicos do sistema de aquisição. Possuem uma matriz de sensores sensíveis à luz onde, para cada elemento da matriz, é obtida a intensidade de luz que incide naquele ponto. No ambiente a ser observado é aplicada uma iluminação sobre o objeto de interesse. A luz é refletida no objeto e incide sobre a câmera. A intensidade de cada um dos pontos forma uma imagem digital, que é utilizada posteriormente para análise.

Conforme explicado anteriormente, a iluminação exerce um papel fundamental no sistema de captura, uma vez que a câmera captura a luz refletida sobre a cena observada. Existem diversas técnicas de iluminação, que variam entre si na fonte de luz, intensidade, direção, etc. Cada uma delas oferece algum tipo de benefício e ressalta determinadas características dos objetos. Por isso, a escolha do tipo de iluminação é muito importante. Por exemplo, para o reconhecimento de caracteres na embalagem de um produto para verificação da qualidade de impressão da sua data de validade, seria ideal a utilização de um anel luminoso em torno da câmera, que produziria uma luz paralela ao seu eixo óptico. Já para a inspeção de garrafas onde se deseja inspecionar a presença da tampa, seria interessante uma iluminação de fundo, que, por ser posicionada atrás do objeto, destaca a silhueta do mesmo.

Objetivando destacar características específicas dos produtos e da cena observada, também são empregados diversos acessórios à câmera ou ao ambiente. Na câmera, pode-se acoplar, por exemplo, um filtro vermelho para se conseguir um maior contraste na imagem. Também se pode conseguir a visualização de uma face oculta de um produto através da utilização de espelhos.

Constituído de hardware e software, o sistema de processamento é responsável pelo controle de todo o sistema de visão. Diversas arquiteturas de hardware podem ser utilizadas para este fim, desde computadores IBM-PC comuns até arquiteturas especificamente desenvolvidas para sistemas de visão.

Além do hardware básico necessário ao processamento, este sistema deve possuir placas de aquisição de imagens capazes de converter o sinal de vídeo proveniente das câmeras em imagens digitais. Além disso, deve possuir portas de comunicação utilizadas no acionamento de sistemas externos ou no recebimento de sinais provenientes de sensores. A figura 6 exibe uma placa de aquisição de imagens, contendo também portas digitais de comunicação.

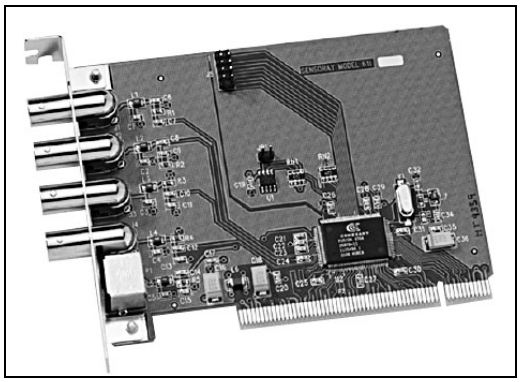

Figura 6 – Exemplo de placa de aquisição

O software utilizado nos sistemas de visão é responsável pelo controle do sistema como um todo, pelo processamento e análise das imagens e pela interface com o usuário. Através destes softwares específicos, é realizada a análise sobre cada uma das imagens em busca de defeitos, irregularidades ou mesmo para realizar classificação. Com base no resultado da análise, o software toma uma decisão à cerca de cada produto analisado, que pode resultar em diferentes ações. Além da inspeção o software pode ainda realizar levantamentos estatísticos importantes sobre a produção. A figura 7 exibe a tela um software

de inspeção desenvolvido para a indústria de bebidas, no momento de configuração da análise de tampa em garrafas.

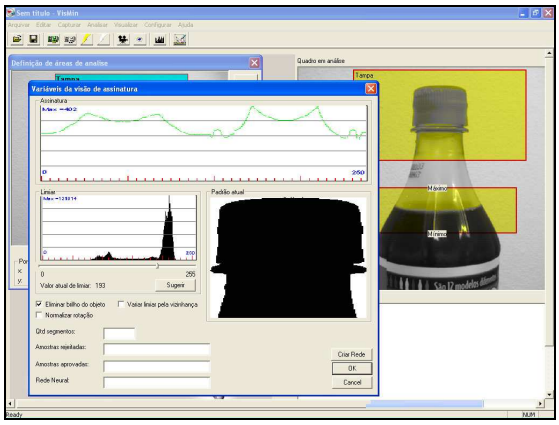

Figura 7 – Exemplo software de inspeção

O sistema de descarte é responsável por retirar a peça rejeitada da linha de produção. Comandados pelo software de inspeção, estes sistemas podem assumir diferentes configurações. Muitos destes sistemas utilizam-se de guias para realizar o descarte da peça, como é o caso do sistema exibido na figura 8.

Figura 8 – Exemplo sistema de descarte

Neste exemplo, o software de inspeção comanda o sistema de descarte, que redireciona a garrafa rejeitada para outro destino, dentro da linha de produção.

2.2 PROCESSAMENTO DE IMAGENS

O processamento de imagens assume um importante papel nos sistemas de visão computacional. A extração de informações de uma imagem, para posterior análise e tomada de decisão a cerca de um produto é conseguida através do emprego de diversas técnicas de

processamento de imagens. Nesta seção serão descritas algumas das técnicas utilizadas nestes sistemas.

2.2.1 REPRESENTAÇÃO DE IMAGENS DIGITAIS MONOCROMÁTICAS

Segundo Gonzales e Woods (2000, p. 4) “o termo imagem refere-se à função bidimensional de intensidade da luz f(x,y), onde x e y denotam as coordenadas espaciais e o valor de f em qualquer ponto(x,y) é proporcional ao brilho da imagem naquele ponto”.

Existem várias maneiras de representar digitalmente imagens. A mais utilizada é a que representa a imagem na forma de uma matriz, em que em cada posição da mesma representa um ponto da imagem e onde é armazenado o nível de cinza daquele ponto. Para o menor elemento de uma imagem digital, ou seja, para cada posição da matriz é dado o nome de *pixel*. Cada *pixel* é representado por um byte, podendo assim assumir 256 valores diferentes.

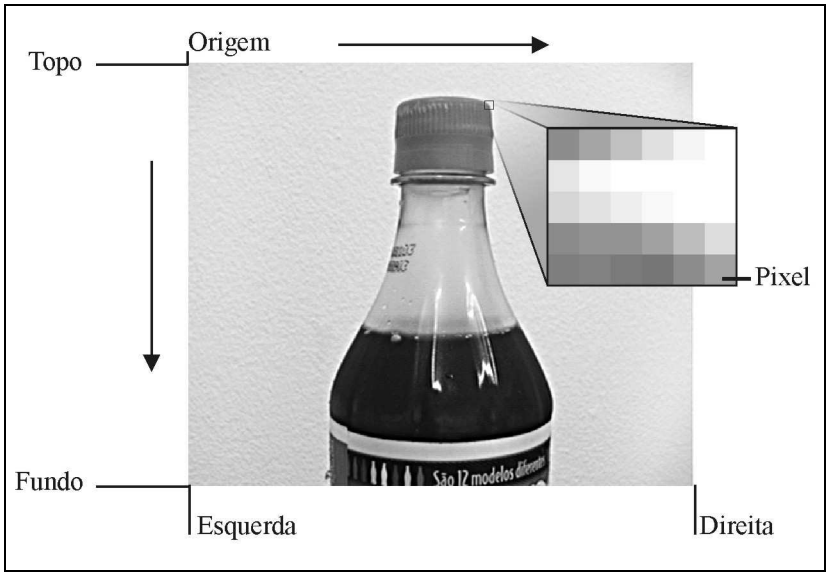

Figura 9 – Representação de imagem digital

A Figura 9 ilustra a representação descrita e que é utilizada neste trabalho. Também é exibida a orientação de sistema de coordenadas mais utilizada nos algoritmos de processamento de imagem, com a origem do sistema no canto superior esquerdo da matriz.

2.2.2 RELACIONAMENTO BÁSICO ENTRE *PIXELS*

Serão descritos nesta seção alguns relacionamentos básicos entre *pixels* de uma imagem. Estes relacionamentos são utilizados em diversas técnicas de processamento de imagens.

2.2.2.1 VIZINHANÇA DE UM PIXEL

Gonzales e Woods (2000, p. 26) explicam a relação de vizinhança entre *pixels* de uma imagem. Um *pixel* p, posicionado nas coordenadas (x,y), possui quadro vizinhos, cujas coordenadas são dadas são exibidas no quadro 1.

$$\boxed{(x+1, y), (x-1, y), (x, y+1), (x, y-1)}$$

Quadro 1 – Coordenadas dos vizinhos-de-4 de *p*

A esse tipo de vizinhança dá-se o nome de vizinhança-de-4, sendo que se refere aos *pixels* horizontais e verticais que estão a uma unidade de distância de *p*.

Já a vizinhança chamada vizinhança-de-8 considera também os vizinhos diagonais de *p*. Adicionalmente aos *pixels* posicionados nas coordenadas exibidas no quadro 1, são considerados os *pixels* posicionados nas coordenadas exibidas no quadro 2.

$$\boxed{(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)}$$

Quadro 2 – Coordenadas dos vizinhos-de-8 de *p*

Observe que alguns dos *pixels* vizinhos não estarão presentes na imagem quando *p* estiver localizado na borda da imagem.

2.2.2.2 CONECTIVIDADE

Gonzales e Woods (2000, p. 27) explicam que a conectividade entre *pixels* é um conceito importante no processamento de imagens, em especial para o estabelecimento das bordas de objetos presentes em uma imagem.

Para estabelecer se dois *pixels* são conexos, é preciso determinar se são vizinhos e se seus níveis de cinza são similares. Observe que, conforme a necessidade, pode ser considerada para determinação de conexidade a vizinhança-de-4 ou a vizinhança-de-8, resultando respectivamente na conectividade-de-4 e conectividade-de-8.

2.2.2.3 ROTULAÇÃO DE COMPONENTES CONEXOS

Gonzales e Woods (2000, p. 28) descrevem um algoritmo para a rotulação de componentes conexos em uma imagem binária. Considere que uma imagem seja percorrida *pixel* por *pixel*, seguindo a orientação descrita na seção 2.1.1. Considerando a vizinhança-de-4, defina-se *p* como sendo o *pixel* em qualquer passo no processo de varredura e sejam *r* e *t*,

respectivamente, os vizinhos superior e esquerdo de *p*. Observe que, obedecendo à seqüência de varredura definida, quando chegar ao ponto *p*, já terão sido encontrados *r* e *t*.

Como passo inicial do procedimento de rotulação, verifique se o valor de *p* é 0. Em caso positivo mova para a próxima posição. Se o valor de *p* é 1, verifique os valores de *r* e *t*. Se ambos forem 0, atribua a *p* um novo rótulo. Note que, neste caso, ou é a primeira vez que o componente conexo aparece, ou o *pixel* se trata de uma ponta de um componente já encontrado. Caso apenas um dos vizinhos for 1, atribua a *p* o seu rótulo. Se ambos forem 1 e possuem o mesmo rótulo, atribua a *p* o mesmo rótulo. Caso forem 1, mas possuem rótulos diferentes, atribua um dos rótulos a *p* e anote que os dois rótulos são equivalentes. Observe que, neste caso, *r* e *t* fazem parte do mesmo componente conexo, já que estão ligados por *p*. Procedendo desta maneira, ao fim da varredura todos os pontos com valor 1 terão sido rotulados. Para finalizar, resta apenas unir as classes marcadas como equivalentes.

O mesmo processo pode ser realizado para a rotulação de componentes conexos considerando a vizinhança-de-8, mas neste caso devem ser considerados os dois vizinhos diagonais superiores de p.

2.2.3 PRÉ-PROCESSAMENTO

De maneira geral, o pré-processamento visa melhorar a qualidade de uma imagem digital com a finalidade de facilitar o processamento subseqüente de análise. Dentro de um sistema de visão computacional o pré-processamento exerce principalmente as funções de realce e restauração das imagens a serem processadas.

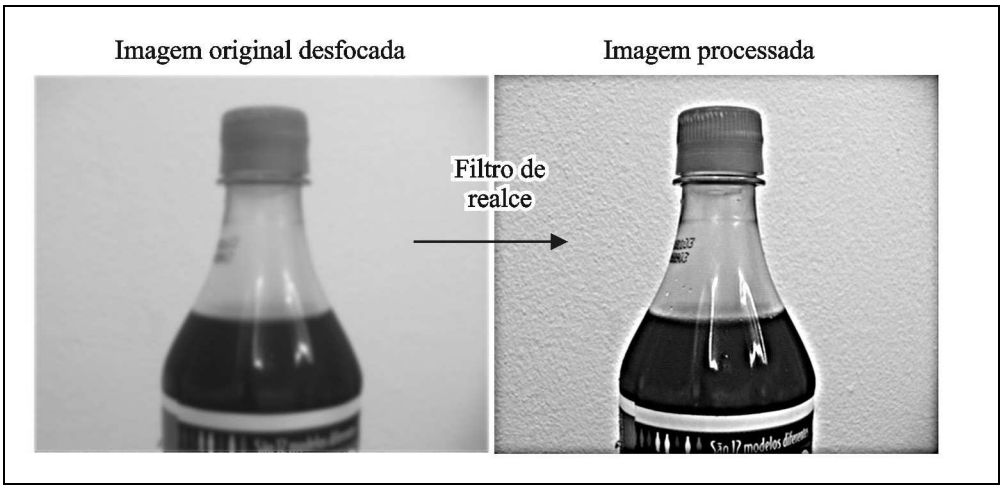

Figura 10 – Aplicação do filtro de realce

Na operação de realce, o objetivo é destacar as características importantes da imagem. Para isso são utilizadas técnicas de contraste, destaque de contornos e suavização. A figura 10 exemplifica a aplicação de realce sobre uma imagem desfocada. Gonzales e Woods (2000, p. 115) ressaltam o fato de que as técnicas de pré-processamento são bastante dependentes da aplicação.

Já a operação de restauração de imagens, visa deixar uma imagem digital o mais próximo possível da cena captada. No processo de digitalização de imagens através de dispositivos eletrônicos é comum a perda de detalhes ou a obtenção de ruídos indesejados. Como exemplo de casos onde podem ser aplicadas técnicas de restauração de imagem pode-se citar a ausência de foco e o movimento dos objetos na imagem capturada.

2.2.4 SEGMENTAÇÃO DE IMAGENS

A segmentação em imagens digitais tenta reproduzir o processo da visão humana descrita por Facon (1993, p. 87) onde são efetuados agrupamentos sobre a imagem percebida, baseados em proximidade, similaridade e continuidade. De maneira geral, esta etapa do processamento de imagens tem como objetivo isolar os objetos de interesse.

Nesta etapa o objetivo é dividir a imagem em elementos significativos. Esta separação pode ocorrer em vários níveis, onde, em um dado momento, pode-se aplicar uma segmentação para obter determinado objeto e em outro momento pode-se obter um detalhe deste objeto.

Gonzales e Woods (2000, p. 4) explicam que a maioria dos algoritmos de segmentação baseia-se na descontinuidade ou similaridade dos valores de nível de cinza. Exemplos de cada uma das abordagens serão descrito nas próximas seções.

2.2.4.1 LIMIARIZAÇÃO

Uma abordagem muito utilizada para a segmentação de imagens através da similaridade é a limiarização. Esta abordagem engloba várias técnicas diferentes. Uma delas é a limiarização global. Nesta técnica, define-se um limiar que representa um nível de cinza qualquer. Para cada *pixel* da imagem é realizado um teste para verificar se o nível de cinza deste *pixel* é maior que o limiar definido. Em caso positivo, o mesmo é rotulado com um valor, caso contrário é rotulado com outro valor. O resultado deste processo é uma imagem

binária, ou seja, a cor de cada *pixel* é representada por apenas um bit. Cada *pixel* da imagem, portanto, apresenta a cor preta (valor 0) ou branca (valor 1).

Para aplicações de inspeção, o resultado da segmentação desejado é uma imagem binária onde a forma do objeto a ser inspecionado esteja representada por um agrupamento de *pixels* com determinado valor e que o fundo esteja representado com *pixels* de outro valor.

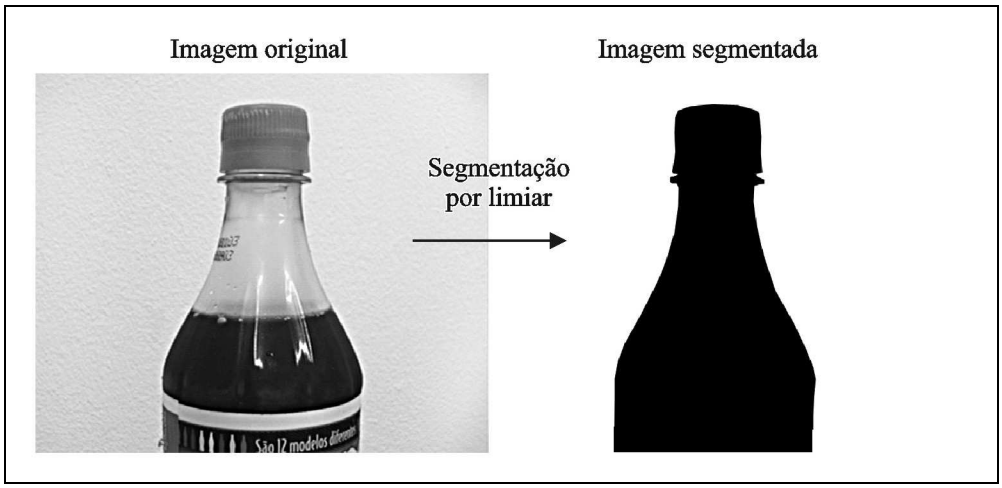

Figura 11 – Segmentação de imagem através de limiar

A figura 11 exemplifica a segmentação através de limiarização global, onde o resultado obtido é a silhueta do frasco a ser inspecionado.

2.2.4.2 DETECÇÃO DE BORDAS

A detecção de bordas é uma das técnicas em que a segmentação é realizada com base na descontinuidade de valores de níveis de cinza. Gonzales e Woods (2000, p. 4) explicam que a detecção de bordas pode ser obtida pelo uso de filtros por derivada.

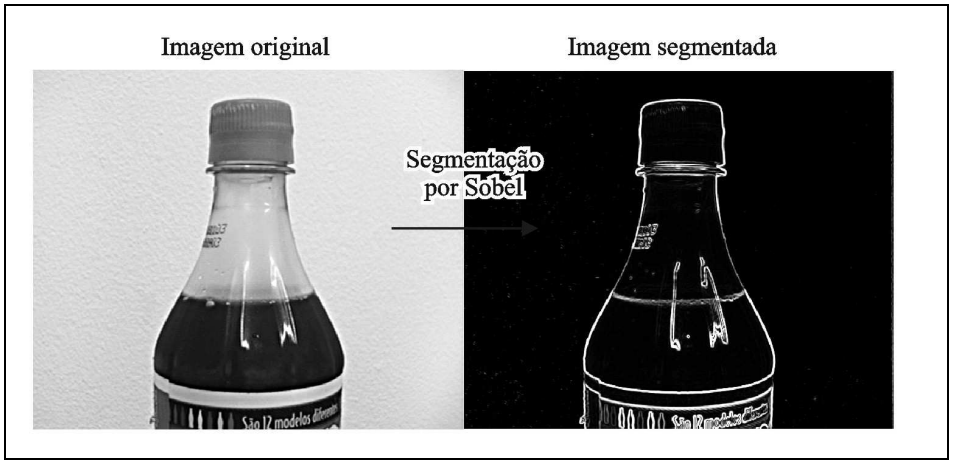

Figura 12 – Aplicação do operador Sobel sobre imagem

Dentre os filtros por derivada encontra se o denominado operador de Sobel. Na figura 12 pode-se verificar o resultado obtido pela aplicação deste filtro em uma imagem. Este filtro faz parte dos métodos de domínio espacial que operam diretamente sobre os *pixels* de uma imagem considerando a vizinhança dos mesmos. A abordagem principal para definir a vizinhança em torno de um *pixel* consiste em usar uma subimagem quadrada, centrada no *pixel* em questão. O centro desta subimagem é movido então de *pixel* a *pixel* sobre toda a imagem, aplicando-se o operador para cada posição da mesma.

a)

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

b)

-1	-2	-1
0	0	0
1	2	1

c)

-1	0	1
-2	0	2
-1	0	1

Figura 13 – a) Posições de uma imagem 3 X 3; b) Máscara para x; c) Máscara para y

A partir das máscaras do operador de Sobel exibidas na figura 13, tem-se que as derivadas baseadas nas mesmas podem ser obtidas pelas equações mostradas no quadro 3.

$$Gx = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$
$$Gy = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

Quadro 3 – Equação da derivada pelas máscaras de Sobel

2.2.5 DESCRITORES DE IMAGENS

Na etapa de segmentação, obtêm-se agrupamentos de *pixels* que representam os componentes da imagem a serem analisados. Porém, estes componentes devem ser descritos de maneira mais apropriada, já que uma análise direta sobre os *pixels* não é muito eficiente.

Os descritores são conjuntos de números, gerados para descrever uma forma. Os descritores podem não reconstruir completamente a forma descrita, mas devem ser suficientes para discriminar diferentes formas.

A descrição de componentes da imagem pode se dar através de descritores simples, como a área. A área é obtida simplesmente através da contagem dos *pixels* que compõe a região. Porém, para a maioria das aplicações, informações simples como a área não são

suficientes para a resolução dos problemas. A seguir serão descritos dois métodos de descrição de imagens que as descrevem com quantidade maior de detalhes.

2.2.5.1 CÓDIGO DE CADEIA

Um exemplo de descritor mais completo é o código de cadeia.

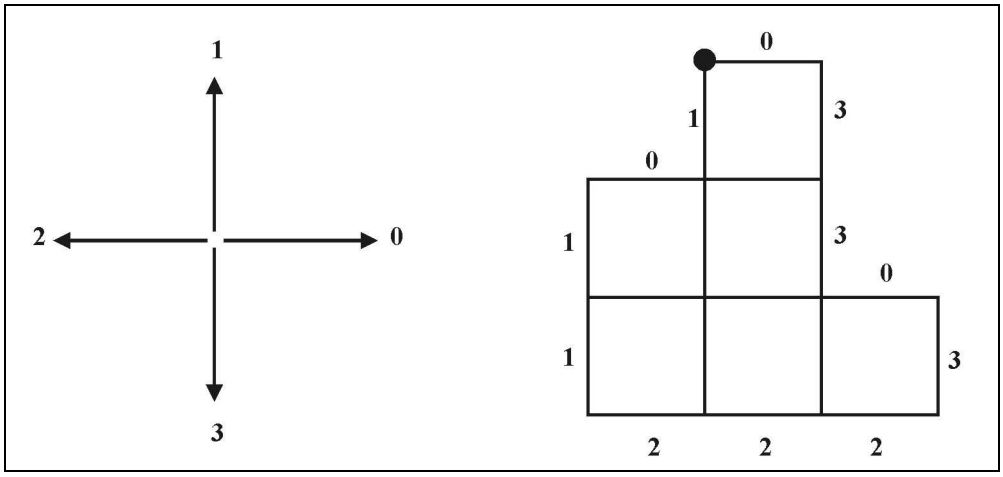

Figura 14 – Código de cadeia

César e Costa (2001, p. 350) descrevem que o código de cadeia pode ser gerado seguindo-se a fronteira de um agrupamento de *pixels* em determinado sentido e atribuindo-se uma direção aos segmentos que conectam cada par de *pixels*. A figura 14 exemplifica a geração de código de cadeia com quatro direções possíveis e onde a cadeia resultante para a forma descrita foi 033032221101.

Esta técnica, apesar de bastante conhecida, não será utilizada neste trabalho devido ao tamanho elevado da cadeia resultante do processo de descrição.

2.2.5.2 DESCRITORES DE FOURIER

César e Costa (2001, p. 350) afirmam que os descritores de Fourier são uma das formas de representação de imagens mais populares para aplicações de visão computacional e reconhecimento de padrões. Estes descritores não constituem um método simples, mas uma classe de métodos já que existem diferentes maneiras de defini-los.

Gonzales e Woods (2000, p. 355) demonstram como uma imagem pode ser representada através de descritores de Fourier. A figura 15 exibe uma fronteira digital de N pontos no plano xy.

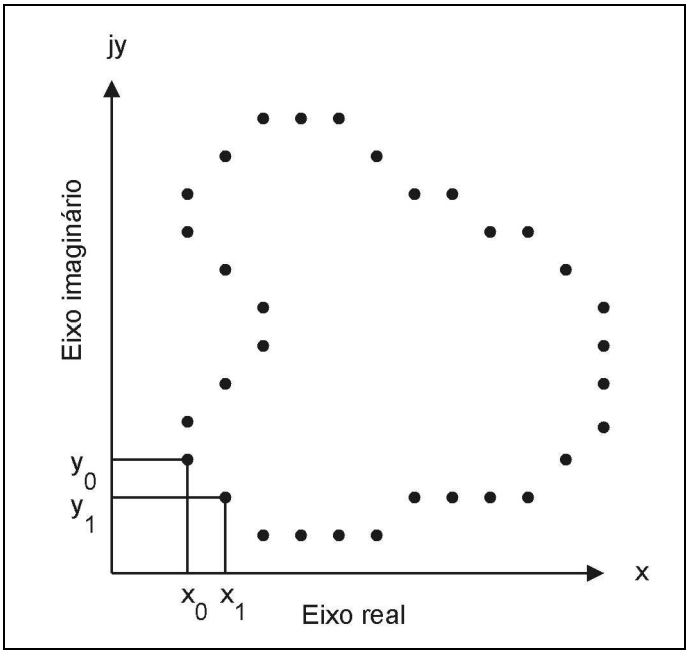

Figura 15 – Código de cadeia

Iniciando de um ponto qualquer (x0,y0) no sentido anti-horário (por exemplo), pode-se encontrar os pares de coordenadas (x0,y0),(x1,y1),(x2,y2),...,(xn-1,yn-1) que representam a fronteira da forma. Cada par pode ser tratado como número complexo da forma exibida no quadro 2, para k = 0,1,2,...,N – 1, onde N é a quantidade de pontos que compõe a fronteira. Esta representação possui a vantagem de reduzir um problema de duas dimensões a uma só dimensão, e pode ser visualizada no quadro 4.

$$s(k) = x(k) + jy(k)$$

Quadro 4 – Formato de número complexo

A transformada discreta de Fourier é definida no quadro 5, para u = 0,1,2,...,N-1, onde N é a quantidade de pontos que compõe a fronteira. Os coeficientes complexos a(u) são chamados de descritores de Fourier.

$$a(u) = \frac{1}{N}\sum_{k=0}^{N-1} s(k)^{-j2\pi uk/N}$$

Quadro 5 – Transformada discreta de Fourier

Observe que podem ser obtidos tantos descritores quanto o número de postos que compõe a fronteira no plano *xy*. Porém, não existiria vantagem em se descrever a forma pela mesma quantidade de pontos. A vantagem deste método de descrição encontra-se justamente

na possibilidade de representar uma forma com uma pequena quantidade de descritores. Gonzales e Woods (2000, p. 357) afirmam que poucos coeficientes de baixa ordem são capazes de capturar a forma geral, e que um grande número de coeficientes é necessário na definição das características mais marcantes, como cantos e retas. A propriedade de que poucos coeficientes capturam a essência geral de uma fronteira possui um grande valor, já que desta maneira podem ser usados para diferenciar formatos de fronteiras distintos.

Oliveira e Bauchspiess (2001) utilizaram os descritores de Fourier gerados pela ferramenta Matlab (MATHWORKS, 2004) para a representação da borda de objetos, previamente descritos por cadeias direcionais. O método utilizado para a obtenção dos descritores difere do método apresentado neste trabalho. As equações utilizadas pelos autores não se encontram na forma exponencial, como são apresentadas nos quadros 5 e 6. Em seu trabalho afirmam que o uso de 10 a 25 descritores mostrou-se suficiente para a representação das formas utilizadas.

A transformada inversa de Fourier de *a(u)* é definida no quadro 6, para $u = 0,1,2,...,N-1$ onde *N* é a quantidade de pontos que compõe a fronteira. A transformada inversa de Fourier é utilizada para reconstruir s(*k)* a partir dos coeficientes *a(u)*.

$$s(k) = \sum_{u=0}^{M-1} a(u)^{j 2\pi uk / N}$$

Quadro 6 – Transformada inversa de Fourier

A variável *M* representa o número de coeficientes utilizados na descrição da fronteira, sendo que este pode ser menor ou igual a *N*-1. Na figura 16 são exibidas as fronteiras obtidas pela reconstrução da fronteira original de um quadrado, geradas a partir de diferentes números de coeficientes. Através deste exemplo pode-se confirmar a premissa de que os coeficientes de baixa ordem capturam a essência geral da forma, sendo os detalhes mais finos capturados pelos coeficientes de alta freqüência.

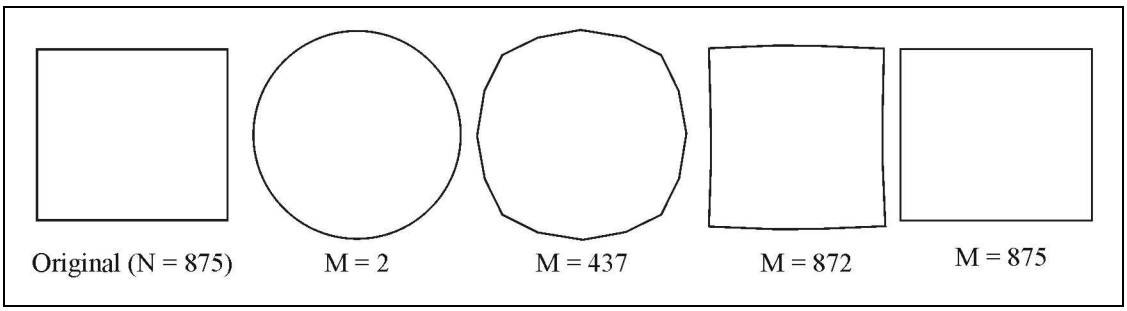

Figura 16 – Forma original e reconstruções a partir de *M* coeficientes

Outra vantagem na utilização dos descritores de Fourier é que sobre os mesmos podem ser aplicadas transformações simples com o objetivo de obter invariância quanto à rotação, translação e escala. Bowman, Soga e Drummond (2001) explicam que a invariância quanto à translação pode ser obtida simplesmente não utilizando o descritor onde $N = 0$, e apresentam a equação para a obtenção de descritores invariantes à rotação, sendo esta exibida no quadro 7.

$$a_r(u) = \sqrt{a_u^2 + b_u^2}$$

Quadro 7 – Transformada para obtenção de invariância

Através do uso da transformação exibida no quadro 7, os descritores obtidos a partir de uma mesma borda em diferentes ângulos, são iguais.

2.2.6 INTERPRETAÇÃO DE IMAGENS

A interpretação consiste em identificar padrões através da análise das descrições da imagem realizadas nas etapas anteriores. Esta análise leva em conta padrões ou regras previamente definidas.

Facon (1993, p.174) exemplifica uma interpretação onde deverão ser classificados objetos dentre cinco classes existentes. Nesta análise os objetos são descritos através das seguintes propriedades: perímetro, área, número de Euler, raio mínimo e raio máximo. Neste exemplo a classificação se dá diretamente pela comparação entre todas as propriedades do objeto analisado com cada um dos padrões existentes. Existem casos, porém, em que os objetos analisados não podem ser descritos por propriedades simples como estas.

Em oposição ao método utilizado, Gonzales e Woods (2000, p. 424) ressaltam que as propriedades estatísticas das classes de padrões são freqüentemente desconhecidas, ou não podem ser estimadas. Problemas de decisão teórica são mais bem tratados por métodos que

levem às funções de decisão através de treinamento. Na seção seguinte será discutido o método de redes neurais, que segue este princípio.

2.3 REDES NEURAIS

Dentre as várias técnicas utilizadas no reconhecimento de padrões encontram-se as redes neurais artificiais. Surgidas como uma tentativa de simular o funcionamento do cérebro e resolver problemas do mesmo modo como o homem o faz, as redes neurais artificiais são utilizadas em várias áreas na resolução de problemas que vão desde a extração de informações de grandes bases de dados até o reconhecimento de imagens de satélite.

Nas seções seguintes serão abordadas as redes neurais artificiais. Para uma melhor compreensão do assunto serão inicialmente descritos a estrutura e funcionamento básico da rede neural biológica.

2.3.1 O CÉREBRO E A REDE NEURAL BIOLÓGICA

Segundo Medeiros (2003, p.1), o cérebro humano é composto por 100 bilhões de neurônios, e é responsável pelo controle da maioria das funções do corpo. Os neurônios são as unidades fundamentais dos tecidos do sistema nervoso. O núcleo de cada neurônio é conectado a outros neurônios por meio de vários filamentos chamados dentritos e um filamento mais longo denominado axônio. Estas conexões são denominadas conexões sinápticas. A figura 17 exibe uma ilustração do neurônio biológico.

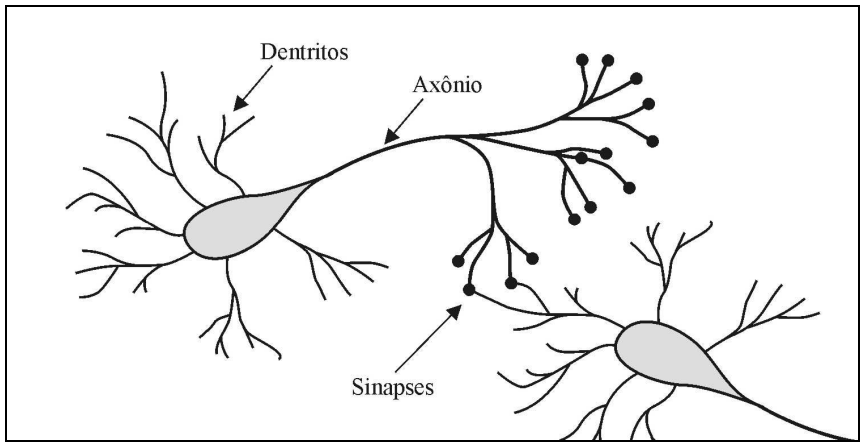

Figura 17 – O Neurônio Biológico

O neurônio pode disparar pulsos elétricos através das conexões sinápticas, sendo estes recebidos pelos dentritos dos outros neurônios. Quando um neurônio recebe pulsos elétricos

pelos dentritos, ele ativa e eventualmente dispara um pulso ao axônio, que emite este pulso aos outros neurônios. Esta ativação e o conseqüente repasse de sinal aos demais neurônios somente é realizada caso os pulsos elétricos recebidos pelas sinapses sejam fortes o suficiente a ultrapassar determinado limiar. As conexões sinápticas e a quantidade de pulsos necessários para ativar o neurônio podem mudar. Estas características permitem a rede neural reter conhecimento, ou seja, aprender. O conhecimento retido fica distribuído por toda a rede neural.

2.3.2 REDE NEURAL ARTIFICIAL

Segundo Loesch e Sari (1996, p. 5), redes neurais artificiais são sistemas computacionais de implementação em hardware ou software que imitam as habilidades computacionais do sistema nervoso, usando um grande número de neurônios interconectados. Conforme comentado anteriormente, sistemas que simulam a rede neural biológica já são utilizados com sucesso nas mais diversas áreas.

Na seqüência serão explanados a estrutura, dinâmica e aprendizado das redes neurais artificiais.

2.3.2.1 ESTRUTURA DA REDE NEURAL

Neste momento serão consideradas inicialmente as características individuais de cada neurônio da rede. Logo após será considerada a organização dos grupos de neurônios que constituem a rede.

2.3.2.1.1 O NEURÔNIO ARTIFICIAL

O neurônio artificial é o elemento básico que forma uma rede neural artificial. Também conhecido como elemento de processamento, simula o funcionamento de um neurônio biológico. A figura 18 apresenta um modelo de neurônio artificial demonstrado por Loesh e Sari (1996, p. 21).

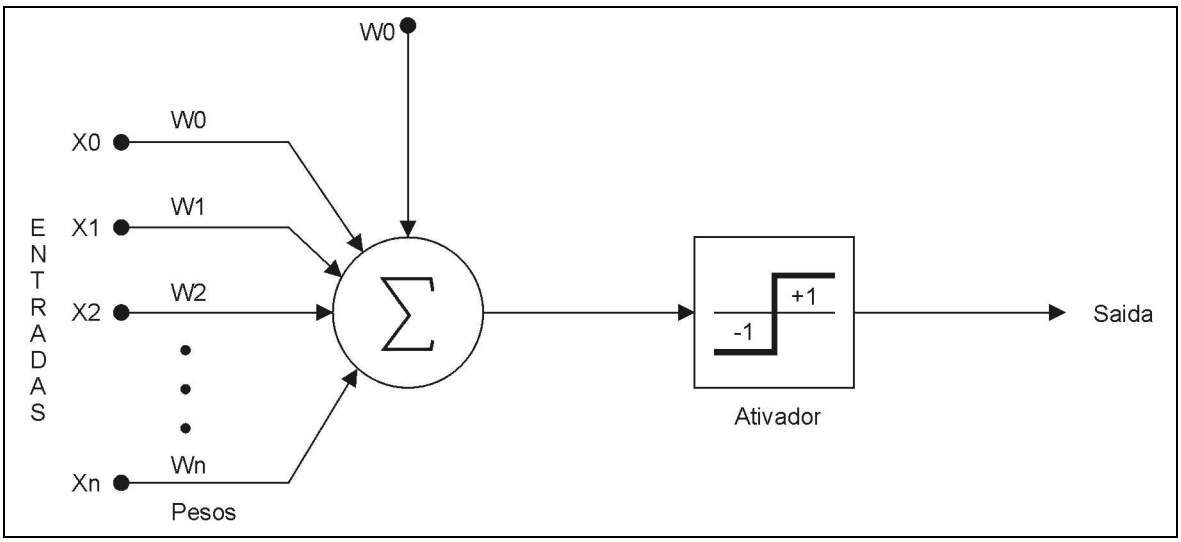

Figura 18 – Modelo de Neurônio Artificial

Baseados no neurônio biológico, os elementos principais que constituem o neurônio artificial são:

a) entradas;
b) pesos sinápticos;
c) função de ativação ou função Soma;
d) função de transferência ou ativador.

O neurônio possui um ou mais sinais de entrada. É através destas entradas que o neurônio recebe os estímulos a serem processados. Assim como ocorre no neurônio natural, no neurônio artificial todas as entradas são consideradas de maneira simultânea no momento do processamento. Não existe situação onde somente o valor de uma ou outra entrada é considerado.

Os pesos são os valores que representam o grau de importância de cada entrada para o neurônio. É através da variação destes valores que se constrói o conhecimento. Os valores dos pesos são obtidos no momento do treinamento da rede neural.

A função de ativação antecede a função de transferência e tem como atribuição repassar o sinal obtido através das entradas à função de transferência. Em modelos mais simples de redes neurais esta função simplesmente realiza a soma dos valores das entradas multiplicados pelos respectivos pesos.

A função de transferência analisa o valor gerado pela função de ativação e gera uma saída para o neurônio. A função muda conforme o modelo de rede utilizado. Uma das funções

mais empregadas é a de sinal, representada graficamente na figura 19, onde o valor obtido pela função de ativação é comparado com determinado limiar. Conforme o resultado da comparação, a saída assume um entre dois valores predeterminados.

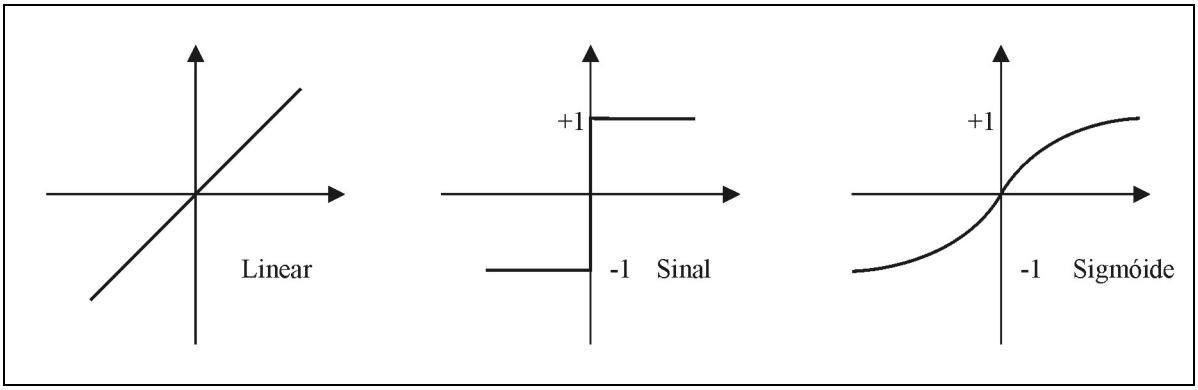

Figura 19 – Gráficos de funções de transferência

Loesh e Sari (1996, p. 23) explicam que nas redes Perceptron Multicamadas tradicionalmente é utilizada a função de transferência sigmoidal. Esta função fornece uma forma para controle automático de ganho. Para argumentos de pequena magnitude o ganho é acentuado. Já para argumentos de alta magnitude, o ganho é menor. Desta maneira, os sinais de grande amplitude não saturam a rede, assim como os sinais baixos não deixam de ser considerados.

2.3.2.1.2 REDE DE NEURÔNIOS

Em uma rede neural artificial os neurônios são agrupados em camadas. A figura 20 exibe a organização das diferentes camadas na rede:

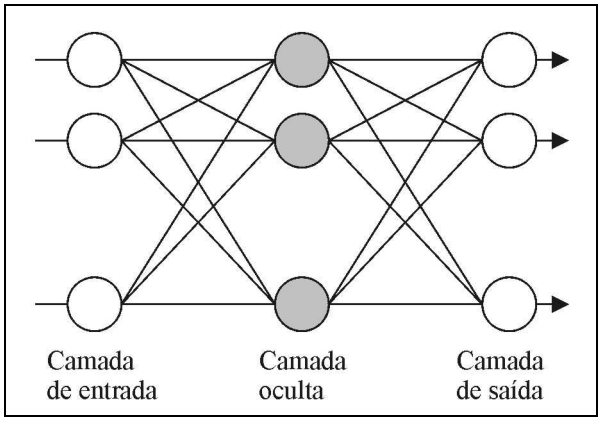

Figura 20 – Camadas da rede neural

Tafner (1996, p. 62) explica que os neurônios da camada de entrada não realizam processamento. Sua única função é armazenar a informação de entrada para ser repassada aos neurônios da próxima camada

Uma rede neural artificial pode também possuir camadas intermediárias ou ocultas, que se situam entre a camada de entrada e a camada de saída. A estrutura destas camadas é igual à da camada de saída, porém não tem contato com o exterior. Segundo Medeiros (2003, p. 11) estas camadas tem como objetivo melhorar o desempenho da rede, aumentando a possibilidade de divisão do espaço de entrada de maneira não linear.

Por fim tem-se a camada de saída. Além de realizar processamento através de seus neurônios esta camada também é responsável por repassar o resultado do processamento da rede ao mundo exterior. A quantidade de neurônios da camada de saída é igual ao número de saídas esperadas da rede.

Quanto às camadas, devem-se considerar as diferentes organizações possíveis:

a) quantidade de camadas;
b) quantidade de neurônio por camadas;
c) o tipo de conexão entre as camadas e seus neurônios.

Loesch e Sari (1996, p. 24) explicam que estas diferentes organizações de camadas distinguem os tipos de arquitetura de redes neurais artificiais existentes:

a) multicamadas, rede *feedforward*;
b) camada simples, redes conectadas lateralmente;
c) bicamadas, redes *feedforward*/*feedback*;
d) multicamadas, redes cooperativas;
e) redes híbridas.

Dentre os tipos de arquitetura citados será considerado o tipo Multicamadas *feedforward*, que será descrito em seção própria.

2.3.2.2 FASES DO PROJETO DE REDE NEURAL

Apesar das redes neurais artificiais serem utilizadas em diversas áreas, seu objetivo é quase sempre o mesmo, reconhecer e classificar padrões, além de generalizar informações. Tafner (1996, p. 62) esclarece que diante de um projeto de uma rede neural ao invés de se

pensar em procedimentos e fórmulas algorítmicas de processamento de dados deve-se ter em mente tipos de dados de entrada, dados de saída e tratamento de dados. Também afirma que a rede tem dois momentos distintos de processamento: o momento de aprendizado e o momento de utilização. Segundo ele, apesar destas duas fases bem distintas de processamento, um projeto de rede possui 3 fases principais, que serão vistas a seguir.

2.3.2.2.1 DEFINIÇÃO DA REDE NEURAL

Na fase de definição da rede neural devem-se identificar as variáveis relacionadas ao problema que contém as informações necessárias à resolução do mesmo. Uma vez identificadas estas variáveis é escolhido o modelo da rede neural a ser utilizado. Também devem ser definidos os seguintes aspectos da rede:

a) tamanho da rede: deve-se definir o tamanho da rede no que diz respeito a quantidade de camadas, quantidade de neurônios por camada, quantidade de entradas e quantidade de saídas;
b) tipo de problema a ser resolvido: o problema pode ser de classificação, padronização ou otimização;
c) tipo de aprendizado: o algoritmo de aprendizado deve ser selecionado entre supervisionado e não-supervisionado.

2.3.2.2.2 TREINAMENTO DA REDE NEURAL

Dentre as formas de treinamento possíveis para a rede neural artificial, o treinamento supervisionado é o mais utilizado. Tafner (1996, p. 65) explica o processo de aprendizado supervisionado de uma rede neural. Neste processo, deve-se possuir um conjunto de treinamento organizado em pares, onde para cada entrada se tenha a saída desejada. Estes pares são então apresentados à rede e para cada entrada deve ser verificado se a saída obtida corresponde à saída desejada. Caso a saída obtida seja diferente da saída desejada deve ocorrer o ajuste dos pesos sinápticos dos neurônios da rede. Caso a saída obtida seja igual à saída desejada, deve-se apresentar o par seguinte à rede. Este processo deve se repetir para todos os pares do conjunto de treinamento, até que se obtenha uma taxa de acerto satisfatória.

O ajuste sináptico citado nada mais é do que o aprendizado do fato apresentado. É através deste ajuste que o conjunto de neurônios representa a informação que foi apresentada à rede. O ajuste sináptico é resultado de um cálculo que visa somar ao peso atual um valor que

corresponda ao grau de erro gerado pela rede diante a uma entrada, a fim de corrigir o valor do peso. Dentre os cálculos existentes o mais encontrado é a chamada Regra Delta, que pode ser visualizado no quadro 8:

wi(n+1) = wi(n) +Δ i
e
Δi = c * s * xi,
onde: Δ i = correção associada com a i-ésima entrada xi
wi(n+1) = novo valor do peso
wi(n) = valor antigo do peso
s = saída desejada - saída obtida
c = constante de aprendizado

Quadro 8 – Regra Delta

2.3.2.2.3 UTILIZAÇÃO DA REDE NEURAL

Após ter sido realizado o treinamento satisfatório da rede neural, ela está pronta para ser utilizada. Segundo Tafner (1996, p. 81) a fase de utilização é propriamente a execução da rede neural, que se inicia quando uma entrada é apresentada à rede e termina quando a rede gera uma saída.

É importante lembrar que, na fase de utilização, nenhum ajuste de peso sináptico é realizado. O processo de utilização consiste na obtenção de uma resposta da rede a um estímulo de entrada.

Caso ainda surja a necessidade de reconhecimento de novos conjuntos de dados ainda ou caso sejam identificados erros significativos na execução da rede, será necessária uma manutenção na rede neural. A manutenção pode ser realizada pelo processo de treinamento ou mesmo por alterações na definição da rede, dependendo do caso.

2.3.2.3 MODELO DE REDES NEURAIS

Diferentes combinações quanto aos aspectos estruturais das redes neurais artificiais resultam na existência de diferentes modelos, cada um com sua arquitetura, aprendizagem e capacidade diferente dos demais. As aplicações práticas também variam para cada modelo.

Segundo Medeiros (2003, p.10), os modelos desenvolvidos sempre se basearam em modelos biológicos ou situações reais. Assim, sempre se procurou reproduzir o funcionamento fisiológico ou simbólico do cérebro humano.

Loesch e Sari (1996, p. 5) apresentam uma relação de redes contendo informações sobre as mesmas. Alguns modelos indicados para diferentes aplicações são exibidos na Tabela 1.

Tabela 1 – Aplicações básicas para modelos de rede neural artificial

Modelo	**Aplicação básica**
Adaline/Madaline	Filtragem de sinal adaptativo
feedforward multicamadas	Reconhecimento de padrões
Recurrent	Controle robótico
Time-Delay	Reconhecimento de fala
Rede de ligações funcionais	Classificação
Boltzmann Machine	Reconhecimento de padrões
Hopfield	Evocação autoassociativa
Neocognitron	Reconhecimento de caracteres manuscritos

Na seqüência será apresentado o modelo *feedforward* multicamadas, também conhecido como *perceptron* multicamadas.

2.3.2.4 REDES PERCEPTRON MULTICAMADAS

Segundo Loesch e Sari (1996, p. 67) a rede *perceptron* multicamadas é o modelo mais implementado dentre todas as arquiteturas conhecidas. As capacidades da rede foram responsáveis pela popularização do modelo. A rede possui, além da capacidade de abstração, a capacidade de generalização. Com isso, é capaz de classificar um padrão mesmo quando este não pertença ao conjunto de treinamento. Também é uma rede robusta, sendo assim imune a pequenas falhas nos padrões apresentados.

A arquitetura da rede pode ser visualizada na figura 20, onde a rede possui uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada neurônio recebe diversos valores de entrada e produz apenas uma saída. Os neurônios de uma mesma camada atuam em paralelo. O fluxo de processamento inicia na camada de entrada e são propagados os valores produzidos até a camada de saída. As equações utilizadas no processamento para frente são exibidas no quadro 9.

$$s_j^{(K)} = w_{0j}^{(K)} + \sum_{i=1}^{N_k} w_{ij}^{(K)} \cdot x_i^{(K-1)}$$

onde:

- $x_i^{(K)}$ saída da função de ativação do neurônio i da camada k
- $s_j^{(K)}$ soma ponderada dos pesos pelas entradas
- $w_{ij}^{(K)}$ pesos das conexões sinápticas na entrada do neurônio j da camada k, onde i é o índice da conexão.
- N_k número de neurônios da camada k

e

$$x_j^{(K)} = f(s_j^{(K)})$$

Quadro 9 – Equações *feedforward*

O treinamento de uma rede *perceptron* multicamadas segue o procedimento descrito na seção 2.2.2.2.2, onde são apresentados à rede pares de treinamento. Ao se apresentar um padrão de entrada, o fluxo é alimentado pra frente, camada por camada. O valor de saída obtido é comparado com a saída desejada, e em caso de erro ocorre um reajuste dos pesos da rede. Este ajuste é realizado de trás para frente, ou seja, da camada de saída à primeira camada oculta. O algoritmo de retropropagação é apresentado no quadro 10.

Passo 1. Inicialize os pesos $w_{ij}^{(K)}$ da rede om valores aleatórios próximos a zero.

Passo 2. Seja (x,d) o par de treinamento. Aplique o vetor x na camada de entrada e propague a rede até a camada de saída. Seja y a saída da rede, calcule o erro quadrático $\varepsilon^2 = \sum_{j=1}^{m} (d_j - y_j)^2$. Se for inferior ao valor de tolerância, pare com sucesso, senão prossiga.

Passo 3. Faça k = última camada.

Passo 4. Para todo elemento j da camada k faça:

- Calcule $\varepsilon_j^{(k)}$ empregando

$$\varepsilon_j^{(k)} = d_j - y_j \quad \text{se k for a última camada}$$

$$\varepsilon_j^{(k)} = \sum_{i=1}^{N_{k+1}} (\delta_i^{(k+1)} \cdot w_{ji}^{(k+1)}) \quad \text{se for uma camada oculta}$$

- Calcule $\delta_j^{(k)}$ empregando $\delta_j^{(k)} = \varepsilon_j^{(k)} \cdot f'(s_j^{(k)})$.

Passo 5. k← k - 1. Se k > 0 vá para o passo 4, senão prossiga no passo 6.

Passo 6. Recalcule todos os pesos de conexão da rede empregando

$$W_j^{(k)}(n+1) = W_j^{(k)}(n) + 2\mu\delta_j^{(k)}(n)x_j^{(k)}(n)$$

tome outro par de treinamento e retorne ao passo 2.

Quadro 10 – Algoritmo de retropropagação

Loesch e Sari (1996, p. 77) fazem um comentário sobre a implementação do algoritmo de retropropagação em software, onde existe a necessidade do cálculo da derivada da função

de transferência no passo 4. Segundo ele este cálculo pode ser realizado com maior eficiência computacional quando a função de transferência for alguma função logística. Neste caso, a derivada pode ser obtida pela equação exibida no quadro 11.

$$f'(x) = f(x) * (1 - f(x))$$

Quadro 11 – Equação da derivada da função logística

Para usufruir desta vantagem, pode então ser utilizada como função de transferência a função logística apresentada por Tafner (1996, p. 61), que pode ser vista no quadro 12.

$$f(x) = \frac{1}{(1 + e^{-x})}$$

Quadro 12 – Equação da função de transferência

3 DESENVOLVIMENTO DO PROTÓTIPO

Neste capítulo são abordadas as atividades relativas ao projeto e desenvolvimento do protótipo proposto pelo trabalho.

3.1 ESPECIFICAÇÃO DE REQUISITOS

O protótipo de sistema a ser desenvolvido deve analisar imagens de produtos capturadas previamente por uma câmera de vídeo, com o objetivo de detectar defeitos de produção nos mesmos. A inspeção deve ser baseada em parâmetros definidos pelo usuário, portanto o sistema deve permitir a configuração prévia das diferentes análises a serem realizadas.

O protótipo deve possuir independência quanto ao produto a ser inspecionado, permitindo assim analisar diferentes produtos sem a necessidade de nova implementação.

O protótipo deve realizar as análises em tempo real, considerando-se que sistemas de inspeção automatizada são inseridos em uma linha de produção onde interagem com um sistema de descarte, de maneira a descartar um produto defeituoso imediatamente após a detecção do defeito.

3.2 ESPECIFICAÇÃO

O sistema será representado a seguir através da utilização de alguns dos diagramas da UML, visto que, segundo Cardoso (2003, p.1), tem se tornado a cada dia um padrão mais usado para análise e projeto de software. Para a confecção dos diagramas foi utilizada a ferramenta *Rational Rose* (IBM CORPORATION, 2004).

3.2.1 DIAGRAMAS DE CASO DE USO

A figura 21 apresenta o diagrama do caso de uso Configurar inspeção. Este caso de uso representa a funcionalidade no que diz respeito à definição e configuração de cada tipo de análise a ser realizada sobre as imagens dos produtos. O usuário deve informar quais as análises a serem consideradas, e, para cada uma delas, informar parâmetros inerentes às mesmas, necessários a sua realização.

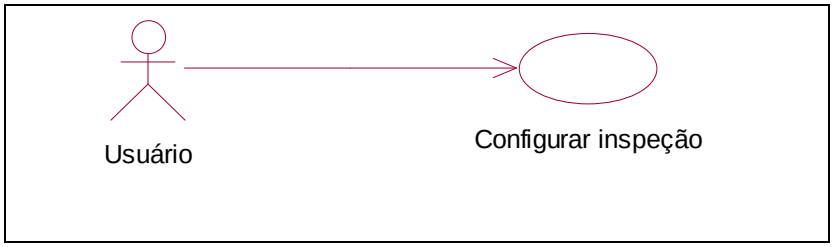

Figura 21 – Caso de uso Configurar inspeção

Na figura 22 pode-se observar o diagrama do caso de uso Inspecionar produto, que representa a funcionalidade no que diz respeito à inspeção dos produtos. O sistema externo responsável pela captura das imagens sinaliza ao sistema de inspeção, através de uma interrupção, a passagem de um produto na linha de produção. O sistema de inspeção realiza as análises previamente configuradas pelo usuário. Baseando-se nos parâmetros especificados, o sistema determina a qual classe o produto se enquadra. Caso o produto se enquadre na classe de produtos reprovados, o sistema de inspeção sinaliza o resultado ao sistema de descarte, através de uma saída digital. Caso o produto se enquadre na classe de produtos aprovados o sistema não faz nada, deixando-o passar.

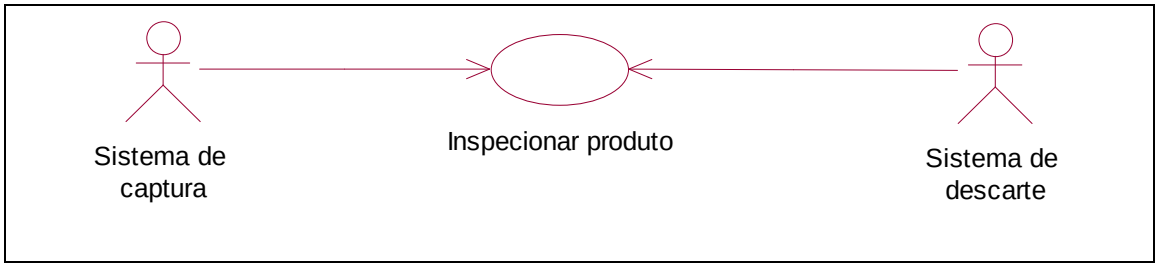

Figura 22 – Caso de uso Inspecionar produto

Os atores presentes neste caso de uso são sistemas externos, normalmente presentes em uma aplicação do sistema de inspeção em uma linha de produção industrial. Foram mantidos estes atores para ilustração de uma situação real de aplicação. Porém, no protótipo desenvolvido o papel dos sistemas externos será realizado por um usuário. Os sinais de interrupção e saídas digitais serão representados através da interface.

3.2.2 DIAGRAMAS DE CLASSE

A seguir serão expostos os diagramas de classe do protótipo implementado. Os diagramas estão divididos em diferentes níveis, para um melhor entendimento.

3.2.2.1 CLASSES DE PROCESSAMENTO DE IMAGEM

Na seqüência serão descritas as classes utilizadas no desenvolvimento do protótipo mais relacionadas ao processamento e análise de imagens.

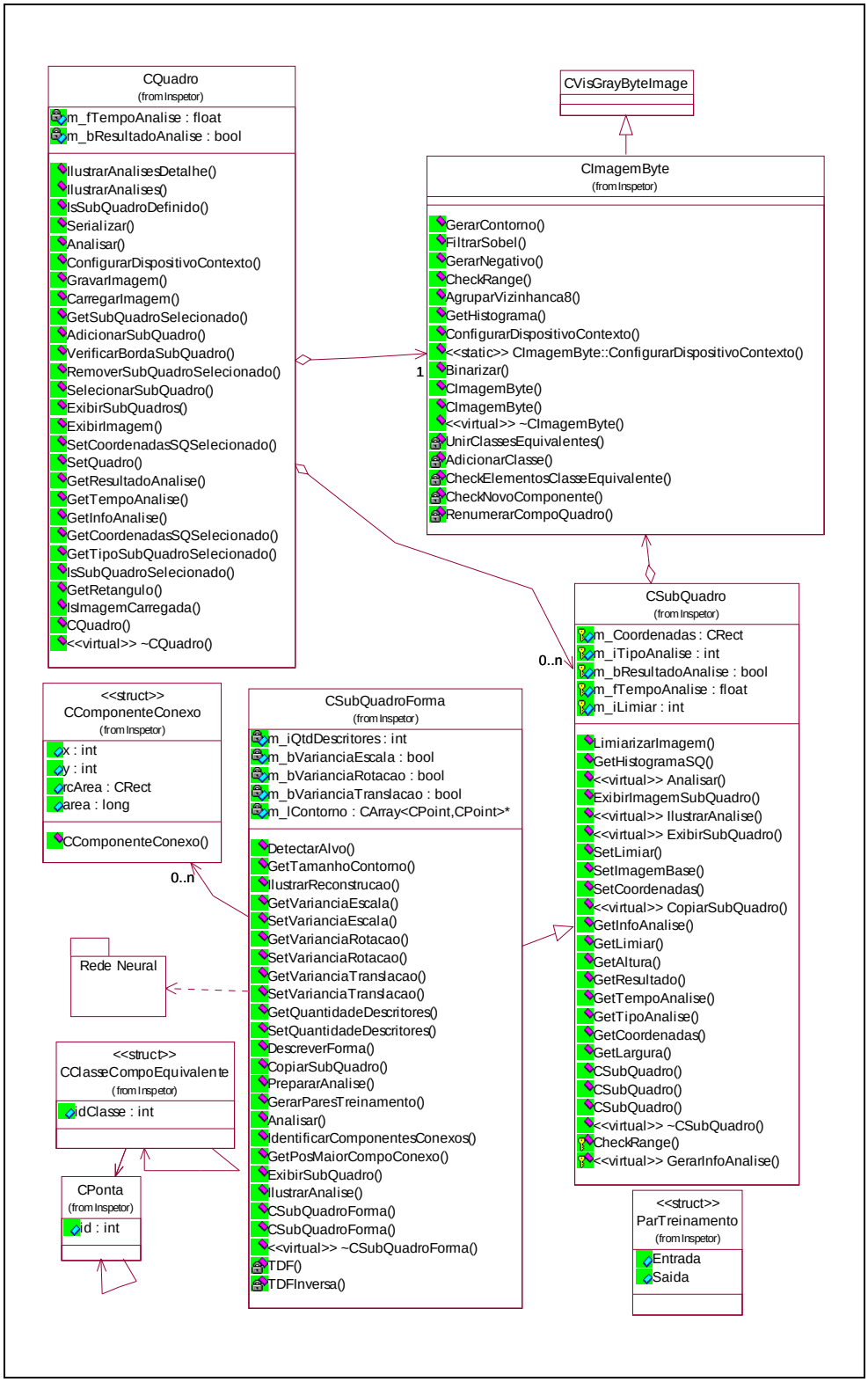

Figura 23 – Diagrama de classes de processamento e análise de imagem

O diagrama de classe exibido na figura 23 exibe as classes responsáveis pelo processamento e análise de imagens. No diagrama apresentado, as classes referentes à rede neural estão agrupadas em um pacote. Furlan (1998, p. 310) explica que esta notação da UML é indicada para agrupar componentes relacionados entre si, para uma melhor organização dos diagramas. As classes da rede neural implementada serão vistas em detalhe na seqüência.

3.2.2.1.1 CLASSE CIMAGEMBYTE E O PROCESSAMENTO DE IMAGEM

A classe CImagemByte é a estrutura utilizada para representar imagens no protótipo implementado. Além de ser utilizada como estrutura para qualquer imagem criada durante os processamentos, também é utilizada para armazenar e manipular as imagens capturadas pelo sistema de aquisição e que serão analisadas pelo sistema de processamento. Observa-se que, conforme o próprio nome indica, esta imagem somente pode ser utilizada para imagens em tons de cinza.

Esta classe é uma especialização da classe CVisGrayByteImage, que compõe a biblioteca Vision SDK e que será discutida mais a diante. A especialização objetiva adicionar funcionalidades de processamento de imagens, já que este tipo de processamento não está presente na classe mãe. As principais vantagens oferecidas pela utilização desta classe da biblioteca CVisGrayByteImage, em contrapartida aos tipos nativos oferecidos pela linguagem C++ utilizada na implementação do protótipo, são o acesso otimizado à memória que representa a matriz de *pixels*, funções de exibição em dispositivos de contexto, rotinas de gravação e leitura no disco, além da presença de rotinas específicas à aquisição de imagem independente de dispositivo. Esta última funcionalidade não é aproveitada no protótipo desenvolvido neste trabalho, porém poderá ser utilizado nas possíveis expansões futuras do mesmo.

O objetivo na criação desta classe é centralizar todo e qualquer tipo de processamento de imagem que se faça necessário e que tenha ação somente sobre os *pixels*. Além da memória utilizada para armazenar a matriz de *pixels* e do cabeçalho contendo propriedades básicas da imagem, nenhuma outra informação relativa à análise é armazenada. A idéia é que, não abrangendo as funções e estruturas utilizadas na análise, esta classe poderia ser utilizada na implementação de outros sistemas, como um editor de imagens, por exemplo.

Uma das funcionalidades implementadas na classe é a rotulação dos componentes conexos presentes na imagem, baseada no algoritmo apresentado na seção 2.1.2.3 e implementada na função AgruparVizinhança8(). Para a rotulação dos componentes são utilizadas as estruturas CPonta e CComponenteEquivalente que podem ser visualizados no diagrama de classes exibido na figura 23.

Outra funcionalidade importante é o processamento dos operadores de Sobel, que foi implementada na função FiltrarSobel(). Esta função refere-se ao filtro por derivada e que é utilizado para o realce de *pixels* localizados em fronteiras. Uma explicação mais detalhada da técnica pode ser encontrada na seção 2.1.4.2.

3.2.2.1.2 CLASSE CQUADRO E A ANÁLISE GERAL

A classe CQuadro representa o quadro a ser analisado pelo sistema. Além de possuir agregada a estrutura contendo a imagem a ser analisada, esta classe agrega e provê acesso à lista de análises definidas pelo usuário, representadas por subquadros e que serão descritos posteriormente.

Esta classe não realiza nenhuma análise direta sobre os *pixels* da imagem, mas é responsável pela efetivação e obtenção do resultado da análise geral, através de troca de mensagens com os subquadros que executam as análises parciais. Possui funções utilizadas na definição e gerenciamento dos subquadros, assim como funções de desenho e exibição.

3.2.2.1.3 CLASSE CSUBQUADRO E AS SUBANÁLISES

A classe CSubquadro é a classe base para os subquadros de análise a serem implementados. Um subquadro referencia determinada região do quadro a ser analisado, delimitando a área a ser considerada. Também possui funcionalidades e estruturas utilizadas nas análises em geral.

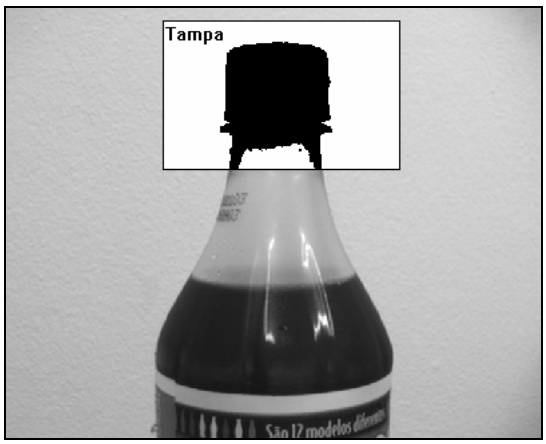

Figura 24 – Subquadro de analise

A idéia por trás de um subquadro é a de que nem todos os tipos de análise necessitam ser realizadas em toda a área da imagem. A figura 24 demonstra a definição de um subquadro para análise de tampa em uma garrafa. Neste exemplo, a análise de integridade de tampa sobre garrafas pode ser direcionada à região do quadro capturado onde se estime que o gargalo deva estar localizado. Esta delimitação da região de interesse reflete diretamente na velocidade de análise, já que o processamento não considerará todos os *pixels* da imagem.

3.2.2.1.4 CLASSE CSUBQUADROFORMA E A ANÁLISE DE FORMA

A classe CSubQuadroForma é uma especialização da classe CSubquadro com o objetivo de disponibilizar um tipo de análise baseado na forma de produtos. Este tipo de análise possibilita a inspeção de diferentes tipos de produtos, já que a presença de muitos defeitos de produção tem reflexo direto sobre a forma externa dos mesmos.

A análise de forma de produtos implementada é realizada observando o contorno dos mesmos. O contorno é considerado como sendo o conjunto de *pixels* localizados na fronteira do produto na imagem digital. O procedimento para a extração do contorno do produto e sua análise é descrito abaixo.

Para a obtenção dos contornos presentes no quadro analisado, foi implementada na classe CImagemByte a função FiltrarSobel(), que correspondente à técnica de detecção de bordas por operadores de Sobel, descrita na seção 2.1.4.2. Conforme exibido na figura 12, a partir do processamento do operador de Sobel os valores dos níveis de cinza dos *pixels* presentes na fronteira do produto são aproximados a 255. Os valores dos níveis de cinza dos *pixels* que não pertencerem à fronteira serão aproximados a 0. Observe que, conforme dito

anteriormente, todo o processamento sobre os *pixels* da imagem que não resultem em informação foram implementados na classe CImagemByte.

Conforme explicado, após a aplicação do filtro operador de Sobel os valores dos níveis de cinza dos *pixels* são aproximados a 0 ou 255. Para evitar que os processamentos que virão a seguir considerem faixas de valores próximos as estes extremos, é realizada a binarização da imagem através da técnica de limiarização descrita na seção 2.1.4.1. Esta funcionalidade está implementada na função Binarizar() da classe CImagemByte. A binarização permite que somente sejam considerados valores de 0 ou 255.

A partir da imagem segmentada pela detecção de bordas e pela limiarização, se poderia varrer a imagem e identificar os *pixels* que compõe um contorno. Porém nem todos os *pixels* identificados serão necessariamente do mesmo objeto. Podem existir mais de um objeto em um quadro analisado, assim como mais de um contorno pode pertencer a um único objeto. Para resolver este problema foi utilizada a técnica de rotulação de componentes conexos descrita na seção 2.1.2.3. A função AgruparVizinhança8() implementa esta técnica na classe CImagemByte. Através dela os diferentes agrupamentos de *pixels* presentes na imagem serão rotulados com diferentes valores de níveis de cinza. Observa-se agora que, como a própria faixa de valores dos *pixels* entre 0 e 255 é utilizada como identificador dos componentes, existe uma limitação a 253 diferentes agrupamentos. Uma abordagem diferente com utilização de nova estrutura poderia resolver o problema, porém esta quantidade mostra-se suficiente para a resolução do problema aqui exposto.

A função IdentificarComponentesConexos() foi implementada na classe CSubQuadroForma para extrair informações dos componentes conexos da imagem, sendo também criada uma estrutura para armazenar informações básicas dos componentes. Para cada componente identificado a função insere em uma lista associada à classe CSubQuadroForma uma instância da estrutura CComponenteConexo, contendo informações como posição dos cantos, área, além de um ponto inicial para acesso.

As informações extraídas dos componentes são utilizadas então para decidir qual dos contornos será considerado. Partindo do princípio que a área de atuação de um ponto de inspeção em uma linha de produção deve ser controlada e também previamente preparada, assume-se que o contorno a ser inspecionado é o maior contorno presente no quadro. Somente

o componente conexo que possuir a maior área será considerado, já que representa o maior contorno presente.

Uma vez identificado o componente conexo a ser considerado, pode-se extrair as coordenadas dos *pixels* que compõe o contorno. A função GerarContorno() da classe CSubQuadroForma percorre todo o contorno de um componente conexo, partido do ponto inicial recebido como parâmetro. Após isso, retorna uma lista contendo as coordenadas de todos os pontos pertencentes à fronteira do componente.

Com a lista de coordenadas dos pontos no plano xy que formam o contorno, já há informação suficiente para diferenciar produtos. Porém, a lista gerada geralmente é muito grande. Considerando-se que os quadros capturados por câmeras específicas para aplicações de inspeção normalmente são compostos por 640 *pixels* de largura por 480 *pixels* de altura, frequentemente são obtidos contornos com mais de 1000 *pixels*. Esta quantidade de informação é inviável para a maioria das técnicas de interpretação.

Para resolver este problema foi implementada a técnica de descrição por Fourier apresentada na seção 2.1.5.2. A função TDF() implementada na classe CSubQuadroForma gera, a partir de uma lista que representa um contorno, uma aproximação com precisão variável. Pelo uso desta função, pode-se reduzir em muito a quantidade de informação que será utilizada para descrever a forma dos produtos.

Após a redução da quantidade de informações que descrevem um produto, pode-se interpretar os mesmos através da técnica de redes neurais, descrita na seção 2.2. A classe CSubQuadroForma é associada uma rede neural implementada na classe CRede. Detalhes de implementação internos à rede neural serão abordados na seção seguinte. A utilização desta classe ocorre em dois momentos distintos: na preparação da análise e na execução da análise.

A preparação da análise deve anteceder a execução da mesma. No momento da preparação, a rede neural associada é criada e posteriormente treinada. Na criação da rede são utilizados parâmetros informados pelo usuário, exceto pela quantidade de neurônios da camada de entrada, que será igual à quantidade de descritores utilizados na representação do contorno, e pela quantidade de neurônios da camada de saída, que será igual a um, podendo representar assim aprovação e reprovação.

Criada a rede neural, é preparada através da função GerarParesTreinamento() uma lista contendo pares de treinamento que são utilizados no treinamento da mesma. Cada par possui um padrão de entrada e sua saída desejada. Para a preparação dos pares de treinamento deve-se possuir quadros com padrões de produto aprovados em um diretório e quadros com padrões de produtos reprovados em outro diretório. Para cada par, a entrada é obtida pela execução dos passos descritos anteriormente para a obtenção dos descritores de Fourier. A saída desejada será zero para os padrões contidos no diretório de produtos reprovados, e será um para os produtos contidos no diretório de produtos aprovados. A lista de pares de treinamento é passada então à rede neural, para treinamento da mesma. Após o treinamento da rede, a subanálise estará pronta para a execução.

No momento de execução, o processamento para a obtenção dos descritores de Fourier ocorre para cada quadro analisado, conforme procedimento descrito anteriormente. Os descritores são apresentados então à rede neural. A rede neural é ativada. Após isso, obtém-se o resultado do processamento da mesma, que é um valor real. Caso este valor seja próximo a zero o resultado da análise será reprovado, caso o valor seja próximo a um o resultado será aprovado.

3.2.2.2 CLASSES DE REDE NEURAL

Na seqüência serão descritas as classes utilizadas na implementação da rede neural utilizada.

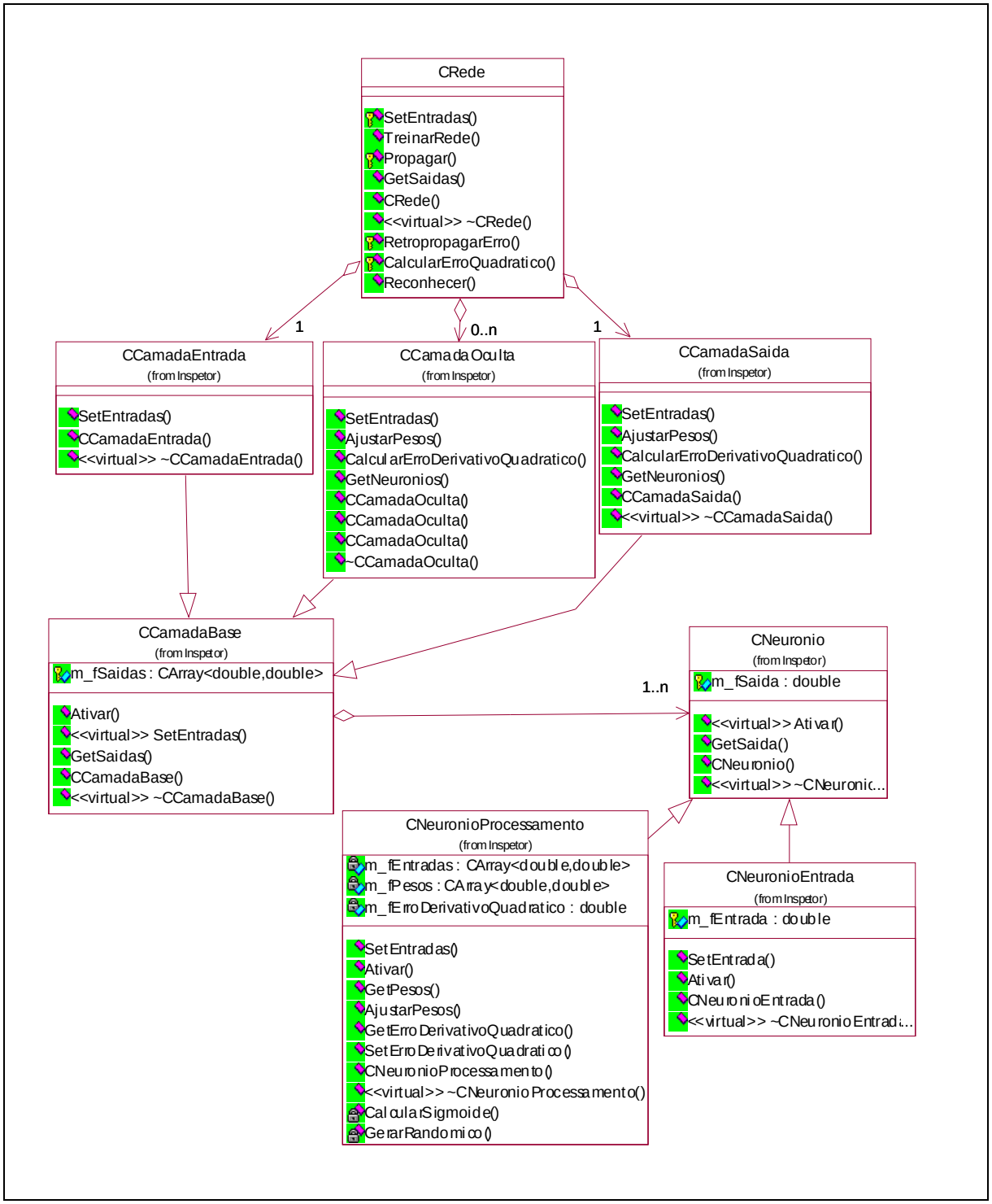

Figura 25 – Diagrama de classes da rede neural

O diagrama de classes exibido na figura 25 refere-se ao pacote de classes as classes que compõe a rede neural utilizada na análise de forma como método de interpretação de produtos.

Observa-se que a implementação é baseada nas equações de propagação exibidas no quadro 9 e no algoritmo de retropropagação exibido no quadro 10. Como a implementação segue o paradigma de orientação a objeto e deve aproximar-se do modelo real, as equações e os passos do algoritmo encontram-se distribuídos nas diferentes classes que compõe a arquitetura da rede. A indicação da localização de cada passo será feita nas seções que seguem.

3.2.2.2.1 CREDE

A classe CRede provê acesso a funcionalidades de uma rede neural, incluindo os métodos para sua criação, treinamento e execução. Esta classe é responsável pela interface da rede com o ambiente externo, assim como pelo gerenciamento da rede como um todo. O modelo de rede neural implementado é o *perceptron* multicamadas, descrito na seção 2.2.2.4.

Na criação da rede são informados parâmetros relativos à sua arquitetura da mesma. Seguindo a arquitetura encontrada na literatura, a rede implementada é composta por uma camada de entrada, uma ou várias camadas ocultas e uma camada de saída. Todas as camadas podem ter *N* neurônios.

Para realização do treinamento, a classe implementa o algoritmo de retropropagação, exibido no quadro 10, através da função booleana TreinarRede(). Os passos do algoritmo mais associados às camadas, como o cálculo de erro da camada, foram implementados nas classes que representam as mesmas. Nestes casos, a classe CRede envia uma mensagem às camadas para a realização do cálculo. A classe também implementa a função CalcularErroQuadratico(), que calcula o erro quadrático descrito no passo 2 do algoritmo. Adicionalmente ao algoritmo, foi implementado um controle em que, caso a quantidade de iterações tenha ultrapassado um valor estipulado e o erro da rede ainda não tenha atingido um valor inferior à tolerância informada, o *loop* de treinamento para e a função retorna falso. Este controle foi adicionado para os casos em que a rede não consegue aprender, e sem o mesmo continuaria o processamento infinitamente.

O reconhecimento de um padrão através da rede é implementado na função Reconhecer(), onde é informada como parâmetro a entrada para a rede. Pela chamada a esta função a rede é alimentada com os valores de entrada. É chamada então a função Propagar(). Esta função ativa todas as camadas, da camada de entrada à camada de saída, repassando

entre elas os resultados das ativações. O retorno da função é o resultado da execução da rede, diante do padrão apresentado.

3.2.2.2.2 CLASSE CCAMADABASE

Pelas diferenças existentes entre os tipos de camadas que compõe uma rede neural do tipo *perceptron* multicamadas, optou-se por criar diferentes classes de camadas ao invés de utilizar apenas uma, onde fossem implementadas todas as funções.

A classe CCamadaBase representa a classe mãe utilizada para especialização de diferentes tipos de camadas. Ela possui variáveis e implementa funções comuns aos diferentes tipos de camada implementados. Pode-se citar a existência de listas para os neurônios e para as saídas, assim como funções de ativação, informação das entradas e acesso às saídas.

3.2.2.2.3 CLASSE CCAMADAENTRADA

A classe CCamadaEntrada representa a camada de entrada da rede neural. Observe que, como as entradas dos neurônios desta camada são alimentadas por fonte externa, ela não é considerada no algoritmo de retropropagação de erro. A única função dos neurônios desta camada é armazenar a informação de entrada para ser passada para a camada seguinte. Na verdade, esta camada existe para uma aproximação do modelo, sendo que não necessariamente precisaria existir.

3.2.2.2.4 CLASSE CCAMADAOCULTA

A classe CCamadaOculta representa as camadas ocultas utilizadas na rede neural. Algumas particularidades a diferenciam da camada de saída.

Nesta camada é implementada a função CalcularErroDerivativoQuadratico(), conforme descrito no passo 4 do algoritmo apresentado no quadro 10. Nesta função, para cada neurônio da camada é realizado o cálculo do erro a ser considerado no ajuste sináptico. O valor do erro é atribuído a cada neurônio através da função SetErroDerivativoQuadrático(). Assim, existe um cálculo de erro específico para os neurônios da camada oculta. O cálculo de erro exige o cálculo da derivada da função de transferência. Na seção 2.2.2.4 existe uma observação exclusiva referente a este cálculo, sendo que esta foi seguida na implementação.

Adicionalmente também existe a função AjustarPesos(), utilizada para indicar a todos os neurônios presentes na camada o momento de ajuste dos pesos sinápticos.

3.2.2.2.5 CLASSE CCAMADASAIDA

A classe CCamadaSaida representa a camada de saída da rede neural. Assim como na camada CCamadaOculta, esta classe implementa uma função CalcularErroDerivativoQuadratico(). Porém conforme diferenciado no passo 4 do algoritmo apresentado no quadro 10, existe um cálculo específico para os neurônios da camada de saída. Também o cálculo da derivada da função de transferência dos neurônios foi implementado baseado no comentário específico sobre o assunto e que pode ser visto na seção 2.2.2.4.

Nesta classe também existe a função AjustarPesos(), similar à existente na classe CCamadaOculta.

3.2.2.2.6 CLASSE CNEURONIO

Conforme comentado anteriormente, a função dos neurônios na camada de entrada difere da função que assumem nas camadas de saída e camadas ocultas. Na camada de entrada o neurônio não realiza processamento sobre o valor de entrada. Já nas outras camadas o neurônio deve processar as entradas para gerar uma saída. Por este motivo foram criadas diferentes classes de neurônio, especializadas a partir de uma classe base.

A classe CNeurônioBase representa a classe mãe utilizada para especialização dos diferentes tipos de neurônios. Ela possui variáveis e implementa funções comuns aos diferentes tipos de neurônios implementados. Pode-se citar a variável da saída, assim como funções para ativação e acesso à saída. A função para alimentação dos valores e as variáveis para os mesmos foram implementados nas especializações, já que difere entre elas.

3.2.2.2.7 CLASSE CNEURONIOENTRADA

A classe CNeuronioEntrada representa os neurônios a serem utilizados na camada de entrada. Este neurônio não realiza processamento sobre os valores de entrada. Desta forma a função de ativação nesta classe simplesmente repassa-os às saídas. Por este motivo, também não possuem pesos e funções de ajuste.

3.2.2.2.8 CLASSE CNEURONIOPROCESSAMENTO

A classe CNeuronioProcessamento representa os neurônios a serem utilizados tanto nas camadas ocultas quanto na camada de saída.

Diferente da classe CNeurônioEntrada, esta classe realiza processamento considerando os valores de entrada e seus respectivos pesos. Este cálculo é implementado na função Ativar() e segue as equações exibida no quadro 9. A segunda equação refere-se ao processamento da função de transferência. Este cálculo foi implementado na função CalcularSigmoide(), e o tipo da função escolhida foi a função sigmoidal descrita por Tafner (1996, p. 61). A escolha desta função foi baseada no comentário de Loesch e Sari (1996, p. 77), em que explica que o uso de uma função deste tipo facilita o cálculo da derivada, necessária no calculo de erro realizado pelas camadas oculta e de saída. Maiores detalhes podem ser encontrados na seção 2.2.2.4.

Esta classe implementa ainda o cálculo de ajustes de peso, implementado na função AjustarPesos() e descrita no passo 6 do algoritmo exibido no quadro 10. Neste cálculo são utilizados a taxa de aprendizado e o valor do erro derivativo quadrático. A taxa de aprendizado é informada à rede neural no momento de criação da classe CRede. Já o erro derivativo quadrático para cada neurônio é repassado ao mesmo no momento da execução da função CalcularErroDerivativoQuadrático() da classe de camada em que se encontra.

3.2.3 DIAGRAMAS DE SEQUENCIA

A seguir são exibidos e comentados os diagramas de seqüência dos principais casos de uso do protótipo a ser implementado.

O diagrama exibido na figura 26 refere-se ao caso de uso Configurar Inspeção.

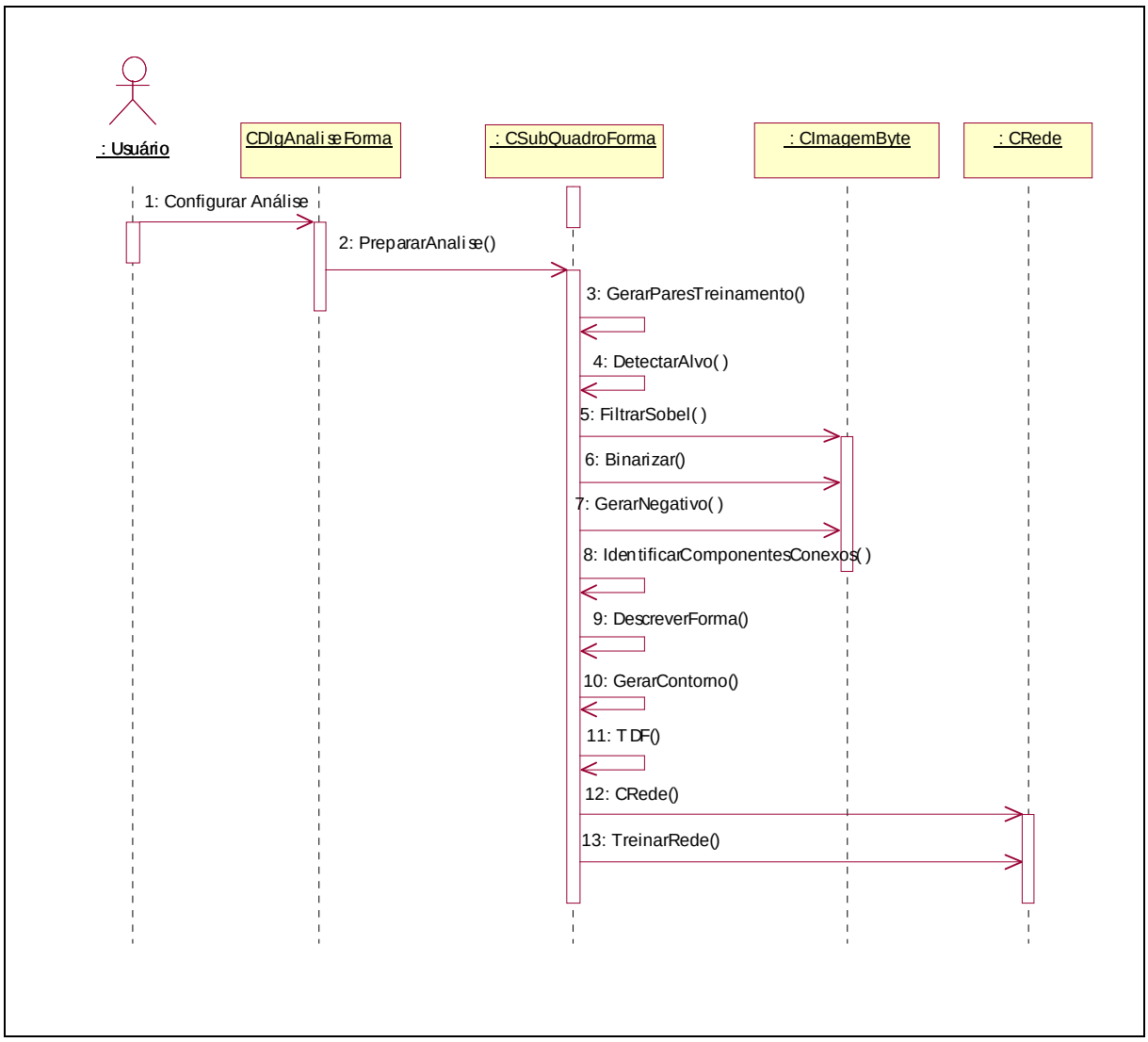

Figura 26 – Diagrama de seqüência Configurar Inspeção

A preparação da análise deve anteceder a sua execução. Neste momento, a rede neural a ser utilizada é criada e posteriormente treinada com o conjunto de treinamento.

O processo de treinamento inicia com a geração de um evento pelo usuário na caixa de diálogo relativa a preparação da análise. A função associada a este evento solicitará a execução da função PrepararAnalise() do objeto da classe CSubQuadroForma. Através dos parâmetros desta função, é enviada uma série de informações fornecidas pelo usuário no diálogo, e que serão utilizadas durante o processo de criação e treinamento da rede neural.

Além de informar a quantidade de descritores a serem utilizados na descrição da forma, a função PrepararAnalise() solicita a geração do conjunto de treinamento a partir dos quadros padrão, que devem estar localizados em dois diretórios, previamente informados pelo usuário. A função GerarParesTreinamento() prepara os pares de treinamento. Nesta função,

para cada padrão é obtida a descrição do produto na forma de descritores de Fourier, conforme procedimento abaixo.

Inicialmente, é chamada a função DetectarAlvo() da própria classe, que deve identificar o produto e destacá-lo de outros objetos possivelmente presentes na imagem. Para isso, a função realiza uma chamada à função de FiltrarSobel() do objeto CImagemByte que representa a imagem, objetivando destacar contornos presentes na mesma. Após destacados os contornos, é chamada a função IdentificarComponentesConexos() para se obter informações dos contornos e a partir delas identificar o contorno do produto alvo. Após isso, a função própria DescreverForma() é chamada. Será chamada a função GerarContorno() para obter uma lista contendo as coordenadas dos pontos que compõe a fronteira do segmento de *pixels* que representa o contorno do produto. Como a lista gerada anteriormente é muito grande, ela é representada por descritores de Fourier pela chamada à função TDF(), considerando a quantidade de descritores desejados informada pelo usuário.

De mãos da descrição do produto presente no quadro padrão, a este é associada uma saída desejada. A saída desejada será zero para os padrões contidos no diretório de produtos reprovados, e será um para os produtos contidos no diretório de produtos aprovados. Desta forma são criados os pares de treinamento que compõe o conjunto de treinamento.

Após a criação do conjunto de treinamento, o objeto da rede neural é criado através do construtor da classe CRede(). No momento da criação da rede são informados parâmetros referentes a quantidade de entradas, quantidade de saídas, quantidade de neurônios na camada oculta e quantidade de camadas.

A função TreinarRede() é chamada após a criação da rede, sendo o conjunto de treinamento passado como parâmetro. Neste momento, a rede realizará o treinamento baseado nos padrões apresentados. A seqüência de eventos a partir da ativação deste método pode ser visualizada no diagrama a seguir. Depois de treinada a rede, o objeto CSubQuadroForma está pronto para proceder as análises de forma.

O diagrama apresentado na figura 27 representa a seqüência de eventos no momento do treinamento da rede neural, realizado pela classe CRede. O presente diagrama dá seqüência ao diagrama anterior, a partir da chamada do método TreinarRede().

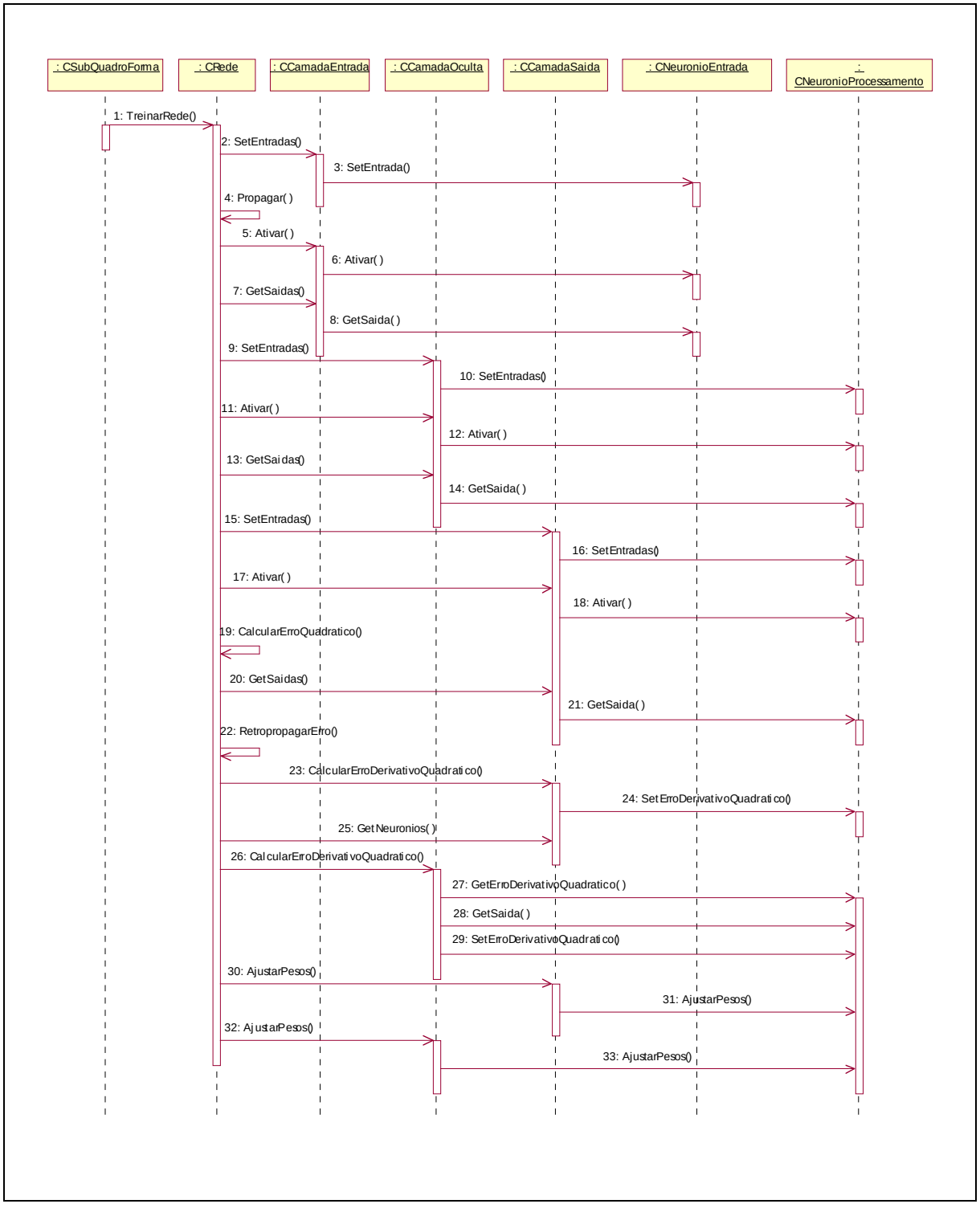

Figura 27 – Diagrama de seqüência do treinamento da rede neural

Ativada pelo objeto de análise da classe CSubQuadroForma, a função TreinarRede() realiza o treinamento da rede neural baseado no conjunto de treinamento, contido em uma lista de pares de treinamento. O procedimento de treinamento baseia-se no algoritmo apresentado no quadro 10, contido na seção 2.2.2.4.

Para cada par da lista do conjunto de treinamento, a entrada será apresentada à rede pela atribuição do mesmo à camada de entrada. Esta atribuição ocorre pelo método SetEntradas(). Atribuídos os valores de entrada, o processamento da rede inicia com a chamada da função Propagar().

No procedimento de propagação, para cada camada ocorre a atribuição dos valores obtidos pelo processamento da camada anterior. A camada atual então é ativada, e o resultado obtido é utilizado para alimentar a próxima camada. Este procedimento ocorre da camada de entrada até a camada de saída, e é realizado pela ativação dos métodos SetEntradas(), Ativar(), GetSaidas(), para cada camada.

Ao final da propagação dos valores da entrada de um par na rede, é efetuado pelo objeto CRede o cálculo do erro quadrático apresentado no passo 2 do algoritmo de treinamento, implementado na função CalcularErroQuadratico(). Caso o erro seja menor do que o valor de tolerância informado pelo usuário, o treinamento será realizado com o próximo par, senão será realizado o ajuste na rede baseado no erro obtido pelo processamento do par atual.

Caso seja necessário ajuste na rede, o objeto CRede executará a função RetropropagarErro(). Esta função implementa os passos 4 e 5 do algoritmo de treinamento. Nesta função, da camada de saída até a primeira camada oculta é chamada a função CalcularErroDerivativoQuadrático(), para cada objeto de camada. Conforme descrito pelo algoritmo, no cálculo do erro das camadas ocultas são utilizados valores de pesos e de erro dos neurônios da camada seguinte. Por este motivo, na chamada da função CalcularErroDerivativoQuadrático() para camadas ocultas, é enviado como parâmetro um ponteiro que aponta para os neurônios contidos na camada seguinte. Tanto nas camadas ocultas como nas camadas de saída, o valor do erro é calculado para cada neurônio.

Calculados os erros para cada neurônio, é chamada a função AjustarPesos() para todas as camadas ocultas e para a camada de saída. Na execução desta função, para todos os neurônios das camadas é chamada a função que implementa o ajuste sináptico definido no 6 do algoritmo de treinamento.

Este procedimento de treinamento é repetido até que, para todos os pares contidos no conjunto de treinamento, o erro seja menor que o valor de tolerância informado pelo usuário.

O diagrama exibido na figura 28 se refere ao caso de uso Inspecionar produto.

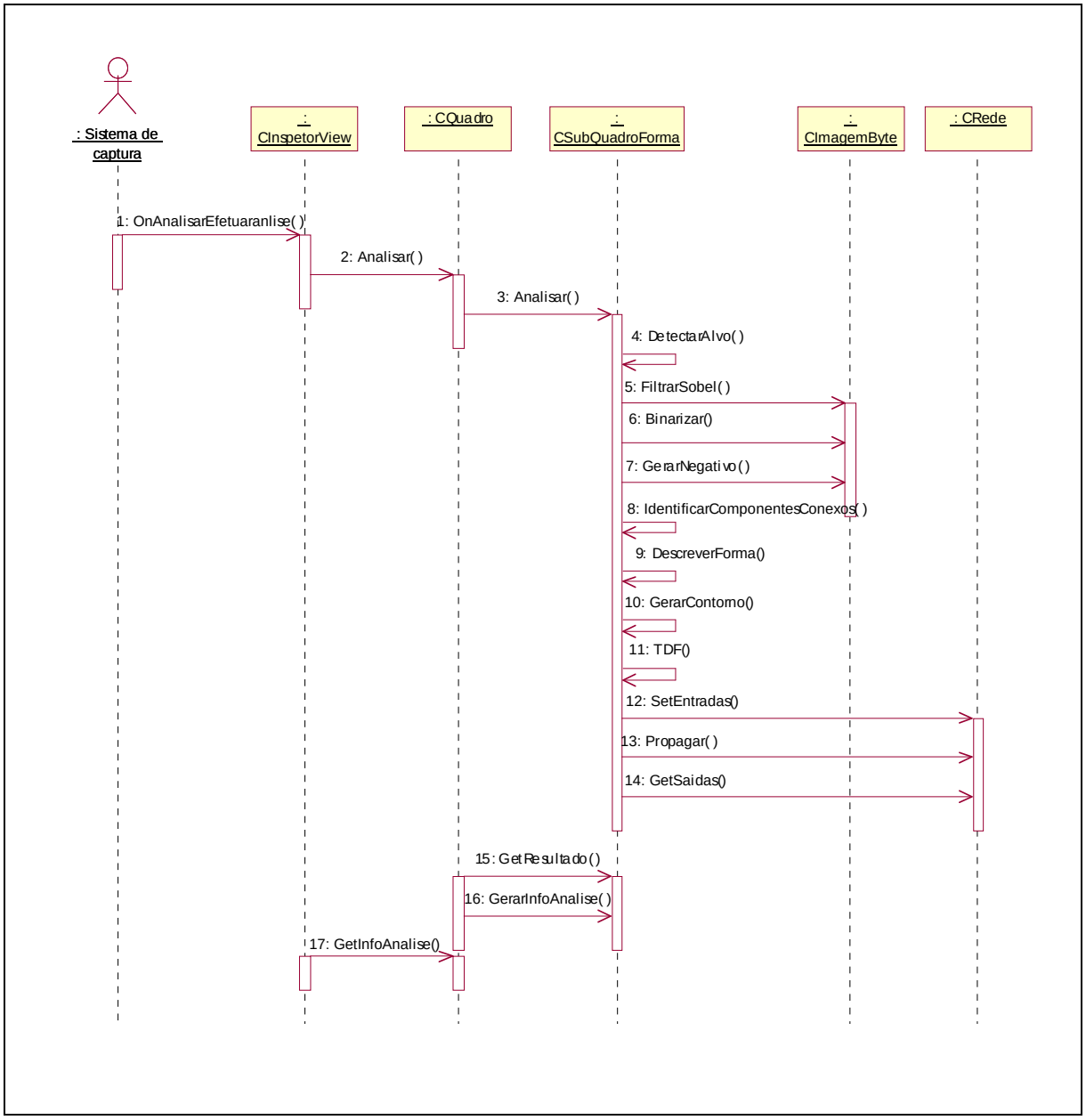

Figura 28 – Diagrama de seqüência da inspeção

A inspeção é iniciada pelo envio de uma mensagem pelo objeto da classe CInspetorView ao objeto CQuadro. Conforme comentado anteriormente, no protótipo implementado, para a classe CInspetorView o evento que encadeia o envio desta mensagem é gerado pelo comando de um usuário. Na prática, para um sistema de inspeção implantado em uma linha de produção para a realização de inspeção em linha, o evento de ativação seria uma interrupção gerada a partir do sistema de captura associada à função.

Uma vez chamada a função Analisar() do objeto da classe CQuadro, este realiza trocas de mensagens com cada um dos objetos CSubQuadro agregados, solicitando a realização de cada uma das análises parciais.

Este diagrama representa a situação em que somente uma subanálise foi definida, sendo esta uma análise de forma. Caso fossem implementados mais tipos de análise e os mesmos tivessem sido definidos, existiria uma chamada Analisar() para cada objeto CSubQuadro agregado ao objeto da classe CQuadro.

Quando o objeto da classe CSubQuadroForma recebe a mensagem para execução da função Analisar() o processo de análise da forma do produto é desencadeado. Inicialmente é chamada a função DetectarAlvo() da própria classe, que deve identificar o produto e destaca-lo de outros objetos possivelmente presentes na imagem. Para isso, a função realiza uma chamada à função de FiltrarSobel() do objeto CImagemByte que representa a imagem, objetivando destacar contornos presentes na mesma. Após isso, é chamada a função IdentificarComponentesConexos() para se obter informações dos contornos e a partir delas identificar o contorno do produto. Será chamada a função GerarContorno() para obter uma lista contendo as coordenadas dos pontos que compõe a fronteira do segmento de *pixels* que representa o contorno do produto. Como a lista gerada anteriormente é muito grande, esta é representada por descritores de Fourier pela chamada à função TDF().

A lista obtida pelo processamento anterior é utilizada para alimentar a rede neural associada, previamente criada e treinada. A função Reconhecer() é chamada, sendo enviado como parâmetro os descritores de Fourier obtidos pela descrição da forma do produto, que serão utilizados como entrada. A rede é então executada e é devolvido o valor de saída, correspondente ao resultado do processamento da rede diante da entrada apresentada.

Baseada no resultado obtido anteriormente, a função Analisar() da classe CSubQuadroForma decide a qual das classes o produto pertence: aprovados ou reprovados.

Baseado no resultado das análises parciais obtidas de cada objeto CSubQuadro, o objeto CQuadro gera o resultado da inspeção como um todo, determinando a qual das classes o produto pertence: produtos aprovados ou produtos reprovados.

Na existência de um sistema externo de descarte, este seria sinalizado para a retirada do produto caso o mesmo pertencesse à classe de produtos reprovados. Finalmente algumas

mensagens contendo informações relativas à análise são repassadas à classe CInspetorView, terminando assim o processamento da inspeção.

3.3 IMPLEMENTAÇÃO

Nesta seção são abordados detalhes relativos à implementação do protótipo.

3.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

O ambiente de desenvolvimento utilizado para a implementação do protótipo foi o Visual C++ 6.0 (MICROSOFT CORPORATION, 2004). Esta ferramenta de desenvolvimento foi escolhida por oferecer uma diversidade de recursos úteis ao desenvolvimento de aplicações, mas principalmente pela disponibilidade de várias bibliotecas para aquisição e manipulação de imagens.

A biblioteca utilizada para a manipulação de imagens é a Vision SDK (MICROSOFT CORPORATION, 2004), que esta será descrita na seção 3.3.1.1.

A biblioteca utilizada para a implementação da interface gráfica do usuário é a Microsoft Foundation Class Library (MFC), descrita na seção 3.3.1.2.

3.3.1.1 MICROSOFT VISION SDK

Conforme descrito pela Microsoft, o kit de desenvolvimento de software Vision é uma ferramenta destinada ao desenvolvimento de sistemas de manipulação de imagens. Sendo uma biblioteca de baixo nível, esta ferramenta não oferece nenhum operador para o processamento de imagens. Apesar disso, torna-se bastante útil por definir uma série de tipos específicos para suportar imagens.

Os tipos disponibilizados pela biblioteca são mais completos do que os disponibilizados pelas bibliotecas originalmente oferecidas pelo ambiente Visual Studio. As vantagens oferecidas por estes tipos se encontram nas funcionalidades de manipulação, aquisição através de câmeras, exibição, dentre outras.

O tipo definido pela biblioteca utilizado no desenvolvimento do protótipo foi o CVisGrayByteImage. Este tipo é compatível com imagens em tons de cinza, onde para cada *pixel* são utilizados 8 bits para armazenar o valor da intensidade. A classe CImagemByte,

definida na especificação do protótipo, é especializada a partir deste tipo. O objetivo da especialização foi herdar as vantagens de manipulação implementadas na classe mãe, e adicionar a ela rotinas de processamento de imagens.

Uma vantagem a ser citada sobre a utilização deste tipo de dados é a possibilidade de se criar uma imagem que não possui memória de dados própria, e sim referencia uma seção de uma imagem maior. Esta funcionalidade é utilizada na relação existente entre um objeto CQuadro e um objeto CSubQuadro. Conforme explicado na seção 3.2.2.1.2, a classe CQuadro contém a estrutura para armazenar a imagem capturada. Já a classe CSubQuadro e as classes de análise, referem-se a uma região da imagem capturada, delimitando assim a área de interesse para a análise. Para permitir que os objetos das classes de análise tenham acesso à imagem capturada, contida no objeto da classe CQuadro, uma subimagem da mesma é criada e é agregada a cada objeto de análise. Esta será uma imagem nova, contendo origem, tamanho e coordenadas próprias, porém a memória de dados da imagem será compartilhada com a imagem agregada ao objeto CQuadro. A função que possibilita a criação de uma subimagem é SubImage().

Outra funcionalidade utilizada na implementação do protótipo é o acesso rápido aos *pixels* que constituem a imagem. Existem várias maneiras de acessar o *pixel* de uma imagem. O mais comum é através do uso das coordenadas x e y referentes à posição do *pixel*. Os tipos da biblioteca disponibilizam um método de acesso mais eficiente, mantendo um vetor de ponteiros que apontam à coluna zero para cada linha da imagem. Para uma maior eficiência ao processar os *pixels* de uma imagem, primeiro deve-se encontrar o ponteiro para a coluna zero da linha desejada e depois deslocar o ponteiro pela coluna de cada *pixel*. Para encontrar o ponteiro para a coluna zero em uma linha, a função utilizada é RowPointer(). O quadro 13 mostra o código fonte do método da classe CImagemByte do protótipo implementado, onde é gerado o negativo de uma área da imagem utilizando este recurso.

```
void CImagemByte::GerarNegativo(CRect rcArea)
{
      for (int register y = rcArea.top; y < rcArea.bottom; y++)
      {
            byte *pByte = RowPointer(y);
            for (int register x = rcArea.left; x < rcArea.right; x++)
            {
                  //BRANCO = 255
                  pByte[x] = BRANCO - pByte[x];
            }
      }
}
```

Quadro 13 – Método utilizando acesso RowPointer()

Dentre as vantagens oferecidas pela utilização da biblioteca pode-se ainda citar a possibilidade de criar novos tipos de imagens, disponibilidade de tipos com vários canais de cor e a aquisição de imagens independente de dispositivo.

3.3.2 MICROSOFT FOUNDATION CLASS

A MFC é uma infra-estrutura de aplicação disponível no Visual Studio 6 para desenvolvimento na plataforma Windows. Desta maneira a biblioteca disponibiliza muitos componentes necessários para criar e gerenciar uma aplicação.

A MFC trabalha com o conceito de documentos e visões. Neste modelo, um documento representa os dados com que o usuário interage e edita. Ele é criado pelo comando Novo ou Abrir e é tipicamente salvo em um arquivo. Já uma visão é uma janela onde o usuário interage com o documento.

No protótipo implementado, considerou-se como dados do usuário as definições das análises feitas pelo mesmo, em relação a um produto. As configurações para cada tipo de análise também são consideradas como documento. Por exemplo, para o tipo de análise realizado pela classe CSubQuadroForma, são armazenados a arquitetura da rede neural utilizada, assim como os pesos das conexões sinápticas de cada neurônio. Foi adicionado como membro da classe CDocument, definida pela framework da aplicação, uma variável membro do tipo CQuadro, sendo que ela centraliza todos os dados citados.

Já na visão do documento no protótipo, são exibidas as imagens a serem inspecionadas e outros detalhes relativos à análise.

3.3.3 OPERACIONALIDADE DA IMPLEMENTAÇÃO

Nesta seção será apresentada a operacionalidade da implementação, através do desenvolvimento do estudo de caso Configurar Análise, seguido do estudo de caso Inspecionar.

A tela inicial do protótipo é exibida na figura 29. O primeiro passo necessário para a sua utilização na inspeção de produtos é a definição e configuração das análises a serem consideradas na inspeção. Para isso, deve ser acessada a caixa de diálogo de definição de análises, através do botão Configurar.

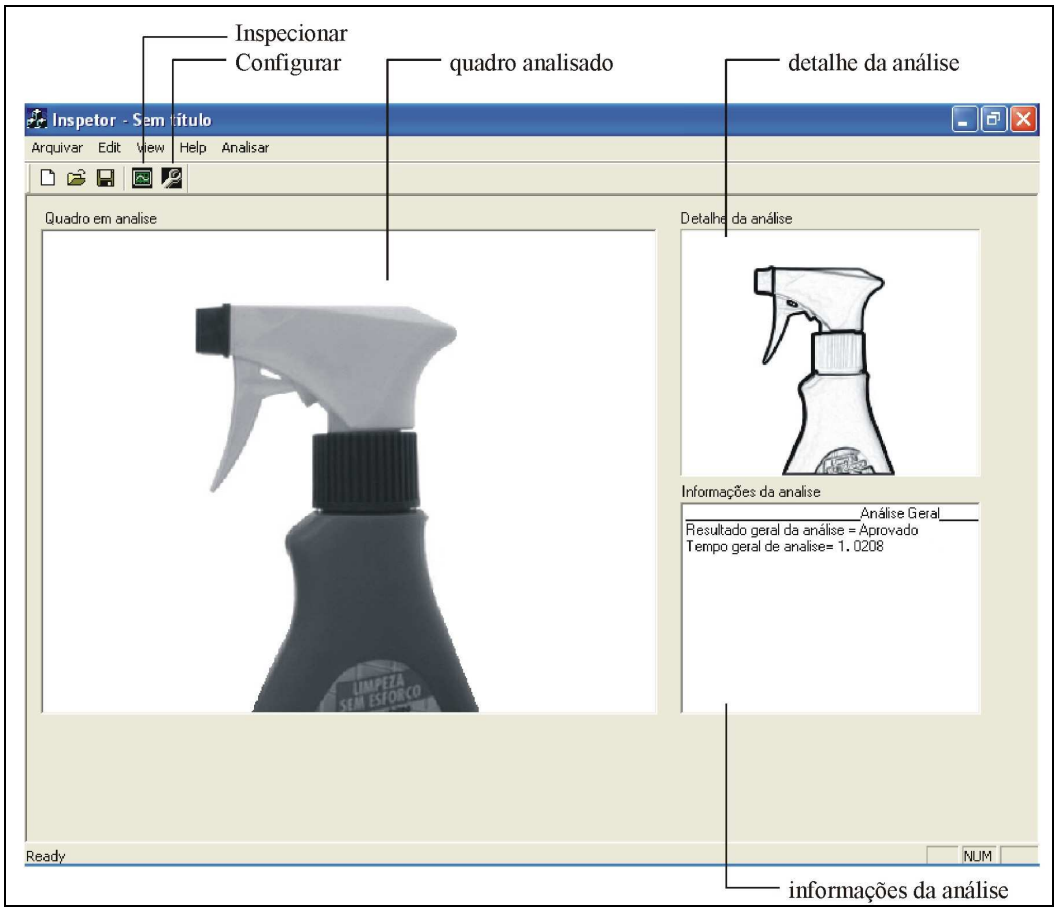

Figura 29 – Tela principal do protótipo

A tela exibida na figura 30 é utilizada na definição dos subquadros de análise. O usuário deve pressionar o botão da análise desejada, e então definir a área de atuação da mesma. Para isto basta arrastar uma marquise sobre a área da imagem a ser analisada.

Figura 30 – Tela de definição de subquadros

Definida a área de atuação do subquadro de análise, o usuário deve editar as variáveis pertinentes à mesma. Para isso, deve ser pressionado o botão editar.

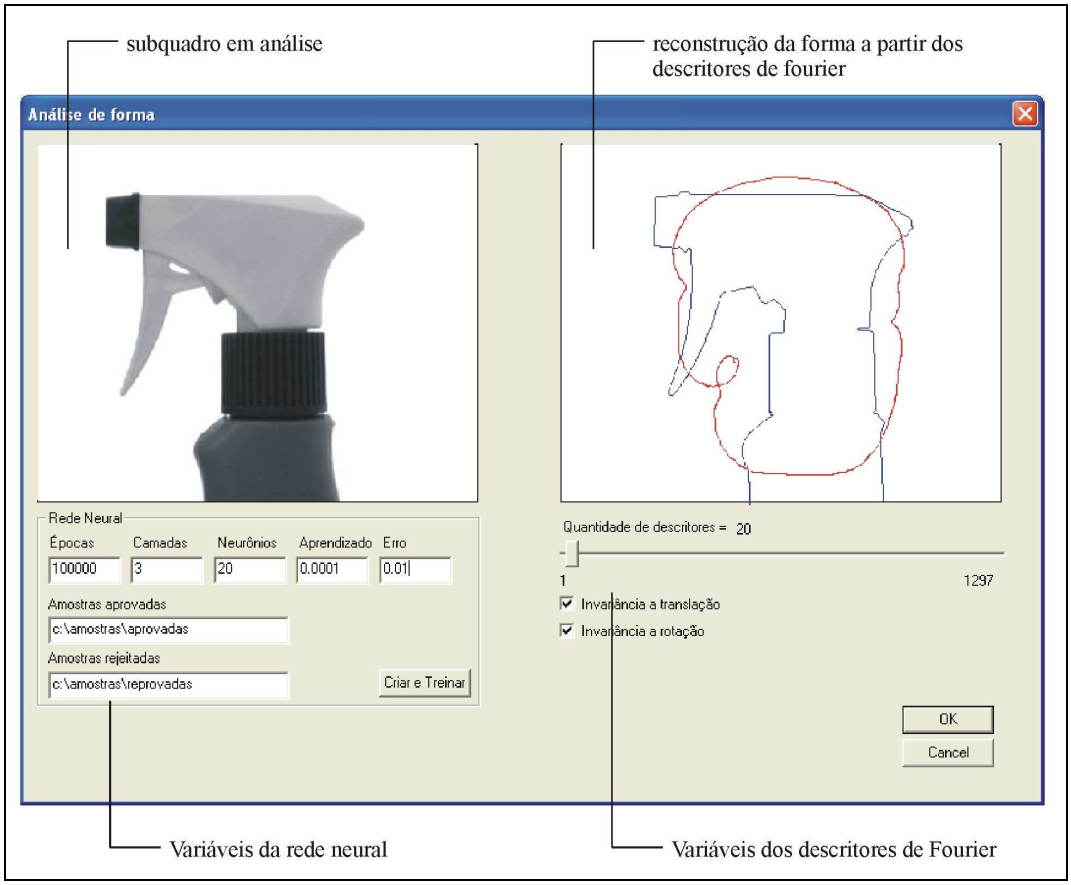

Figura 31 – Tela de definição de subquadros

A tela exibida na figura 31 é utilizada para configuração da análise de forma. Quanto à rede neural, devem ser informados os parâmetros utilizados na criação da mesma, assim como o local onde se encontram os padrões a serem utilizados no treinamento da rede. Quanto à descrição de forma, devem ser informadas a quantidade de descritores a serem utilizados, assim como as invariâncias a serem observadas. Definidos estes parâmetros, deve ser pressionado o botão Criar e Treinar para que a análise seja preparada para uso.

Após a configuração da análise, o protótipo encontra-se pronto para inspecionar imagens de produtos. Para isso, a partir da janela principal do protótipo, basta carregar a imagem através do menu, e pressionar o botão Inspecionar. Os resultados obtidos pela análise serão exibidos na lista de saída.

3.4 RESULTADOS E DISCUSSÃO

Dois casos de teste foram utilizados para verificar o cumprimento dos requisitos para o protótipo proposto. Cada caso de teste refere-se à inspeção de um tipo de produto, contendo diferentes particularidades a serem analisadas, para a distinção entre as classes de produtos aprovados e reprovados.

Para a captura das imagens dos produtos a serem utilizadas nos testes, contou-se com um sistema completo de aquisição de imagens, disponibilizado pela empresa Artvision. Este sistema é composto por câmera de vídeo, placa de aquisição, lentes, dispositivos de iluminação e um computador. O tamanho da imagem capturada pela placa de aquisição é de 640 por 480 *pixels*. Na figura 32 são exibidas as câmeras e o componente de iluminação de fundo. A iluminação de fundo é obtida pelo uso de uma lâmpada, sobreposta por uma peça de acrílico branca. Esta técnica é também utilizada na inspeção manual nas indústrias, onde, para produtos com orientação vertical, um painel luminoso é instalado no lado oposto do produto, em relação ao inspetor. Para produtos que permaneçam na horizontal, pode ser utilizada uma esteira teflonada, juntamente com uma iluminação inferior.

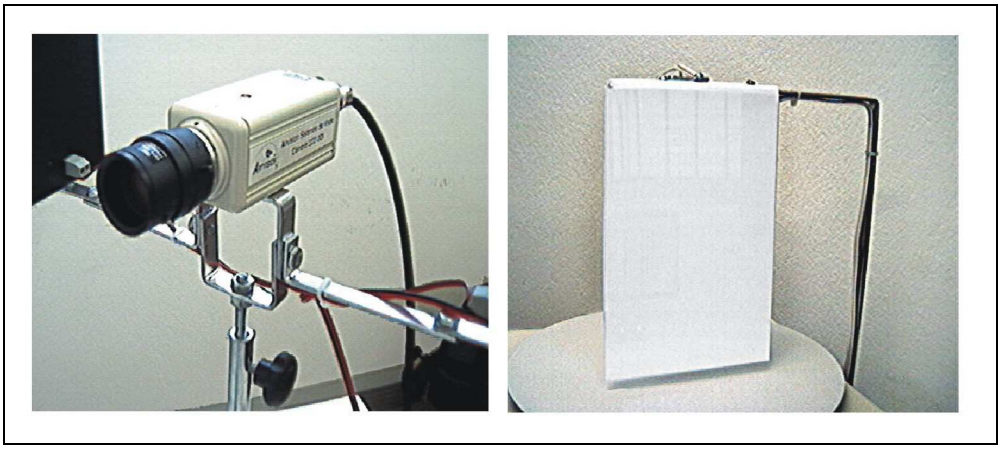

Figura 32 – Câmera e painel de iluminação

Diferentes configurações foram testadas, para determinar a melhor maneira de capturar as imagens dos produtos. Para os produtos escolhidos para serem utilizados como objeto de análise, a iluminação de fundo mostrou-se mais eficiente para o destaque das características do produto desejadas.

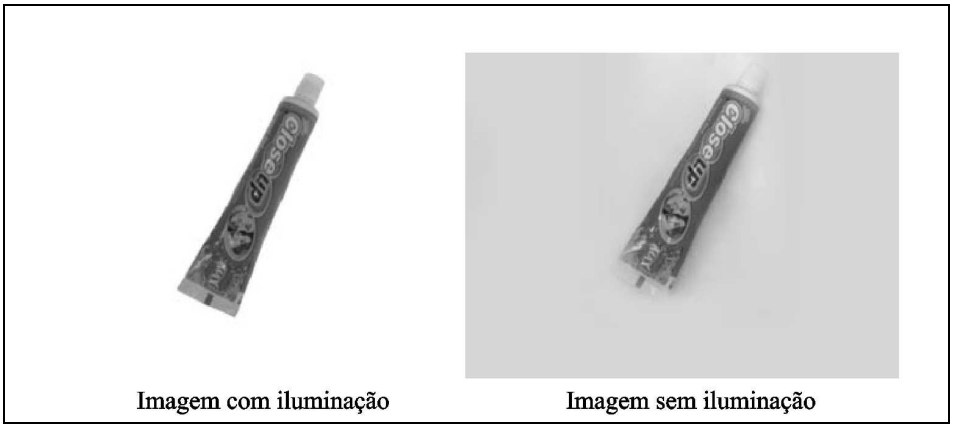

Figura 33 – Imagens capturadas com diferente iluminação

Na figura 33 podem ser vistas uma imagem capturada utilizando-se a luz de fundo, e outra sem o uso desta técnica de iluminação. Com o uso da iluminação de fundo obtiveram-se imagens com as bordas dos produtos bem definidas, ausência de reflexos e de sombras.

Definidas as variáveis de captura, as imagens foram adquiridas e armazenadas em diretórios, separadas pela respectiva classe de produto para posterior uso pelo protótipo.

No primeiro caso de teste, foi utilizado como objeto de análise um frasco empregado como embalagem de produtos para limpeza. Esta embalagem foi escolhida por ser composta por várias peças, cuja montagem deve ser inspecionada para a identificação de possíveis problemas.

Para este produto, foram definidos os seguintes itens como sendo potenciais pontos de defeito, e, portanto, de análise necessária na fase de inspeção:

a) presença e correta posição do bico;
b) presença e correta posição do gatilho;
c) alinhamento, presença e correta posição do sistema de borrifo.

A figura 34 mostra a imagem de um produto considerado aprovado.

Figura 34 – Amostra de produto aprovado

A figura 35 exibe imagens de produtos reprovados, exemplificando os defeitos citados anteriormente. Aqui não é considerada a situação em que o produto transita pela esteira e passe pela câmera com sua base em um ângulo diferente do exibido nas amostras da figura 34 e 35. Isto se dá pelo fato de que este controle pode ser conseguido fisicamente, pela utilização de duas guias, uma em cada lado do produto. Desta maneira pode-se reduzir em muito a quantidade de posições a serem consideradas pelo sistema de inspeção.

Outra observação a ser feita é que, sendo uma simulação de inspeção para a indústria química, não é realizada inspeção de forma sobre a parte inferior do frasco. Isto por que esta análise deve ser realizada ainda na indústria plástica, após o processo de injeção de plástico para a moldagem do frasco.

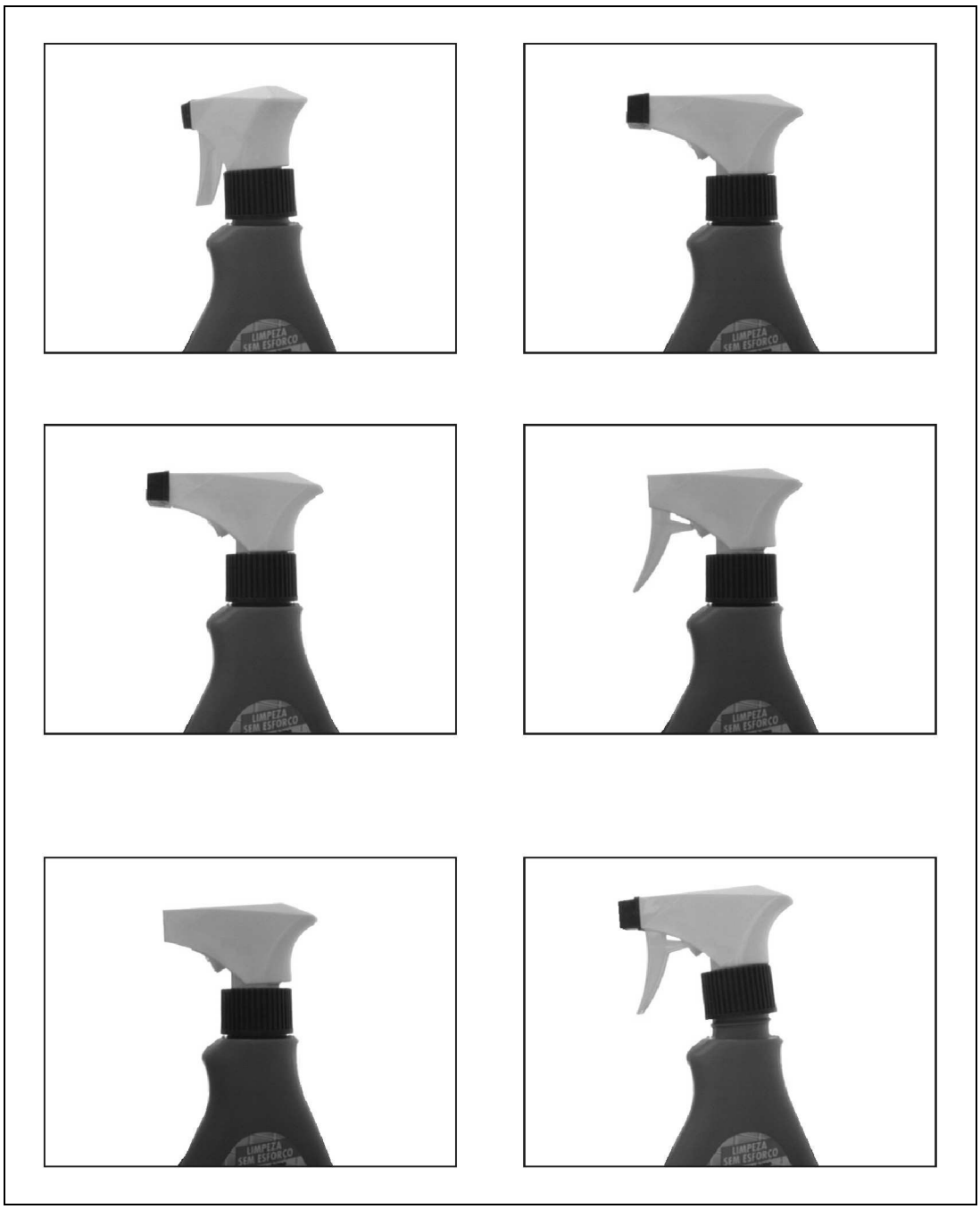

Figura 35 – Amostras de produtos reprovados

Para a descrição do produto foram utilizados 15 descritores de Fourier. Quanto às variações possíveis na estrutura da rede neural implementada, foi utilizada uma camada oculta, contendo 15 neurônios. Este mesmo número de neurônios foi utilizado na camada de entrada. Como a saída deve assumir um entre dois estados possíveis, foi utilizado na camada de saída apenas 1 neurônio.

Para este primeiro caso de teste, foi seguida a recomendação de Tafner (1996, p. 99), onde o conjunto de fatos apresentados à rede no momento do treinamento deve ser, no

mínimo, igual ao dobro do número de conexões entre os neurônios. Considerando que foram utilizadas na camada de entrada e na camada oculta 15 neurônios cada, tem-se aí um total de 225 conexões. Somando a este número valor igual à quantidade de conexões existentes entre a camada oculta e a camada de saída, obtem-se um total de 240 conexões. Calculando-se o dobro, a recomendação para este caso é de 480 fatos.

Foram utilizadas então 480 amostras para a fase de treinamento, dentre produtos aprovados e reprovados. Para a formação deste conjunto de treinamento, foram capturadas 50 imagens de produtos através do sistema de captura. O restante das amostras foi gerado a partir do processamento em lote de diferentes transformações geométricas, a partir das imagens originais. O objetivo da aplicação destas transformações foi o de gerar pequenas variações na forma do produto analisado. Na prática, estas variações podem ser resultado de variação no processo produtivo ou ainda alterações de configuração nos dispositivos de captura. Para realizar este processamento, foi utilizado o programa de edição de imagens denominado Photoshop CS (ADOBE SYSTEMS INCORPORATED, 2004).

O processo de treinamento mostrou-se demorado, assim como observado por Oliveira e Bauchspiess (2001) em seu trabalho. Para treinar a rede com a configuração citada anteriormente, o processo de treinamento para o conjunto criado foi de aproximadamente 3 horas. Foi utilizado um computador IBM-PC equipado com processador Pentium 4 de 2.5GHz e 256MB de memória.

Treinada a rede, foi iniciado o processo de reconhecimento. Nesta fase, 200 novas amostras de produtos aprovados e de produtos reprovados foram apresentadas à rede. Novamente foi utilizado o recurso de gerar variações de imagens a partir de 50 originais, através de transformações geométricas. A rede reconheceu 200 delas, resultando assim numa taxa de acerto de 100% na classificação dos produtos. Observe que as imagens exibidas à rede não haviam sido utilizadas no processo de treinamento. Este fato mostra a capacidade de generalização da rede.

A afirmação de Oliveira e Bauchspiess (2001), quando defendiam que o uso entre 10 e 25 coeficientes para a descrição da borda são suficientes foi confirmada, já que a partir dos 15 coeficientes utilizados, a rede neural conseguiu determinar uma função de decisão satisfatória no processo de treinamento.

Quanto ao requisito de velocidade em relação às análises, os resultados obtidos foram considerados satisfatórios. Para este caso de teste, o tempo necessário para a análise de cada produto girou em torno de 0,1 segundos. Deste tempo, verificou-se que a maior parcela é utilizada pelos algoritmos de processamento de imagem, sendo o restante consumido pelo reconhecimento através da rede neural. Quanto a este requisito, é importante comentar que não foram despendidos maiores esforços para a otimização do código. Portanto, o tempo gasto pela análise pode ainda ser reduzido.

Diante dos resultados positivos obtidos pela realização do caso de teste anterior, optou-se por criar um novo caso de teste a fim de validar o protótipo sobre outra condição de uso. Neste caso de teste, foi selecionado como objeto de análise um tubo utilizado como embalagem para creme dental. Esta embalagem foi escolhida por possibilitar a verificação da capacidade do protótipo de analisar produtos com variação na orientação em relação à câmera.

Para este produto, foram definidos os seguintes itens como sendo potenciais pontos de defeito, e, portanto, de análise necessária na fase de inspeção:

a) presença e correta posição da tampa;
b) integridade do tubo.

A figura 36 mostra a imagem de um produto considerado aprovado.

Figura 36 – Amostra de produto aprovado

A figura 37 exibe imagens de produtos reprovados, exemplificando os defeitos citados anteriormente. É importante observar que neste momento são consideradas todas as variações possíveis quanto à orientação do tubo.

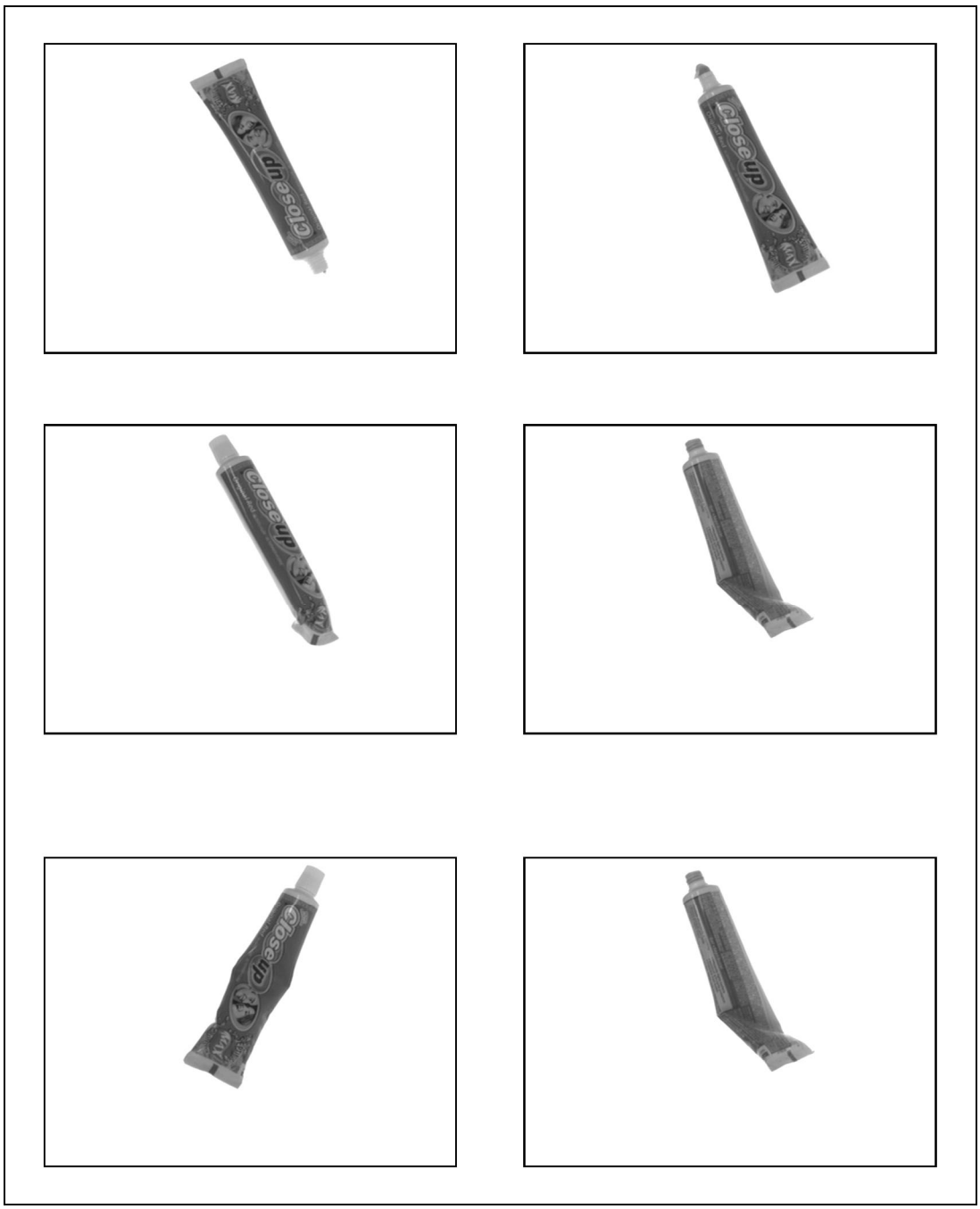

Figura 37 – Amostras de produtos reprovados

Para este caso de teste foi utilizada a mesma quantidade de descritores utilizados no caso anterior para a representação da borda, assim como a mesma estrutura para a rede neural. O tamanho do conjunto de treinamento, e o seu processo de geração, seguiram o processo realizado no caso de teste anterior.

Para este caso, o processo de treinamento mostrou-se ainda mais demorado, sendo que somente foi alcançado sucesso depois de 5 horas de treinamento e do aumento do valor de tolerância em relação à taxa de erro da rede. Enquanto para o primeiro caso de teste foi obtido

sucesso no treinamento da rede com um erro máximo de 0.1, para este caso foi necessário um erro máximo de 0.2. Observou-se que isto foi necessário pelo motivo de a diferença da borda entre os produtos aprovados e reprovados utilizados no primeiro caso de teste ser mais gritante do que a diferença encontrada nos produtos do deste caso.

Os resultados positivos obtidos no primeiro caso de uso, quanto à taxa de 100% de acerto no reconhecimento dos produtos, a quantidade de descritores necessária para descrição da borda e quanto ao tempo gasto para cada análise, se repetiram neste caso.

Adicionalmente, confirmou-se neste caso a eficiência no uso dos descritores quanto à invariância a translação e rotação, já que o protótipo reconheceu produtos em diferentes ângulos, sendo que na fase de treinamento não foram apresentados padrões abrangendo todas as combinações possíveis.

4 CONCLUSÕES

Este trabalho apresentou um método para a análise de produtos e o desenvolvimento de um protótipo de software para inspeção industrial automatizada. Para realizar tal tarefa foram utilizadas diversas técnicas de processamento de imagens, além de uma rede neural para o reconhecimento e interpretação.

A escolha das técnicas empregadas tanto na extração de informação das imagens quanto na interpretação das mesmas mostrou-se acertada. A combinação do método de descrição de fronteiras por Fourier e da técnica de interpretação por redes neurais permitiu o desenvolvimento de um protótipo de inspeção automatizada versátil quanto ao produto a ser inspecionado, com velocidade e com resultado satisfatório na execução da análise.

A utilização dos descritores de Fourier para a descrição de formas foi baseada na capacidade do método em representar as formas a partir de uma quantidade pequena de descritores, sendo esta característica muito útil à etapa de interpretação. Outra propriedade importante considerada no momento da escolha do método foi a possibilidade de obtenção de descritores invariantes à translação e rotação. Isso amplia as possibilidades de aplicação do protótipo a uma série de produtos em que existe variação na orientação.

Já a utilização de redes neurais do tipo Perceptron Multicamadas baseou-se na capacidade de generalização da rede e também na sua facilidade de implementação. Como a descrição dos produtos pode ser obtida por um número pequeno de descritores, é possível utilizar uma rede com quantidade reduzida de neurônios, tornando-a muito eficiente na classificação dos produtos. Apesar disto, o treinamento mostrou-se relativamente lento, principalmente para os casos em que a diferença do contorno entre os produtos da classe aprovado e reprovado não é gritante.

Finalmente, é importante comentar que, apesar do tipo de análise baseado na forma ser aplicável a uma grande diversidade de produtos, para muitos destes o mesmo não será suficiente para a determinação de ausência de todos os defeitos de produção possíveis. Muitos produtos demandam análises específicas, como, por exemplo, uma garrafa de bebida onde é necessária a análise do volume do líquido. Para este tipo de análise, o método apresentado não é recomendado, pois a variação no volume do líquido não tem reflexo sobre a forma da garrafa.

4.1 EXTENSÕES

A complexidade e abrangência dos sistemas de visão computacional possibilitam que o presente trabalho seja estendido com diferentes objetivos. Durante o desenvolvimento do protótipo este item foi levado em consideração, tanto no projeto do sistema como na escolha das ferramentas utilizadas na implementação.

Primeiramente, pode-se criar, partir do protótipo desenvolvido, um *framework* de aplicação voltado para sistemas de software para inspeção industrial. Para isto outros tipos de análises genéricas podem ser implementados, sendo implementados em classes estendidas a partir da classe CsubQuadro, definida com este propósito. Estas análises poderiam realizar verificações pertinentes às dimensões dos produtos e também verificações estatísticas, dentre outras.

Por outro lado, o protótipo pode ser utilizado como ponto de partida para a implementação de um sistema de software para inspeção, específico a um determinado produto. Neste caso seriam implementadas as análises fortemente dependentes da aplicação.

No protótipo implementado não foram considerados os sistemas externos de captura e de descarte. Podem ser desenvolvidos trabalhos com este fim, objetivando contemplar todos os componentes de um sistema de inspeção automatizada. A biblioteca de manipulação de imagens oferece subsídios para a implementação do sistema de captura, sendo que isto foi levado em conta no momento da escolha da biblioteca.

REFERÊNCIAS BIBLIOGRÁFICAS

ADOBE SYSTEMS INCORPORATED: Adobe Photoshop CS. [S.l.], 2004. Disponível em: <http://www.brasil.adobe.com/products/photoshop/newfeatures.html>. Acesso em: 20 novembro 2004.

BOWMAN, Elisabeth T.; Soga, Kenichi; Drummond, Tom W. Particle shape characterization using fourier analysis. **Geotechnique**, Cambridge, v. 51, n. 6, p. 545-554, 2001.

CARDOSO, Caíque. **UML na prática:** do problema ao sistema. 1. ed. Rio de Janeiro: Editora Ciência Moderna, 2003.

CÉSAR JR, Roberto M.; COSTA, Luciano F. **Shape analysis and classification.** 1. ed. Boca Raton: CRC Press, 2001.

FACON, Jacques. **Processamento e análise de imagens.** 1. ed. Embalse: EBAI, 1993.

FURLAN, José D. **Modelagem de objetos através da UML.** 1. ed. Makron Books: São Paulo, 1998.

GONZALES, Rafael C.; WOODS, Richard E. **Processamento de imagens digitais.** 1. ed. São Paulo: Edgard Blucher, 2000.

IBM CORPORATION: Rational Rose. [S.l.], 2004. Disponível em: <http://www-306.ibm.com/software/awdtools/developer/datamodeler>. Acesso em: 20 novembro 2004.

LOESCH, Cláudio; SARI, Solange T. **Redes neurais artificiais:** fundamentos e modelos. 1. ed. Blumenau: Editora da Furb, 1996.

MATHWORKS INTERNATIONAL: Matlab and simulink for technical computing. [S.l.], 2004. Disponível em: <http://www.mathworks.com>. Acesso em: 20 novembro 2004.

MEDEIROS, Luciano F. **Redes neurais em Delphi.** 1. ed. Florianópolis: Visual Books, 2003.

MICROSOFT CORPORATION. **Visual C++**. [S.l.], 2004. Disponível em: <http://www.microsoft.com/brasil/msdn/Tecnologias/visualcpp/default.mspx>. Acesso em: 20 novembro 2004.

MICROSOFT CORPORATION. **Vision SDK**. [S.l.], 2004. Disponível em: <http://www.research.microsoft.com/scripts/VisionSDK>. Acesso em: 29 agosto 2004.

OLIVEIRA, Kleyton C.; BAUCHSPIESS, Adolfo. Classificação de imagens codificadas por cadeias direcionais utilizando redes neurais artificiais. In: CONGRESSO BRASILEIRO DE REDES NEURAIS, 5., 2001, Rio de Janeiro. **Proceedings...** Rio de Janeiro: PUC, 2001. p. 73-79.

TAFNER, Malcon A. **Redes neurais artificiais:** introdução e princípios de neurocomputação. 1. ed. Blumenau: Editora da Furb, 1996.