CONTROLE DIMENSIONAL ATRAVÉS DE TÉCNICAS DE PROCESSAMENTO DE IMAGENS

Alexandre Gaspary Haupt[1,2]
alexandre.haupt@unilasalle.edu.br
Edison Pereira Dachi[1]
edison.dachi@senairs.org.br
Felipe Ludwig Piovezan[1]
f_piovezan@hotmail.com

1 Faculdade de Tecnologia SENAI Porto Alegre. Porto Alegre, RS
2 Centro Universitário La Salle. Canoas, RS

RESUMO

Atualmente, empresas se dedicam a melhorar a qualidade de seus produtos a fim de ganhar espaço no mercado e conquistar novos consumidores. Processos manuais de medição na indústria têm sido falhos, deixando chegar ao consumidor peças fora de especificação. Este trabalho aborda técnicas de processamento de imagem para inspecionar peças de modo a reduzir erros de medição, monitorando-as e mantendo-as dentro das especificações de engenharia. São analisados resultados obtidos com imagens artificiais e reais. A realização dos testes em imagens reais de tamanho 640x480 pixels, utilizando a técnica proposta, mostra um erro médio de 0,32% para o comprimento e de 2,15% para a largura. O maior erro é obtido quando a dimensão é menor do que 100 pixels e o objeto de interesse possui tonalidade próxima a cor de fundo. A técnica proposta mostra um baixo erro comparando as medidas reais com as obtidas a partir do processamento de imagens, tornando possível sua aplicação nas linhas de produção industrial com vistas à automação do controle de qualidade.

Palavras Chave: Processamento de Imagens, automação de medidas, inspeção visual.

ABSTRACT

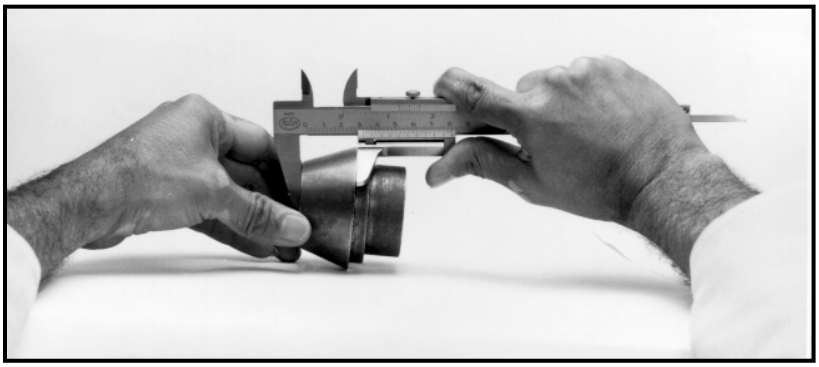
Nowadays, companies are dedicated to improving the quality of their products to win space in the market and attract new consumers. In this sense, techniques to automate the quality control have been explored to reduce the number of pieces out of specification that arrives to the consumer. This work approaches techniques to inspect pieces, through the image processing. Results are analyzed obtained with artificial and real images. The performance of the tests on real images of size 640x480 pixels showed that the average error is 0.32% to length and 2.15% to width. The largest error is obtained when the dimension to be measured is less than 100 pixels and has the nearest color between foreground and background. The proposed technique shows a low error compared with the actual measurements obtained from image processing, making possible their application in industrial production lines with a view to automating quality control.

Keywords: Image Processing, measures automation, visual Inspection.

1 INTRODUÇÃO

O foco em sistemas de inspeção visual engloba diversas áreas da indústria. Entre elas a indústria de alimentos, de celulose, montadoras de automóveis, farmacêutica, de componentes eletrônicos. A constante busca por melhoria da qualidade dentro das empresas possibilitou que sistemas de visão ganhassem espaço e credibilidade. No controle de qualidade de produtos industriais é comum o uso de pessoas para inspecionarem produtos nas linhas de produção. Processos de medidas manuais tendem a ser repetitivos, reduzindo ao longo do tempo a concentração do inspetor. Além disso, trabalhos repetitivos reduzem a produtividade devido ao cansaço do operador. Neste sentido, técnicas de processamento de imagem surgem como uma alternativa viável para solução deste problema.
A figura 1 ilustra um processo de medida manual com a utilização de um paquímetro. Processos de medida manual tendem a serem lentos e sujeitos a erros de operador. Peças com medidas erradas sendo liberadas pela empresa podem gerar devoluções de produtos, recall, insatisfação de clientes, redução de credibilidade na marca da empresa, redução de produtividade e lucratividade.

Figura 1 - Inspeção manual nas medidas de uma peça utilizando um paquímetro quadrimensional.



.
Fonte: ebah, 2013.

Além disso, medidas manuais, utilizando instrumentos tais como o paquímetro, dependem da sensibilidade e experiência do operador em determinar a pressão exercida sobre o instrumento. Existindo, assim, um erro inerente à medida utilizando-se este instrumento, seja ele digital ou analógico.
Para tentar suprir a necessidade de utilizar mão de obra humana, poderiam ser investidas câmeras para processamento de imagens de peças ou produtos para diversas finalidades, como por exemplo, detecção de borda, furos, cores, medições, contagem de características da peça, identificação de caracteres alfanuméricos, leitura de código de barras e outras possibilidades.
O uso de processamentos de imagens digitais é a forma para extrair informações de interesse. As características retiradas da imagem são processadas em um programa para serem comparadas com a peça padrão. Esse processo é capaz de localizar falha em peças ou produtos que em certos momentos é imperceptível à visão humana. E também podem realizar históricos de falhas para controle de qualidade. Com isso pode-se conceituar peças como aprovadas ou reprovadas.

Este trabalho visa demonstrar o uso de inspeção visual através de métodos de processamento digital de imagens, analisando e processando imagens com técnicas para pré-processamento, segmentação e extração de informações de interesse da imagem. A inspeção automática das imagens tem por objetivo reduzir o erro de medição nas dimensões de uma peça. Neste trabalho são apresentadas funções para processamento de imagens, elaboradas em linguagem C, que visam extrair as medidas de imagens artificiais utilizadas para teste e imagens reais.

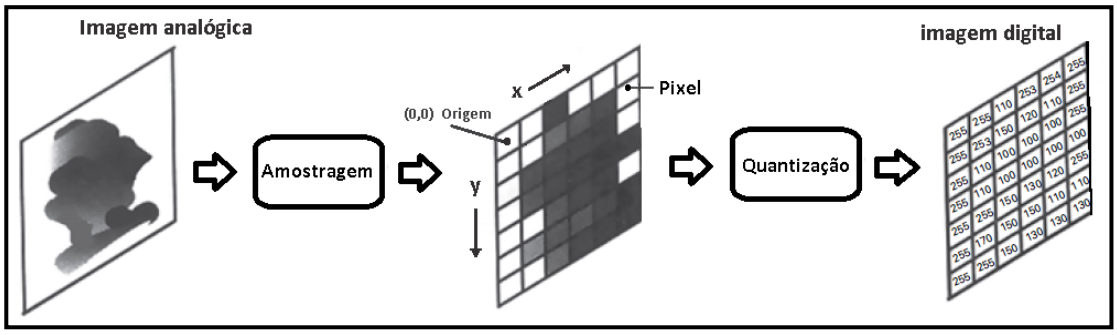
2 REVISÃO BIBLIOGRÁFICA

Para representação de uma imagem digital consideram-se coordenadas espaciais (x,y) e níveis de cinza, bem como alguns dos elementos básicos como pixel, resolução e modelos de cor. A representação de uma imagem é uma função f(x,y), onde x e y são coordenadas espaciais de coluna e linha, respectivamente e f(x,y) possui um valor em níveis de cinza em qualquer ponto da imagem. A origem (0,0) está situada no canto superior esquerda da imagem (GONZALEZ e WOODS, 2000).

Cada ponto da imagem é chamado de "pixels" ou "pels" (abreviação do inglês picture element). É também o menor ponto de uma imagem, com dimensões específicas de acordo com a amostragem. Um pixel pode receber um valor de 0 a 255, sendo o valor 0 para o preto e o valor 255 para o branco (GONZALEZ e WOODS, 2000). Ou seja, esse valor será a cor que representa um ponto da imagem. Sendo assim, o conjunto de pixels e seus respectivos tons formam a imagem.

A resolução de uma imagem é dada em função do produto de colunas em x por linhas em y. Como por exemplo, 640x480 pixels, é uma imagem que possui 640 amostras em pixel nas colunas em x e 480 amostras em pixel nas linhas em y. Atualmente nos monitores, as resoluções de imagem podem ir desde 800x600 pixels até 1280x800 ou mais. "É importante notar que uma imagem contendo um grande número de pixels não necessariamente possui resolução maior do que outra contendo menor número de pixels". (PEDRINI; SCHWARTZ, 2008, p.18). Quanto maior o número de amostras na direção y e x, menor será a dimensão de um pixel e, portanto melhor será a resolução da imagem. Os processos de amostragem de uma imagem analógica e sua quantização em tons de cinza são ilustrados na figura 2.

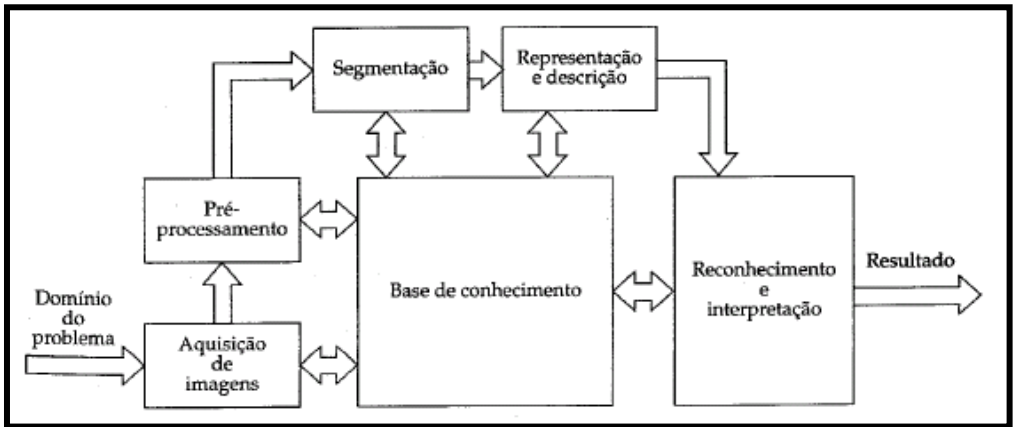
Figura 2 – Processos de amostragem e quantização de uma imagem.



Fonte: Unicamp, 2013.

A tarefa do processamento de imagem digital ocorre em várias etapas, desde o domínio do problema até o resultado (GONZALEZ, WOODS; 2000). Conforme a figura 3, pode-se observar no fluxograma os passos fundamentais.

Figura 3 – Etapas principais em processamento de imagens digitais.



Fonte: GONZALEZ, 2010, p. 5.

A etapa aquisição de imagens realiza a captura da imagem através de dispositivos como câmeras digitais e scanners que tem a função de capturar imagens. O pré-processamento tem a função de melhorar a qualidade da imagem com a aplicação de técnicas, pois na etapa de aquisição podem ocorrer alguns defeitos oriundos de ruídos e distorções de contraste e brilho. Na segmentação são abordadas as áreas de interesse da imagem. A representação serve para armazenar e manipular os objetos de interesse. A descrição é a extração de característica, provenientes desse objeto, resultando em informações quantitativas. Finalmente, passamos para a etapa de reconhecimento e interpretação. O reconhecimento serve para atribuir um identificador para o objeto de interesse. E a interpretação permite fornecer um significado para os objetos reconhecidos.

Quando olhamos para uma imagem, é possível fazer uma análise mais detalhada através cores. A importância de estabelecer padrões de cores é fundamental para processarmos e representarmos uma imagem. Segundo GONZALEZ (2010, pg. 160), “Essencialmente, um modelo de cor é uma especificação de um sistema de coordenadas tridimensionais e um subespaço dentro deste sistema onde cada cor é representada por um único ponto.” São modelos de cores conhecidos: RGB, CMY, YIQ, HSI e HSV. Dentre esses, destaca-se o modelo RGB (Red, Green, Blue) que representa as cores vermelho, verde e azul.

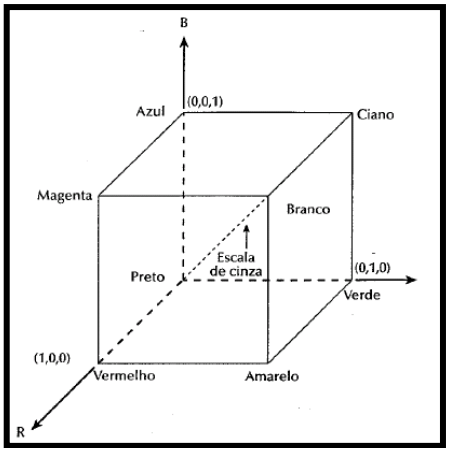
Esse modelo é encontrado principalmente em monitores e câmeras coloridas. CMY (Cyan, Magenta, Yellow) representa as cores ciano, magenta e amarelo. Esse padrão é usado em impressoras coloridas. O YIQ possui os elementos luminância, matiz, saturação, que representam respectivamente as siglas Y, I e Q. Ele também é usado em televisões em cores. O HSI representa a matiz, saturação e intensidade. Esse modelo é usado em manipulação de imagens. E finalmente, o HSV (Hue, Saturation, Value) que significa respectivamente matiz, saturação e valor ou brilho.

2.1 Cubo RGB

A figura 4 apresenta o modelo de cores RGB baseado em um sistema de coordenadas cartesianas, formando um cubo. Nesse modelo, estão normalizados os valores de vermelho, verde e azul em 1 ou 0. As cores primárias do modelo são o vermelho (1,0,0), o verde (0,1,0) e o azul (0,0,1). E as cores complementares são formadas pelo ciano (0,1,1), magenta (1,0,1)

e o amarelo (1,1,0). O ponto de origem (0,0,0) é correspondente ao preto e o ponto mais distante da origem é o branco, formado pelo ponto (1,1,1). Portanto o preto poderia ser dito como a ausência de todas as cores e o branco pela união de todas as cores. A diagonal formada do preto até o branco representa os tons de cinza ou escala de cinza, conforme mostra a figura 4.

Figura 4 - Cubo de cores RGB. Os pontos ao longo da diagonal principal tem
Valores de cinza desde o preto na origem até o branco no ponto (1,1,1).

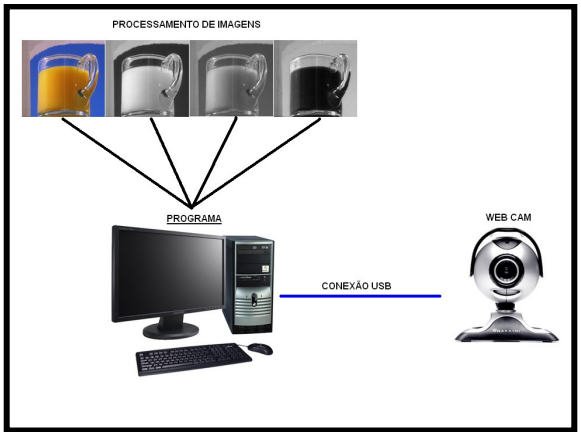


Fonte: (GONZALEZ, 2000, p. 161)

3 MATERIAIS E MÉTODOS

Neste capítulo abordam-se a forma como foram feitos os ensaios e os softwares e hardwares que foram utilizados no ambiente de testes. Os ensaios realizados foram feitos utilizando uma web cam, a qual realiza a aquisição de imagens e envia ao computador para processar a imagem. A figura 5 mostra um exemplo de topologia e com processamento de uma imagem RGB para escalas de cinza em tons de vermelho, verde e azul.

Figura 5 - Topologia do Ensaio.



Fonte: Autor

3.1 AQUISIÇÃO DE IMAGEM

A aquisição da imagem é feita por equipamentos como máquinas digitais, filmadoras, scanners, etc. Neste caso, utilizou-se uma web cam com as seguintes características.

- Comunicação: conexão USB.
- Resolução Máxima: 640 x 480 pixels (350K pixels).
- Tipo de sensor: VGA CMOS (300K).
- Alcance de Foco: 5 cm ao Infinito.
- Formatos de Vídeo: 24 bits, 1420 RGB.
- Compressão de imagem integrada
- Ajuste de Brilho Automático

As imagens capturadas pela web cam são enviadas ao equipamento de processamento.

3.2 PROCESSADOR

O processamento de imagens requer, em geral, altas velocidades para realizar tarefas complexas. Dependendo do software utilizado o processamento pode ser mais lento. Assim como depende também do algoritmo que está sendo executado pelo software. Contudo, o programa pode ser curto e não ser necessário o uso de um processador veloz. É recomendado sempre um processador com uma boa velocidade para o melhor desempenho do sistema. O processador utilizado nos ensaios foi o modelo Intel Core 2 Quad Q6600 com freqüência de 2,4GHz.

3.3 SOFTWARE

Atualmente existem no mercado alguns programas que trabalham com visão computacional ou processamento de imagens. No entanto, softwares dedicados têm custo elevado, mas possuem funções prontas que facilitam a operação. O software In-Sight Explorer Trial 4.1.0, utilizado neste trabalho, é mostrado na figura 6. Ele agrega muitas funções e possibilita gerar scripts de programação. Este software foi utilizado no trabalho com a finalidade de comparar os resultados obtidos em rotinas criadas em linguagem C com funções comerciais existentes no mercado.

A Biblioteca de Processamento de Imagens do LaPSI, ou simplesmente Lili (LaPSI Image Processing Library), foi desenvolvida pelo Laboratório de Processamento de Imagens da UFRGS com o objetivo de oferecer uma maneira eficiente para manipulação de imagens. Visando a portabilidade, a biblioteca foi escrita na linguagem ANSI/ISO C.
A importância que a biblioteca LILI possui é fornecer funções básicas em linguagem C para processamento de imagens e permitir que o usuário contribua com suas funções, por ser este um ambiente aberto.

3.4 REPRESENTAÇÃO DE IMAGENS NO FORMATO BITMAP

Imagens são definidas no formato bitmap através de uma matriz A x B pixels, onde cada elemento da matriz possui um valor de cor e luminosidade. É um dos formatos mais usados em processamento de imagens. Alguns softwares como, por exemplo, o C++Builder versão 3.0 utilizam apenas este formato.

A desvantagem do formato bitmap é o tamanho do arquivo, principalmente quando se trata de imagens de alta resolução. Já outros formatos possuem algoritmos de compactação da imagem, como o JPEG, ocupando menos espaço para armazenamento, a custas de uma qualidade inferior quando comparada com o formato bitmap. Uma imagem no formato bitmap de 3,00 Mb e de dimensões 1024x1024, poderia ser convertida, através de um editor de imagens em uma imagem JPEG de 78 Kb. Ou seja, o formato compactado é 38 vezes menor do que a imagem sem compactação.

3.5 DETECÇÃO DE BORDAS

"Uma borda é o limite entre duas regiões com propriedades relativamente distintas de nível de cinza." (GONZALEZ, 2000, p.297). Portanto, para que seja processado como borda deve haver uma diferença brusca entre os valores de cinza dos pixels vizinhos. Os detectores de borda são funções bastante utilizadas e importantes para a segmentação de imagens.
Existem técnicas de detecção de bordas. Dentre elas podemos citar Sobel, Prewitt, Laplaciano.
Processar objetos brancos em fundos claros dificulta a extração de informações da imagem, como ilustrado na figura 6. Porém, para melhorar a manipulação da imagem utilizamos detectores de borda como o sobel para que o objeto fique bem definido e possa fazer a extração das características. Imagens com ruído também podem prejudicar o processamento visto que podem ser interpretadas como bordas, dificultando assim a interpretação de outras bordas importantes.

Figura 6 - Exemplo de detecção de borda sobel.
a) imagem original. b) imagem em escala de cinza. c) operador de sobel.

(a) (b) (c)

Fonte: Autor

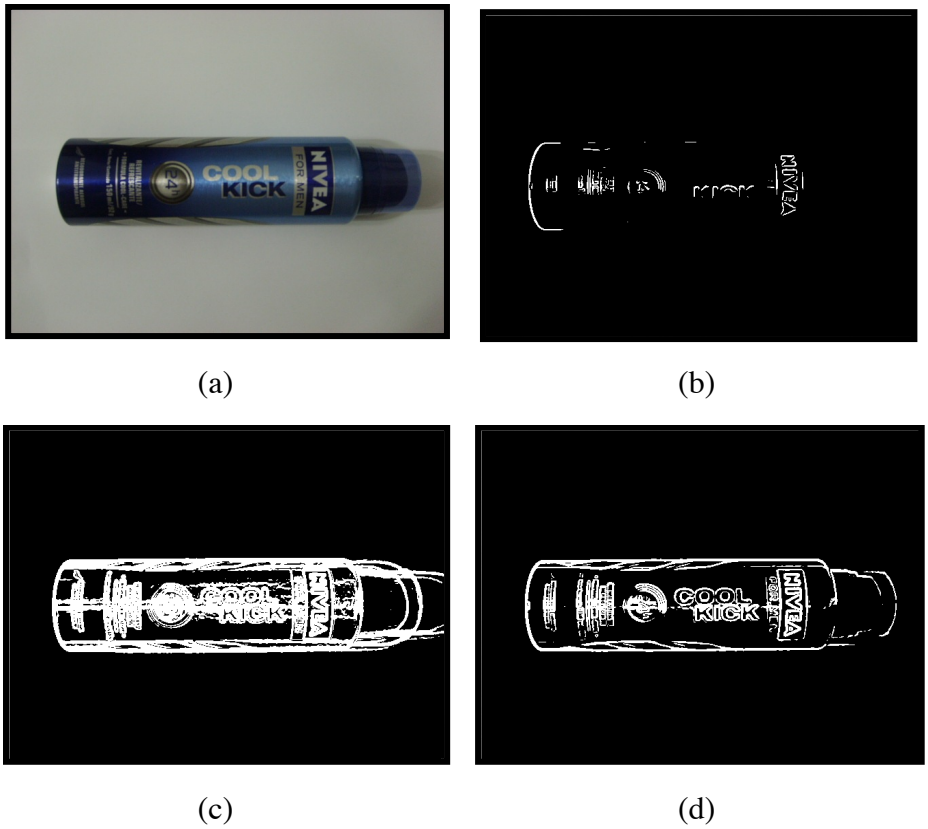
3.6 BINARIZAÇÃO

A binarização é uma etapa que faz parte da segmentação no processamento de imagens. Essa técnica é usada em imagens em tons de cinza. A principal tarefa é transformar uma imagem que está em tons de cinza em uma nova imagem em preto e branco. Isso é feito utilizando-se um valor, chamado de limiar de binarização. Pixel abaixo ou igual a este valor de limiar serão convertidos para preto (0) e pixels com valor acima do limiar serão convertidos para branco (255). Assim é gerada uma nova imagem g(x,y). Um limiar de g(x,y) pode ser definido conforme mostrado na equação 1.

$$g(x,y)=\begin{cases}0, \text{ se } f(x,y)\leq threshold\\ 255, \text{ se } f(x,y)>threshold.\end{cases} \quad (1)$$

O valor de limiar pode realçar ou reduzir detalhes importantes em uma imagem. Este valor pode ser definido empiricamente, ou através de algumas técnicas para estimar o valor ideal. Uma das técnicas utilizadas é a análise do histograma de níveis de cinza da imagem em tons de cinza. A figura 7 ilustra os resultados de diferentes valores de limiar em uma imagem.

Figura 7 - Exemplo de binarização.
a) imagem original. b) imagem binária com limiar = 80. c) imagem binária com limiar = 10. d) imagem binária com limiar = 30.



(a) (b)

(c) (d)

Fonte: Autor

3.7 CONVERSÃO RGB PARA ESCALA DE CINZA

A conversão de uma imagem em RGB para tons de cinza é uma das técnicas mais usadas quando se trata de processamento de imagens. A maioria dos métodos de segmentação e extração de informações trabalha com imagens em escala de cinza como variável de entrada. Essa conversão faz parte da etapa de pré-processamento. O procedimento para transformar um pixel RGB em tons de cinza é por meio de um cálculo simples.

Escala de Cinza = (R (vermelho) + G (verde) + B (azul)) / 3 (2)

Portanto, o resultado desse cálculo que vai de 0 a 255 está presente na diagonal do cubo RGB, conforme visto em 2.1. Observa-se que a escala de cinza é obtida pela média dos tons de vermelho, verde e azul. A função criada em linguagem C para realizar a conversão de RGB para uma imagem em tons de cinza é mostrada no apêndice B.

A função de conversão para grayscale recebe a imagem RGB por meio de um ponteiro. Dentro da função são lidos os valores de largura e altura da imagem para realizar os laços para varredura da imagem. O primeiro laço faz a varredura de linhas, enquanto o segundo varre as colunas. Portanto, o processamento é feito da esquerda para direita e de cima para baixo. Em seguida, é realizado o cálculo para transformar a imagem RGB em escala de cinza. As variáveis denominadas vetorR, vetorG e vetorB representam respectivamente a intensidade das cores vermelho, verde, e azul. O resultado da equação 2 é a média dos valores RGB que resulta em um pixel na escala de cinza. Ao final, quando todos os pixels da nova imagem estiverem processados, a imagem em escala de cinza estará pronta e atribuída à variável "imageGray".

3.8 EXTRAÇÃO DE INFORMAÇÕES DA IMAGEM

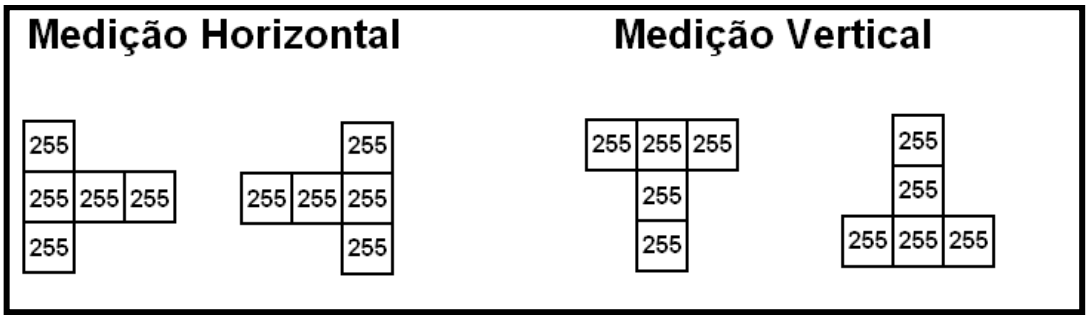
A verificação de produtos em sistemas de qualidade na indústria é muito comum. Uma das formas de garantir isso é a inspeção de medidas em peças. Além disso, é possível comprovar por meio de históricos os valores encontrados de cada peça. A forma como é feita uma medição através do programa de processamento de imagens é pela contagem de pixels de um objeto de interesse. O método de varredura de todos os pixels nesse projeto foi chamado de conexão T, que serão vistos mais detalhes no item 3.11. Podemos expressar uma medida em pixel bem como em centímetros, metros, quilômetros e sucessivamente. Para isso precisa-se conhecer a altura que foi capturada a imagem e uma medida de referência para que possa fazer a conversão de pixel para a medida de interesse. Lembrando que se a altura aumentar, o valor da medida em pixel diminui. O método para obter medidas como comprimento e largura é realizado com a função lcentroideB localizada na lili. Essa função trabalha com imagens binarizadas medindo-se somente o objeto em cor branca. Se o objeto em questão está em preto, há a necessidade de utilizar a função linverte para substituir o preto pelo branco e o inverso. Para obter medidas de diâmetro em peças circulares podemos usar a função lmedida que também se encontra na lili. A imagem também precisa ser binarizada e, se necessário, o uso de inversão de cores com linverte.
Através das médias da largura e comprimento de uma imagem retangular ou quadrada o programa consegue obter o ponto central do objeto. O centro é formado pelas variáveis pxc e pyc. A função lmedida é apresentada no apêndice A. Essa função, elaborada em linguagem C, é capaz de medir o comprimento máximo e largura máxima e o ponto central do objeto na imagem.

3.9 TESTE DE CONEXÃO DE PIXELS

Para evitar que pixels espúrios sejam detectados como parte da borda o objeto, alterando as medições, analisa cada pixel verificando se este pertence à borda do objeto. Se o pixel em avaliação não possuir vizinhos será rejeitado. Após testes com diferentes tipos de vizinhança, observou-se que os melhores resultados foram obtidos com a conexão T. Neste tipo de conexão é necessário que o pixel esteja ligado a outros pixels em forma de T, conforme

ilustrado na figura 8. Como o objeto a ser analisado é branco, a conexão T precisa identificar em forma de T os valores de 255 que correspondem ao branco.

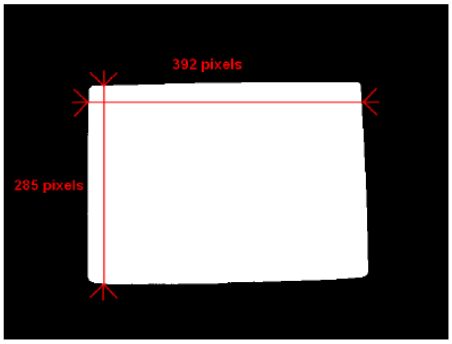
Figura 8 - Conexão T para medição na horizontal e vertical.



Fonte: Autor

Para o programa realizar a medição na horizontal, a varredura dos pixels deve ser feita da esquerda para a direita e de cima para baixo. Isso possibilita ao programa encontrar o valor da largura. E para realizar a medição na vertical, a varredura dos pixels é feita de cima para baixo e da esquerda para a direita, procurando por pixels brancos conectados. Assim, o valor encontrado na vertical será o comprimento. A figura 9 ilustra o tipo de imagem utilizado para medição.

Figura 9 - Imagem invertida e binarizada para medição.



Fonte: Autor

3.10 FILTRAGEM DE IMAGENS

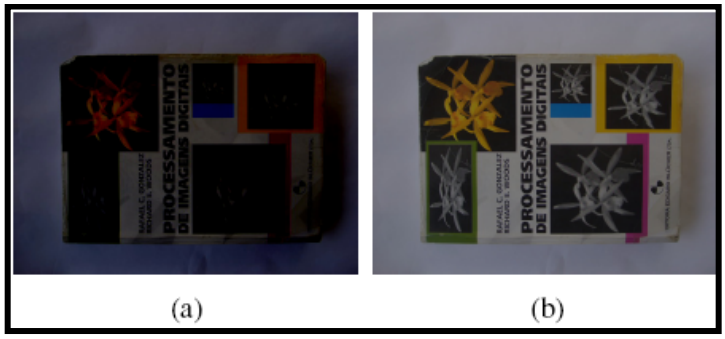
Graças à tecnologia presente na computação gráfica, hoje podemos melhorar a qualidade de imagens por meio de processos de restauração. O objetivo é sempre deixar a imagem o mais real possível ou que facilite o processamento em regiões de interesse. Os principais problemas de degradação de imagens são os ruídos provocados por equipamentos, os borrões que surgem quando um objeto está em movimento, perda de foco e até excesso ou falta de iluminação. Segundo Pedrini e Schwartz (2008), o processo de filtragem do ruído que foi gerado em imagens através de processos de aquisição, transmissão ou processamento, pode degradar partes importantes do objeto. Os tipos de ruídos mais conhecidos são o ruído uniforme, o impulsivo e o gaussiano. A figura 10 ilustra o ruído sal-e-pimenta (salt-and-pepper), que é exemplificado por pontos pretos e brancos sobre a imagem.

Figura 10 - Exemplo de ruído sal-e-pimenta.

Fonte: Autor

Quando se define o tratamento de uma imagem deve-se levar em consideração que o objeto de interesse não fique degradado, prejudicando assim na extração de informações da imagem. A iluminação é um dos fatores que mais prejudicam uma imagem. Quando ela for escura, pode-se clarear a imagem aumentando-se o valor de tonalidade de cor de cada pixel desta imagem gerando uma nova imagem mais clara. Todavia, se for necessário escurecer mais a imagem faz-se o inverso do processo descrito. Ao invés de aumentar o valor do pixel da imagem antiga, diminui-se esse valor para escurecer. A figura 11 ilustra uma imagem original (11.a) e a imagem clareada (11.b).

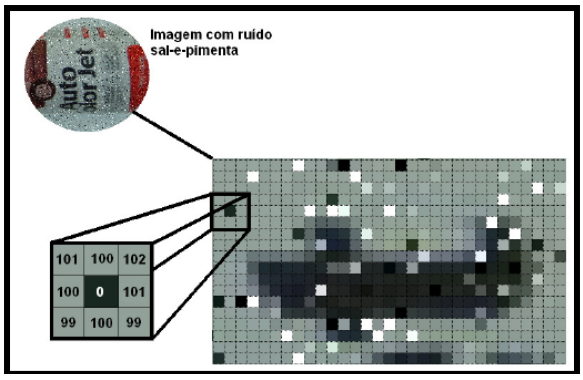
Figura 11 - Clareamento de Imagem.
a) Imagem original. b) Imagem com clareamento.



Fonte: Autor

Imagens com ruídos podem ser tratadas através do filtro da mediana. A figura 12 ilustra uma imagem ruidosa onde se aplica o filtro da mediana.

Figura 12 – Imagem ruidosa com aplicação do filtro mediana



Fonte: Autor

O filtro da mediana é um dos melhores métodos para retirar os ruídos ou suavizar as imagens. Esta função identifica e elimina pixels que tem o valor no limiar do branco (255) ou do preto (0). De acordo com Gonzalez (2009), "a principal função dos filtros de mediana é forçar pontos com níveis de intensidade distintos para serem mais semelhantes aos seus vizinhos". A figura 12 mostra um pixel preto identificado como ruído (valor 0). Na filtragem os valores dos pixels de vizinhança 8 são considerados e organizados em forma de vetor.

[101, 100, 102, 100, 0, 101, 99, 100, 99]

Logo após é organizado o vetor em ordem crescente.

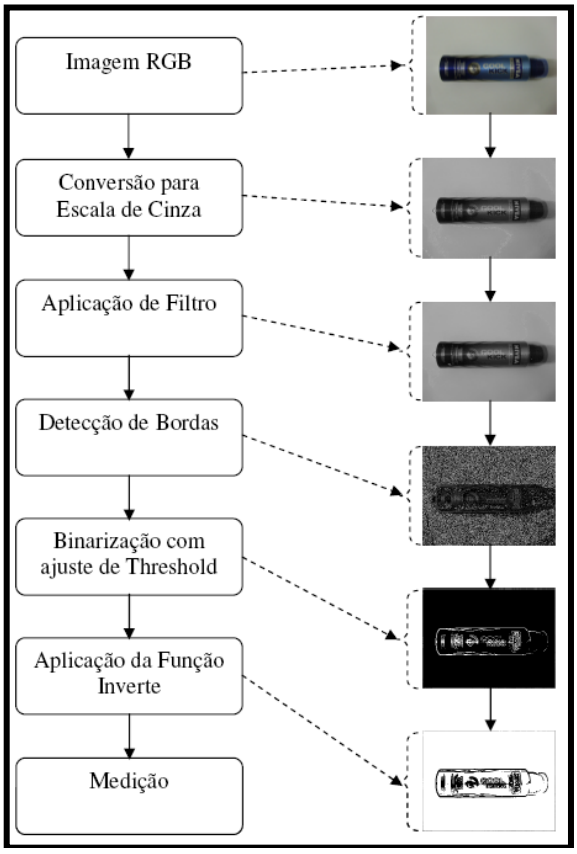
[0, 99, 99, 100, 100, 100, 101, 101, 102]

O valor dessa filtragem é o número central localizado na quinta coluna, cujo valor é 100. Portanto o pixel com ruído será substituído pelo pixel de valor mediano.

4 RESULTADOS

Foram testadas imagens artificiais, construídas no software Paint.net versão 3.5.5, e imagens reais de peças e produtos a serem medidas através do programa de processamento de imagens. Para o processamento de cada imagem utilizou-se o fluxograma apresentado na figura 13.

Figura 13 - Fluxograma utilizado no processamento das imagens.

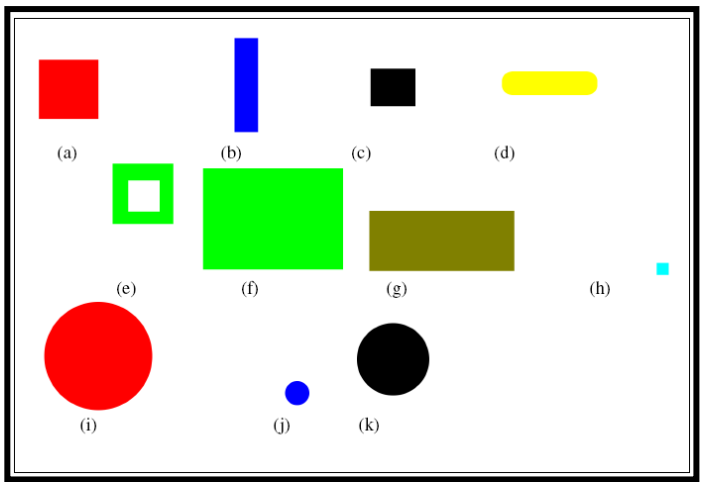


Fonte: Autor

4.1 TESTE COM IMAGENS ARTIFICIAIS

As imagens artificiais, criadas pelo software Paint.NET, foram processadas na seguinte ordem: conversão RGB para escala de cinza, filtros de suavização, binarização e as medições de comprimento e largura. Na figura 14 são apresentadas imagens artificiais utilizadas nos testes do algoritmo.

Figura 14 - Imagens de prova artificiais com tamanhos 640x480.



Fonte: Autor

Na tabela 1, são comparadas as medidas reais e as obtidas com auxílio do programa escrito usando o software C++Builder. Usaram-se imagens de diversos tamanhos e cores com a finalidade de medir o comprimento e a largura de cada amostra e seus respectivos erros.

Tabela 1 - Medidas obtidas nos testes de largura e comprimento com as amostras artificiais.

Imagens	Medidas Reais (Pixels)		Medidas Obtidas (Pixels)		Erro (%)	
	Comp.	Larg.	Comp.	Larg.		
(a)	250	250	250	250	0	0
(b)	100	400	100	400	0	0
(c)	190	160	189	159	-0,52	-0,62
(d)	400	100	381	94	-4,75	-6
(e)	250	250	250	250	0	0
(f)	580	420	580	420	0	0
(g)	600	250	600	249	0	-0,4
(h)	50	50	48	48	-4	-4
(i)	450	450	449	449	-0,22	-0,22
(j)	100	100	99	99	-1	-1
(k)	300	300	299	299	-0,33	-0,33

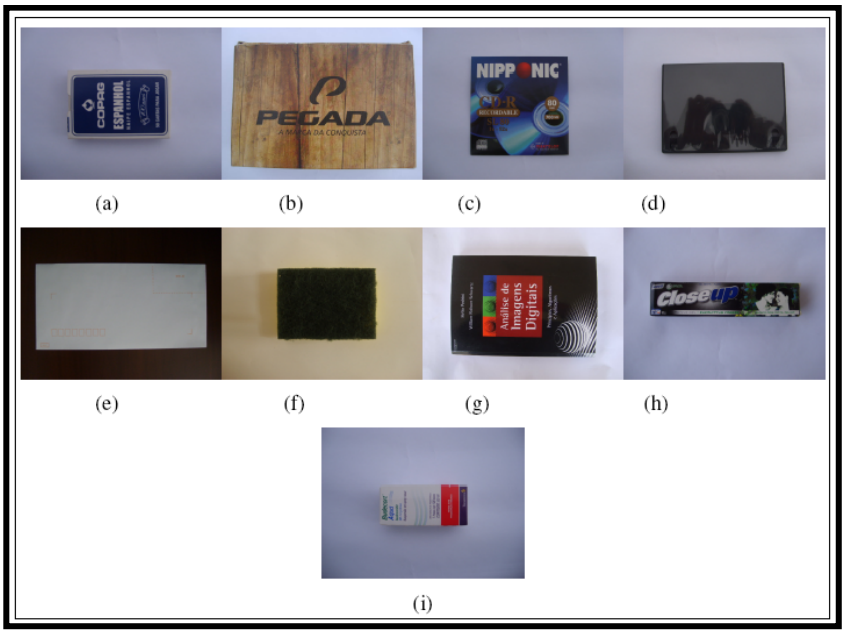
Fonte: Autor

Imagens artificiais com vértices arredondados como a figura 16 (d) resultam em maior erro devido à forma como é feita a análise dos pixels nas bordas da imagem. O limiar das imagens artificiais foi identificado de modo empírico para o valor de 150, pois a alteração do threshold mudaria completamente no valor de comprimento e largura obtidos pelo programa.
Verificou-se que imagens menores obtinham maiores erros enquanto as maiores ou médias perdiam no máximo um pixel em comparação com os tamanhos reais. O processo de medição em imagens artificiais é uma forma fácil de realizar, pois as amostras não tinham diversas cores como numa fotografia real, mas sim uma única cor.

4.2 TESTE COM IMAGENS REAIS

A partir de testes feitos em amostras de imagens artificiais que comprovam as medidas reais com as encontradas pelo programa, analisaram-se imagens de peças ou produtos com o objetivo de medir o comprimento e largura de um objeto de interesse. Preferiu-se que as imagens captadas estivessem com o fundo branco ou claro ou para imagens com muitos tons de cores. E para as de cor preta, apenas uma imagem foi utilizada com o fundo escuro, pois o objeto tinha como predomínio a cor branca. Na figura 15 são mostradas imagens reais de resolução 640x480, que foram testadas pelo programa para medir o comprimento e a largura.

Figura 15 - Imagens de prova reais.



Fonte: Autor

Na tabela 2, são comparados os valores de medidas em pixel aproximados com as medidas obtidas pelo programa. Através dessa comparação verifica-se o erro de cada imagem quanto à largura e o comprimento. As imagens analisadas são as relacionadas na figura 17.

Tabela 2 - Medidas obtidas nos testes de largura e comprimento em amostras VGA 640x480.

Imagens	Medidas Reais (Pixels)		Medidas Obtidas (Pixels)		Erro (%)	
	Comp.	Larg.	Comp.	Larg.		
(a)	329	213	321	218	-2,43	+2,34
(b)	588	394	587	400	-0,17	+1,52
(c)	312	308	314	311	+0,64	+0,97
(d)	392	285	400	286	+2,04	+0,35
(e)	543	270	544	272	+0,18	+0,74
(f)	315	220	318	227	+0,95	+3,18
(g)	458	320	465	317	+1,52	-0,93
(h)	475	100	474	108	-0,21	+8,00
(i)	282	127	283	131	+0,35	+3,14

Fonte: Autor

As imagens obtiveram bons resultados, porém necessitou-se ajustar o limiar para que as variáveis de largura e comprimento ficassem adequadas. A maior dificuldade está em processar imagens que possuem o fundo com tom próximo ao tom do objeto, como ocorre na figura 15(i). Todavia, pode-se trabalhar com fundos escuros e objetos claros para termos melhores resultados, como por exemplo, a figura 15 (e). Conforme a tabela 2 verifica-se que o erro obtido no processamento da figura 15(e). é menor que o da figura 15(i).

5 CONCLUSÃO

Observa-se que o uso de processamento de imagens na medição de objetos pode ser aplicado em indústrias que desejam aumentar o seu nível de produtividade e qualidade, reduzindo perdas. A automação desses sistemas visa aumentar o controle do processo evitando falhas e garantindo qualidade dimensional do produto. A redução do retrabalho e aumento da produtividade são resultados esperados. Imagens reais estão mais sujeitas a erros em função da luminosidade não uniforme, das sombras e da não regularidade do objeto a ser medido. Outro fator importante neste processo é identificar o limiar (Threshold) de binarização. Nos experimentos realizados necessitou-se alterar, para cada imagem real, o valor do limiar de binarização.

Os testes em imagens reais de tamanho 640x480 pixels mostraram que o erro médio é de 0,32% para o comprimento e 2,15% para a largura. Sendo que o maior erro é obtido quando a distância a ser medida é menor que 100 pixels, e o objeto possui tonalidade próxima a cor de fundo. Assim, a tonalidade do fundo da imagem deveria ser diferente da tonalidade do objeto de interesse para melhorar o contraste, otimizando os resultados.

À medida que se diminui a distância a ser medida, maior a precisão necessária na imagem, ou seja, necessita-se de maior resolução, uma vez que se tem uma menor quantidade de pixels para representar determinada medida, aumentando o erro drasticamente.

A detecção de borda melhora a robustez do sistema, principalmente nos casos de baixo contraste entre objeto e fundo. Os resultados apresentados, com baixo percentual de erros, ratifica a hipótese inicial em aplicar a inspeção visual no controle de qualidade dimensional na indústria.

6 TRABALHOS FUTUROS

Observando as dificuldades encontradas nos testes de medição em imagens reais, verificou-se a importância de detectar o ângulo do objeto na imagem e rotacionar até este posicionar-se em um ângulo próximo de 0º. Quando o objeto está desalinhado em relação às coordenadas do sistema, o resultado obtido fica fora da tolerância. Outra sugestão é desenvolver um

método automático para distinguir as formas retangulares e circulares para utilização correta das funções de medição, uma vez que na prática, uma linha de produção pode conter peças de diferentes formatos. Para essa situação é fundamental o programa reconhecer e interpretar a forma do objeto para designar o processo de medição adequado. Outra sugestão para trabalhos futuros é elaborar uma interface com o usuário no C++Builder para apresentar os resultados dos testes comparando a imagem original e a processada, apresentando as coordenadas centrais do objeto e definindo tolerâncias de erro para cada dimensão, classificando cada peça como aprovada ou reprovada.

REFERÊNCIAS

ebah. Disponível em: <http://www.ebah.com.br/content/ABAAABmmMAJ/4-paquimetro-tipos-usos >. Acesso: 22 mar 2013.

FILHO, Ogê M.; NETO, Hugo V. Processamento Digital de Imagens. Rio de Janeiro: Brasport, 1999.

GONZALEZ, Rafael C.; WOODS, Richard E. Processamento de imagens digitais. 1ª . ed. São Paulo: Edgar Blucher, 2000.

HAUPT A.G. Detecção de Movimento, Acompanhamento e Extração de Informações de Objetos Móveis. 2004. 112 p. Tese (Mestrado em engenharia) – Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2004.

Monografia – Curso de Ciência da Computação – Bacharelado, Universidade Regional de Blumenau, Blumenau, 2004.

PEDRINI, Hélio; SCHWARTZ, William R. Análise de Imagens Digitais: Princípios, algoritmos e aplicações. São Paulo: Thomson Learning, 2008.

STIVANELLO E.M. Inspeção Industrial Através de Visão Computacional. 2004. 77 p.
THE MATHWORKS. MATLAB. Disponível em: <http://www.mathworks.com>. Acesso: 22 mai 2010.

UFRGS. LILI- LaPSI Image Processing Library: Documento Oficial. Versão 0.02, 2002; Versão 1.8, 2009. Disponível em: <http://www.lapsi.eletro.ufrgs.br>. Acesso em: 06 abr. 2010

UNICAMP. Disponível em: <http://parati.dca.fee.unicamp.br/adesso/wiki/courseIA369O1S2011/Cap1/view/ >. Acesso: 22 mar 2013.

APÊNDICE A – Função lmedida.

A função lmedida calcula as seguintes medidas do objeto: comprimento máximo, largura máxima. Além disso, determina o ponto central do objeto na imagem.

```
void lmedida(limage* imagem1)
{
int Nc,Nl,temp;
long int cmax,lmax,i,j;
long int xmin,xmax,ymin,ymax,larg,larg_max,comp,comp_max;
long int soma;
larray A1;
A1=limage_as_array(imagem1,LCHANNEL_GRAY);
xmin=0; xmax=0; ymin=0; ymax=0;
cmax= limage_get_width (imagem1);
lmax= limage_get_height(imagem1);
// **** calcula a largura maxima do objeto no quadro em pixels-> larg_max
// **** calcula coluna central onde está o objeto -> pxc
soma=0;
Nc=0;
larg=0;
for (i=2;i<lmax-2;i++)
{
temp=0;
for(j=2;j<cmax-2;j++)
{
if ((A1[i][j]==255) && temp==0 && (A1[i][j+1]==255) &&
(A1[i][j+2]==255) && (A1[i+1][j]==255) && (A1[i-1][j]==255) )
{
xmin=j;
temp=1;
}
if ((A1[i][j]==255) && temp==1 && (A1[i][j-1]==255) &&
(A1[i][j-2]==255) && (A1[i+1][j]==255) && (A1[i-1][j]==255) )
xmax=j;
}
if((xmin!=0) && (xmax!=0))
{
soma=soma+((xmin+xmax)/2);
larg= (xmax-xmin);
if (Nc==0)
larg_max=larg;
Nc++;
xmin=0; xmax=0; temp=0;
}
if(larg_max<larg)
larg_max=larg;
}
if (Nc > 5)
{
pxc=(int)soma/Nc;
```

```
}
if(Nc<=5)
{
pxc=0;
pyc=0;
}
// **** calcula o comprimento máximo do objeto no quadro em pixels
// -> comp_max

// **** calcula a linha central onde está o objeto -> pyc
soma=0;
Nl=0;
comp=0;
for (j=2;j<cmax-2;j++)
{
temp=0;
for(i=2;i<lmax-2;i++)
{
if ((A1[i][j]==255) && temp==0 && (A1[i+1][j]==255) &&
(A1[i+2][j]==255) && (A1[i][j-1]==255) && (A1[i][j+1]==255) )
{
ymin=i;
temp=1;
}
if ((A1[i][j]==255) && temp==1 && (A1[i-1][j]==255) &&
(A1[i-2][j]==255) && (A1[i][j-1]==255) && (A1[i][j+1]==255) )
ymax=i;
}
if((ymin!=0) && (ymax!=0))
{
soma=soma+((ymin+ymax)/2);
comp=(ymax-ymin);
if (Nl==0)
comp_max=comp;
Nl++;
ymin=0; ymax=0; temp=0;
}
if(comp_max<comp)
comp_max=comp;
}
if (Nl >5)
{
pyc=(int)soma/Nl;
}
if (Nl <=5)
{
pyc=0;
pxc=0;
}
}
```

APÊNDICE B – Função lconvert_gray.

A função lconvert_gray converte uma imagem colorida RGB para uma imagem em tons de cinza.

```
limage* lconvert_gray(limage* imageCor)
{
limage* imageGray;
larray vetorR;
larray vetorG;
larray vetorB;
larray vetorGray;
int altura, largura;
int y, x;
31
largura = limage_get_width(imageCor); //encontra a largura da imagem
altura = limage_get_height(imageCor); //encontra a altura da imagem
imageGray = limage_new(largura, altura, LIMAGE_TYPE_GRAYSCALE ) ;
//cria uma nova imagem para tons de cinza.
vetorR = limage_as_array(imageCor, LCHANNEL_RED); //valor de vermelho
vetorG = limage_as_array(imageCor, LCHANNEL_GREEN); //valor de verde
vetorB = limage_as_array(imageCor, LCHANNEL_BLUE); //valor de azul
vetorGray = limage_as_array(imageGray, LCHANNEL_GRAY); //valor em cinza
for (y = 0; y < altura; y++) // linhas
{
for (x = 0; x < largura; x++) // colunas
{
vetorGray[y][x] = (lpixel)((vetorR[y][x] + vetorG[y][x] + vetorB[y][x])/3);
}
}
return imageGray;
}
```