UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

TÉCNICAS DE SEGMENTAÇÃO DE IMAGENS NA GERAÇÃO DE PROGRAMAS PARA MÁQUINAS DE COMANDO NUMÉRICO

Dissertação de Mestrado

Jolvani Morgan

Santa Maria, RS, Brasil

2008

TÉCNICAS DE SEGMENTAÇÃO DE IMAGENS NA GERAÇÃO DE PROGRAMAS PARA MÁQUINAS DE COMANDO NUMÉRICO

por

Jolvani Morgan

Dissertação apresentada ao Curso de Mestrado do Programa de Pós-Graduação em Engenharia de Produção, Área de Concentração em Tecnologia da Informação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Mestre em Engenharia de Produção**.

Orientador Prof. Dr. Alexandre Dias da Silva

Santa Maria, RS, Brasil.

2008

Endereço Residencial: Rua Recreio, n.º 144, Centro, Alto Alegre, RS, 99.430-000
Fone: (0xx) 54 9118 8314; Endereço eletrônico: jmorgan@inf.ufsm.br

Universidade Federal de Santa Maria
Centro de Tecnologia
Programa de Pós-Graduação em Engenharia de Produção

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

TÉCNICAS DE SEGMENTAÇÃO DE IMAGENS NA GERAÇÃO DE PROGRAMAS PARA MÁQUINAS DE COMANDO NUMÉRICO

elaborada por
Jolvani Morgan

como requisito parcial para obtenção do grau de
Mestre em Engenharia de Produção

COMISSÃO EXAMINADORA:

_____
Dr. Alexandre Dias da Silva – UFSM
(Presidente/Orientador)

_____
Dr. Flávio José Lorini – UFRGS

_____
Dr. Leandro Costa de Oliveira – UFSM

Santa Maria, Março de 2008

“– Tudo tem seu apogeu e seu declínio...
É natural que seja assim, todavia,
quando tudo parece convergir para o que supomos o nada,
eis que a vida ressurge, triunfante e bela!...,
Novas Folhas, novas flores, na infinita benção do recomeço.”

Francisco Cândido Xavier

Dedicatória

Dedico essa dissertação à mãe que me criou e ainda hoje continua me ajudando

Maria, a meus pais que me conceberam,

Setembrino e Lurdes (in memorian).

Em especial a minha esposa,

Josiani,

e a todas as pessoas que foram muito importantes na minha vida.

Agradecimentos

Agradecer a Deus;

Agradecer a Família, aos Amigos, especialmente a Esposa Josiani.

A grande pessoa que é Jaqueline Morgan uma sobrinha especial, um exemplo de vida a ser seguido, por toda ajuda e disponibilidade durante meus estudos.

Agradeço a cunhada Ligiane, por ficar com minha esposa enquanto estive fora.

Agradeço aos colegas, em especial Fernando, Eliana, Koiti, Paulo, Márcio.

Agradeço ao orientador Professor Alexandre Dias da Silva responsável pelo Grupo de Pesquisa em Automação e Processos de Perfuração, que caminhou junto, incentivando e auxiliando com paciência o trabalho realizado, e todos que contribuíram com o desenvolvimento dessa dissertação.

À Universidade Federal de Santa Maria e ao Programa de Pós-Graduação em Engenharia de Produção – PPGEP, pela oportunidade concedida.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS

API – (*Application Programming Interface*). Interface de programação para desenvolvimento de software.

AWT – (*Abstract Windows Toolkit*). API que prevê componentes gráficos com o usuário.

BITMAP – (*Device Independent Bitmap*). Formato de imagens que significa mapa de bits.

CAD – (*Computer Aided Design*). Projeto Assistido por Computador.

CAE – (*Computer Aided Engineering*). Engenharia Assistida por Computador.

CAM – (*Computer Aided Manufacturing*). Manufatura Assistida por Computador.

CAPP – (*Computer Aided Process Planning*). Planejamento do Processo Assistido por Computador.

CIM – (*Computer Integrated Manufacturing*). Manufatura Integrada por Computador

CN – Comando Numérico.

CNC – Comando Numérico Computadorizado.

GIF – (*Graphics Interchange Format*). Formato de imagens de mapa de bits muito usado na World wide web.

GUJ – Grupo de usuários Java.

IDE – (*Integrated Development Environment*). Ambiente de desenvolvimento integrado.

IDL – (Interactive Data Language). Linguagem para análise e visualização interativa de dados.

GUIs – (*Graphics User Interface*). Interface gráfica com o usuário.

JAI – (*Java Advanced Imaging*). API Java para processamento de imagens.

JPEG – (*Joint Photographic Experts Group*). Formato de imagem digital do tipo bitmap com compressão de dados.

MAT – (*Medial Axis Transformation*). Transformação do eixo médio.

RGB – (*Read, Green, Blue*). Abreviatura do sistema de cores para Vermelho, verde e azul.

TIFF – (*Tagged Image File Format*). Formato de imagem digital do tipo bitmap.

PREFÁCIO

Lendo uma dissertação me deparei com a seguinte frase de Theodore Roosevelt: “É muito melhor ousar coisas difíceis, conquistar triunfos grandiosos, embora ameaçados de fracassos, do que se alinhar com espíritos medíocres que nem desfrutam muito nem sofrem muito, porque vivem em uma penumbra cinzenta, onde não conhecem vitória nem derrota”. Interessei-me pela frase, pois ela traduz a vida cotidiana de muitos, e eu, também poderia ser um deles se não fosse à coragem de iniciar o mestrado, depois de muito tempo sem estudar. Parece-me que o autor da dissertação tinha algo em comum, apesar de nossas situações serem diferente. Ele sabia que não seria fácil, eu também, à distância de minha cidade até Santa Maria, onde me deslocava duas vezes por semana quase me fizeram desistir, quantas caronas, quantas idas, quantas vindas, algumas não sabia se estava no caminho certo, quanto frio, quanto calor, quanto sol e quanta chuva, mas nem por isso parei. Dormir pouco, má alimentação, provações passadas na busca de objetivos, no começo não tinha certeza do objetivo, só estava fazendo, pois era preciso, e mesmo assim eu seguia adiante. Trabalhar, viajar, estudar como fazer isso com a maior perfeição, nem tudo é perfeito, nem tudo é como nós queremos, mas tudo pode transformar como a certeza que a recompensa valeria a esforço. Não chorei por nenhuma dificuldade, porém quase desisti, desanimei, ia abandonar, pois parecia que mestrado não era pra mim, mas existia algo me empurrando, algo que fazia ir e vir. As disciplinas foram cursadas, faltando para conclusão, elaborar a dissertação, cujo esforço demandaria maiores provações. Nesta fase buscava encontrar assuntos relevantes e o que encontrava? Muitas vezes o vazio. Não sabia tomar decisões adiando tarefas e conseqüentemente o inicio da escrita da dissertação. Em uma ocasião, após apresentar um seminário regional, deparei-me com a decepção, aí foi um mês sem escrever e nem tocar no código que estava sendo desenvolvido. Os dias se passaram e aquilo foi superado por uma força que não conheço simplesmente ela está presente dentro de nós e quando menos esperamos, ela se manifesta. Esse foi o marco, quando na plena escuridão e desilusão do caminho um raio de luz cheio de energia caiu restabelecendo-me à vontade de vencer, para que cada dia mais e mais linhas de código fossem desenvolvidas, mais e mais linhas da dissertação fossem escritas. Hoje olho para traz e sinto orgulho pelo que fiz, pelos novos amigos que conquistei, pelos muitos obstáculos e adversidade passados, o que me enriqueceram muito. Aprendi, estudei, trabalhei, sofri, ri, brinquei e principalmente vivi muito, bons e maus momentos e agradeço a todos, a todas as dificuldades, pois elas me tornaram mais forte. A certeza que “o cara lá de cima” sempre esteve comigo, me confortando, me guiando, me amparando e, a ele a gratidão infinita. Antes mesmo de terminar o mestrado, coisas inesperadas aconteceram, oportunidade de participar em alguns projetos, comprovam que todo o esforço valeu a pena. Novos desafios estão se iniciando, o que peço é a mesma coragem e força para encará-los, entretanto o agradecimento interminável por já ter conquistado objetivos merecidos, e quando nossa vontade é de alcançar novos, a certeza de que observando o caminho do bem, com Deus tudo é possível.

RESUMO

Dissertação de Mestrado
Programa de Pós-Graduação em Engenharia de Produção
Universidade Federal de Santa Maria

TÉCNICAS DE SEGMENTAÇÃO DE IMAGENS NA GERAÇÃO DE PROGRAMAS PARA MÁQUINAS DE COMANDO NUMÉRICO

AUTOR: JOLVANI MORGAN
ORIENTADOR: DR. ALEXANDRE DIAS DA SILVA
Data e Local da Defesa: 14 de abril de 2008, Santa Maria

O presente trabalho descreve uma ferramenta para extração de características em imagens 2D aplicado à automação industrial. O método implementado utiliza técnicas de segmentação de imagens em peças modelo, como detecção de bordas, na extração das coordenadas espaciais da imagem e importação em aplicativos CAD (Computer-Aided Design) para a geração de programas de Comando Numérico Computadorizado (CNC). Este trabalho consiste em copiar um determinado modelo (imagem) partindo de um objeto já existente, o que caracteriza um processo de engenharia reversa. Um protótipo inicial foi desenvolvido usando a linguagem Java com o IDE Eclipse e a biblioteca para manipulação de imagens complexas, o API Java Advanced Image (JAI). Para a aquisição das peças a serem usinadas, foi elaborado um ambiente de iluminação a fim de ressaltar áreas de interesse melhorando a luminância das mesmas e viabilizando a aplicação das técnicas desenvolvidas nesse protótipo. Diferentes operadores de detecção de bordas foram explorados, a fim de aplicar a melhor técnica e que melhor representasse a peça real. Uma nova técnica de segmentação foi adaptada ao protótipo, a partir da integração da tecnologia Java/MatLab e com a aquisição de imagens através da técnica Multi-flash. Após a aplicação de diferentes técnicas de processamento, um arquivo texto contendo as coordenadas da imagem (peça) é gerado e importado no CAD. No ambiente CAD, é executado o cálculo da trajetória da ferramenta que identifica a geometria da peça e define o caminho da ferramenta na geração do programa CNC para reprodução da mesma. Resultados apresentados e a avaliação da ferramenta demonstram a viabilidade de aplicação do sistema desenvolvido como parte automatizada para reprodução de peças em máquinas de comando numérico.

Palavras-chave: programação CNC, segmentação de imagens, técnica multi-flash, engenharia reversa.

ABSTRACT

Master Dissertation
Program of Pos-Graduation in Engineering of Production Federal University of Santa Maria

SEGMENTING TECHNIQUES OF IMAGES IN THE GENERATION OF PROGRAMS FOR MACHINERY OF COMMAND NUMERICAL

AUTHOR: JOLVANI MORGAN
ADVISER: ALEXANDRE DIAS DA SILVA, DR.
Date and Local: April 14, 2008, Santa Maria

This work describes a tool for extraction of features in 2D images applied to industrial automation. The method uses techniques implemented targeting of images into pieces model as detection of edges in the extraction of spatial coordinates of the image and import applications in CAD (Computer-Aided Design) for the generation of programmers for Computer Numerical Control (CNC). This work is to copy a particular model (image) from an existing object, which features a process of reverse engineering. An initial prototype was developed using the Java language with the Eclipse IDE and library for handling complex images, the API Java Advanced Image (JAI). For the acquisition of parts to be machined, was prepared an environment of lighting to highlight areas of interest improving the luminance of the same and enabling the application of techniques developed in this prototype. Different operators for detecting edges were exploited in order to implement the best technical and that best represent the real piece. A new technique of targeting the prototype was adapted from the integration of technology Java / MatLab and with the acquisition of images through technical Multi-flash. After the application of different processing techniques, a text file containing the coordinates of the image (piece) is generated and imported into CAD. In the CAD environment, runs calculating the trajectory of the tool that identifies the geometry of the piece and defines the path of the tool in the generation of CNC program for breeding the same. Results presented and evaluation of the tool demonstrates the feasibility of applying the automated system developed as part of parts to play in numerical control machines.

Keywords: CNC programming, targeting of images, multi-flash technique, reverse engineers.

1 INTRODUÇÃO

O processamento digital de imagens é uma área que vem abrangendo muitos estudos nos últimos anos. Este processo é implementado através de funções matemáticas que transformam as imagens através das diferentes técnicas de processamento. Sua aplicação concentra-se em diversas áreas do conhecimento. Aliada a evolução da tecnologia digital e ao desenvolvimento de novos algoritmos, vem permitindo uma gama de aplicações cada vez maior (GRANDO, 2005).

Dentre as principais áreas, o processamento de imagens é vastamente aplicado na medicina principalmente na ajuda de diagnósticos de patologias, em geo-processamento e sensoriamento remoto na visualização do clima de uma determinada região, na arquitetura e nas engenharias (elétrica, civil, mecânica) para quantificar uma fração de área em uma determinada amostra, por exemplo. Na engenharia mecânica, o processamento de imagens é utilizado para automatizar processos industriais, sendo que uma das principais ferramentas aplicada a esta é o sistema CAD (*Computer-Aided Design*).

Neste contexto, segundo Silva (2003), a automação industrial compreende as máquinas industriais controladas pelo computador, para alcançar melhorias no setor produtivo com menor custo, melhorar a qualidade e obter maior eficiência. Como exemplo, encontra-se controles automáticos na produção de peças, dispositivos de controles programáveis de máquinas e equipamentos (usados para acionar o equipamento ou desligá-lo), robôs (utilizados para operações de fabricação, manuseio ou movimentação de materiais e peças) e estações remotas de monitoração de processos industriais e equipamentos CNC (Comando Numérico Computadorizado) para reprodução de peças. Neste trabalho, assuntos relacionados à tecnologia CNC e CN (Comando Numérico), serão tratados como tecnologia CNC.

Diversas indústrias, em especial a aeronáutica e a automotiva vêm usufruindo com o uso da tecnologia CNC. Sua aplicação no controle de máquinas-ferramenta permite a realização de tarefas repetitivas e de grande complexidade cinemática. Isto possibilita a reprodução de produtos de variadas formas geométricas.

Segundo Costa e Pereira (2006), esta tecnologia, aliada com outros sistemas de informação revolucionaram o modo de operação nos sistemas de produções nos quais foi adotada. Exemplo disso, o CNC associado à modelagem digital em sistemas CAD (*Computer-Aided Design*) e CAM (*Computer-Aided Manufacturing*), suporta em grande parte a

transferência do modelo de um produto para a máquina com pouca intervenção humana, além de propiciar a substituição do meio de transmissão do papel para o eletrônico.

Porém, a tarefa de elaboração de programas para usinagem CNC é um processo um tanto demorado, devido o fato do alto grau de dificuldade proporcional à complexidade da geometria da peça. Esta tarefa, primeiro exige a fase de estudo e elaboração do programa propriamente dito, segundo, a introdução do programa na memória da máquina, testes e simulação.

Para a fase de testes e simulações encontramos vários projetos de sistemas de manufatura, como os sistemas CAD/CAM que podem ser utilizados, porém um sistema CAD é apropriado para construir a geometria da peça, e não extrair características geométricas através da aquisição de uma imagem. Tal função deve ser implementada em uma ferramenta de processamento de imagem, para permitir que os sistemas CAD importem as características desejadas. Esta abordagem consiste em técnicas de engenharia reversa, que neste trabalho busca reproduzir um objeto (peça) partindo de um modelo digital através de outro objeto existente.

Entretanto, a motivação para o desenvolvimento desse trabalho é de integrar o processamento de imagens à programação CNC, para automatizar um processo produtivo industrial. Disponibilizando uma ferramenta, como protótipo inicial e adoção de outras como o MatLab, para tratamento de imagens que aplicado a sistemas CAD permitem a geração do programa CNC e sua reprodução em tornos, ou seja, implementar e usar ferramentas e técnicas de processamento de imagens para importar os dados em sistemas CAD permitindo a geração automática do programa CNC, limitando-se a manipulação de imagens no plano bidimensionais, 2D.

Por fim, busca-se aliar o processamento de imagens a processos de automação industrial na reprodução de peças metalúrgicas, contribuindo na automação dos processos de fabricação assistida por computador.

1.1 Objetivos

O objetivo deste trabalho é elaborar um “protótipo” como ferramenta que integre funções de processamento de imagens para extração de características de uma determinada

imagem digital adquirida através de câmaras digitais para importação dos dados extraídos a sistemas CAD e geração automática do programa CNC.

1.2 Estrutura da Dissertação

Este trabalho está dividido em capítulos, que abrangem assuntos sobre processamento de imagens para automação industrial, programação Java, Biblioteca JAI, MatLab e a técnica *Multi-flash* (integração Matlab e Java), sistemas CAD/CAM e implementação de um protótipo inicial como ferramenta para processamento de imagem para importar os dados geométricos ao CAD e geração automática do programa CNC na criação e reprodução de peças de máquinas e equipamentos em geral. Entretanto este documento está dividido em seis (6) capítulos, assim discriminado:

- Capítulo 1: Refere-se à introdução do presente trabalho, com o objetivo de expor ao leitor o processamento de imagens como ferramenta para solução de diversos problemas, abrangendo diferentes áreas do conhecimento. Assim como, a motivação, os objetivos e a estrutura dessa dissertação, que faz uma breve descrição dos capítulos que compõem o documento.
- Capítulo 2: Será apresentada a revisão da literatura abordando conceitos básicos de processamento de imagens, a segmentação como forma de extrair os objetos de interesse para este trabalho. Uma abordagem envolvendo programas CNC, sistemas CAD/CAM e automação industrial, bem como a técnica de aquisição de imagens *Multi-flash* e a integração Java com Matlab.
- Capítulo 3: Uma abordagem da metodologia utilizada para a realização deste trabalho, bem como as ferramentas, e a preparação do ambiente de trabalho, o uso da linguagem Java, a Biblioteca JAI e os passos para alcançar os objetivos na geração das coordenadas e do programa CNC no CAD.
- Capítulo 4: Apresentação da ferramenta *JImage Automation* e sua aplicação em imagens de demonstração, passando pelas diferentes técnicas de processamento e pelas opções disponíveis na ferramenta. A aplicação da técnica *Multi-flash* e a demonstração do cálculo da trajetória da ferramenta efetuada no Autocad.
- Capítulo 5: Uma aplicação real do trabalho, desde a aquisição da imagem de forma convencional, e a aquisição das imagens Multi-flash, aplicação das

técnicas de segmentação elaboradas, cálculo da trajetória e a geração do programa CNC, onde é efetuada a avaliação da ferramenta e discussão dos resultados.

- Capítulo 6: As considerações finais do trabalho realizado e a abordagem de trabalhos futuros que podem contribuir com o aprimoramento da ferramenta e novos estudos na área.
- Capítulo 7: Referências Bibliográficas.

O objetivo do trabalho é a geração automática do programa CNC, porém o foco de atuação é o processamento de imagens buscando um estudo de melhores técnicas para extrair as características das imagens adquiridas. Entretanto, o trabalho aborda a implementação da ferramenta desenvolvida em Java e a aplicação da técnica *Multi-flash* para segmentação de imagens através da integração da linguagem Java e MatLab.

2 REVISÃO DA LITERATURA

Os métodos de processamento de imagens contemplam duas áreas principais de aplicação conforme Gonzalez e Woods (2002): primeiro, processar a imagem a fim de melhorar a informação visual para interpretação humana, ou seja, aplicar procedimentos e técnicas para melhoria de contraste, realce e restauração de imagens danificadas; e a segunda, aplicam-se procedimentos para extrair da imagem informações de forma adequada, a fim de usá-las para posterior processamento computacional.

Dessa forma, o trabalho se enquadra na segunda parte do processamento, ou seja, na extração das características da imagem para posterior uso da informação em sistemas CAD e reprodução do objeto imageado. Também, relaciona-se com os aspectos de aquisição das imagens, como: equipamento utilizado, iluminação e angulação, e a abordagem da aquisição *multi-flash* para aplicação da técnica de segmentação de profundidade.

Entretanto, este Capítulo aborda assuntos referentes ao processamento de imagens, em especial a segmentação e afinamento de imagens, estudos sobre a integração do Java e Matlab, alguns trabalhos realizados na geração de código CNC, e a técnica *multi-flash* usado na detecção de contornos das peças metalúrgicas.

2.1 Representação da Imagem Digital

A imagem digital pode ser monocromática ou colorida, e é classificada conforme sua quantidade de bits:

Compostas por um (1) bit: Suportam somente duas (2) tonalidades de cores, o preto e o branco, sendo que o preto é representado pelo valor zero (0) e o branco pelo valor um (1), pertencente à classe das imagens monocromáticas, denominadas imagens binárias.

Compostas por oito (8) bits: Suportam até 256 tonalidades de cores, representadas pelo branco, preto, cinza, cinza claro e cinza escuro, cujos valores variam de 0 a 255, também pertencem à classe monocromática, são chamadas de imagens em tons de cinza.

Compostas por vinte e quatro ou trinta e dois (24/32) bits: Suportam até 16 milhões de tonalidades de cores, formadas pela combinação das três cores básicas vermelho, verde e

azul, pertencente à classe das coloridas.

A quantidade de bits informa quantas tonalidades de cores uma imagem pode ter, entretanto toda imagem digital é formada por pontos de forma quadrada chamados *pixel.* O termo *pixel* é uma abreviatura do inglês *Picture element* que significa elemento da figura. Corresponde a menor unidade de uma imagem digital, onde são descritos a cor e o brilho específico de uma célula da imagem. Cada *pixel* é criado quando a cor e brilho de uma dada posição na matriz 2D são mensurados e armazenados como uma quantidade discreta, segundo Laurega (2006).

Conforme Laurega (2006) e Gattass (2007), a quantidade de *pixels* presentes na imagem fornece o tamanho da mesma. O tamanho do arquivo digital não se refere apenas à quantidade de detalhes visível, ou seja, a resolução espacial, mas também ao número de cores presentes, a profundidade de cor. Quanto maior a quantidade de *pixels,* maior será o tamanho da imagem e melhor a qualidade de detalhe visível da mesma. Em imagens coloridas o *pixel* é formado por três cores. A figura 1 mostra a representação do *pixel* da imagem colorida, padrão RGB.

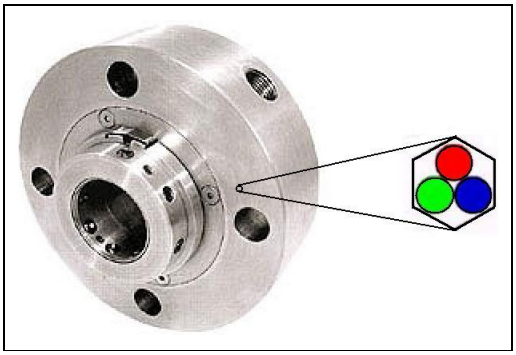
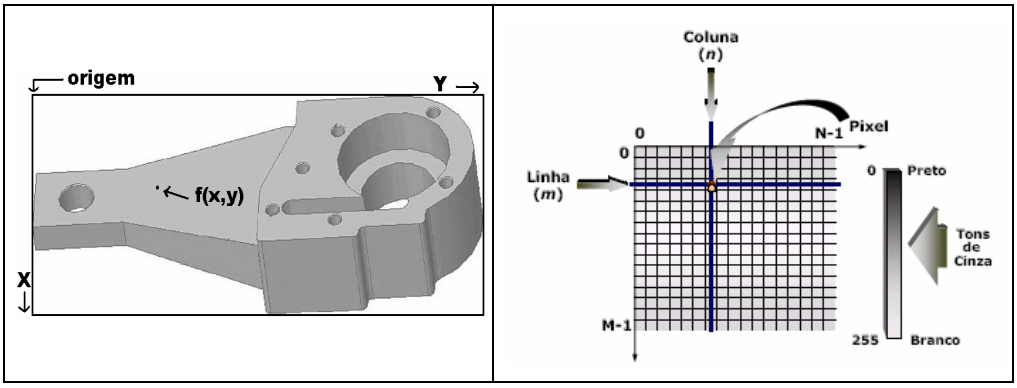


Figura 1 – Composição do *pixel* da imagem

Para processar uma imagem, primeiramente é necessário identificar como representá-la. Assim, a imagem é uma representação gráfica de objetos do mundo real, matematicamente, uma imagem pode ser descrita por uma função bidimensional $f(x,y)$, onde as coordenadas x e y representam a intensidade luminosa (brilho ou nível de cinza) naquele ponto. Ou seja, em uma imagem monocromática a função $f(x,y)$ representa os tons de cinza, ou nível de cinza, no ponto (x,y) do espaço daquele ponto representado, segundo Russ (1999).

Entretanto, os computadores não são capazes de processar imagens contínuas, mas apenas *arrays* de números digitais, então é necessário representar imagens como arranjos bidimensionais de pontos, Queiroz e Gomes (2006). A figura 2 item (a) mostra a

representação de uma imagem digital monocromática e a figura 2 item (b) a representação bidimensional da imagem.



a b

Figura 2 – a) Representação de imagens digitais, onde *f(x,y)* corresponde os *pixels* da imagem, Bastos (2001). b) representação bidimensional da imagem como matriz de pontos. Queiroz e Gomes (2006).

Para cada ponto da grade bidimensional apresentado na figura 2 item (b) que representa a imagem digital, é denominado *elemento de imagem (Picture element)* ou *pixel*. Pode-se observar na figura a notação matricial para a localização de um *pixel* no arranjo de *pixels* de uma imagem bidimensional. O índice *m* representa a posição da *linha* e o índice *n* à posição da *coluna*, onde o *pixel* se encontra. Se a imagem possuir *M* linhas e *N* colunas, o índice *m* irá variar de 0 a $M - 1$, enquanto o índice *n* variará de 0 a $N - 1$, segundo Queiroz e Gomes (2006).

A intensidade luminosa no ponto (x,y) pode ser decomposta em: (i) componente de *iluminação*, $i(x,y)$, associada à quantidade de luz incidente sobre o ponto (x,y); e a componente de refletância, $r(x,y)$, associada à quantidade de luz refletida pelo ponto (x,y). O produto de $i(x,y)$ e $r(x,y)$ resulta em: $f(x,y) = i(x,y) * r(x,y)$, onde $0 < i(x,y) < \infty$ e $0 < r(x,y) < 1$, sendo $i(x,y)$ dependente das características da fonte de iluminação, enquanto $r(x,y)$ dependente das características das superfícies dos objetos, Gonzáles e Woods (2002).

No caso de uma imagem possuir informações em intervalos ou banda distintas de freqüência, é necessária uma função $f(x,y)$ para cada banda. Exemplo disso é o caso das imagens coloridas, padrão *RGB,* mencionadas anteriormente. Tais imagens são formadas por três cores básicas ou primárias, o vermelho *"Red"*, o verde *"Green"*, e o azul *"blue"*. A imagem colorida pode ser vista como a composição de três imagens monocromáticas, ou seja: $f(x,y) = fR(x,y) + fG(x,y) + fB(x,y)$, onde $fR(x,y)$, $fG(x,y)$ e $fB(x,y)$ representam, respectivamente, as intensidades luminosas das componentes vermelha, verde e azul da

imagem, no ponto (*x,y*). Entretanto, pode-se representar uma imagem colorida sendo uma função que atribui a cada ponto do domínio retangular um ponto no espaço de cores, constituindo o cubo de cores RGB. A figura 3 apresenta a imagem colorida representando os três canais: Cubo RGB Gattass (2007).

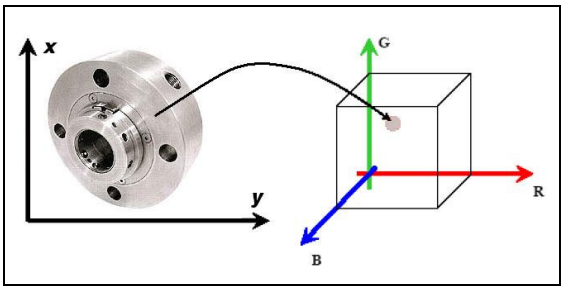
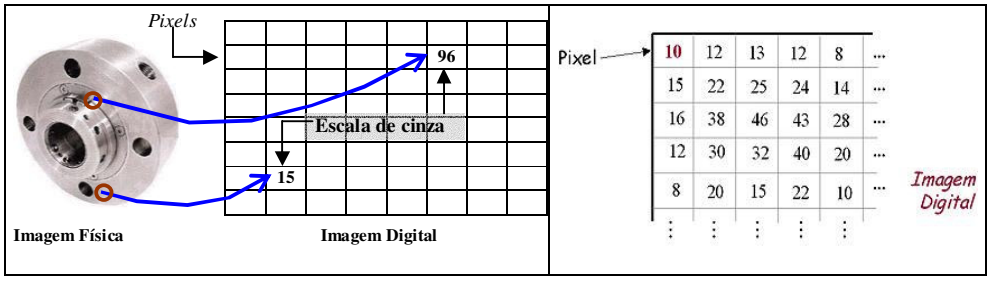


Figura 3 – Cubo RGB, representação em três canais Gattass (2007).

Diferentes das imagens monocromáticas, uma imagem colorida não pode ser representada por um único valor, pois elas são compostas de três cores primárias, e a combinação destas para representarem muitas outras. No entanto, uma forma simples de apresentar o modelo monocromático com três cores consiste em considerar três funções distintas, ao invés de uma. Uma para cada canal do RGB.

Dessa forma, para o tratamento computacional da imagem, é fundamental representar sua informação num formato adequado, assim às imagens são representadas por matrizes, onde os índices *linha* e *coluna* referenciam o brilho médio amostrado no ponto correspondente da cena, segundo Facon (2002).

Entretanto, conforme Albuquerque (2000), a representação da matriz de *pixel*, corresponde aos valores do *pixel* da imagem, definindo o mapeamento dessa imagem para que possa ser representada pelo computador. A figura 4 item (a) mostra o mapeamento de uma imagem física para imagem digital e a figura 4 item (b) exibe um exemplo de uma imagem digital mapeada.



a b

Figura 4 – a) mapeamento da imagem física para imagem digital; b) imagem digital mapeada.

Cada *pixel* possui três (3) coordenadas: às coordenadas ***x*** e ***y*** que definem a posição do *pixel* na imagem e a coordenada com valor de ***z*** que representa o nível de cinza ou o grau de brilho naquele ponto, ou seja, corresponde a uma unidade de informação em termos da quantidade de energia refletida ou emitida pelo objeto imageado.

Assim, o computador cria uma matriz e cada elemento *(x,y)* da matriz, corresponde a um ponto da imagem, esse ponto é denominado *pixel*, ou seja, uma imagem digital *f(x,y)* é representada como uma matriz cujos índices de linhas e de colunas identificam um ponto na imagem que corresponde o valor da intensidade (brilho) do *pixel*, como ilustrado anteriormente na figura 1 e 2 item (b).

2.2 Operações Sobre Imagens

Operações sobre imagens podem ser efetuadas de diferentes maneiras, aquelas cuja manipulação é aplicada diretamente nos pixels, são denominadas operações de domínio no espaço, Queiroz e Gomes (2006). Estas operações podem ser classificadas como *pontuais* (ponto-a-ponto) ou *locais* (localizadas).

Nas operações pontuais cada *pixel* da imagem de saída depende do mesmo *pixel* correspondente da imagem de entrada, representando assim, um mapeamento dos *pixels* da imagem de entrada para a imagem de saída. A figura 5 mostra a operação pontual *unária*.

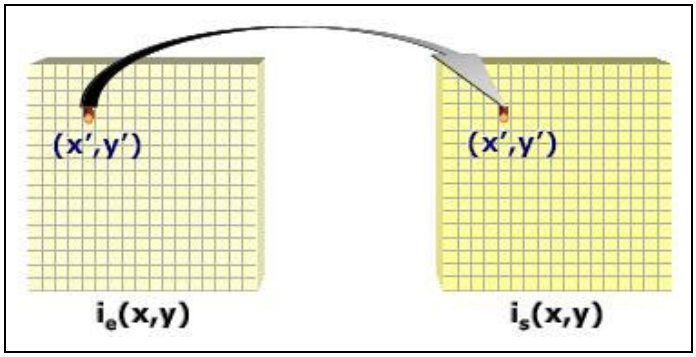


Figura 5 – Operação pontual *unária*, conforme Queiroz e Gomes (2006).

Na imagem de saída, cada ponto é obtido por uma operação entre os pontos de coordenadas homólogas das imagens de entrada, $I_e(x,y) = I_s(x,y)$.

Já nas operações locais, o valor de saída em uma coordenada específica depende dos valores de entrada daquela coordenada e sua vizinhança, assim, são operações efetuadas sobre regiões ou vizinhança de *pixels* da imagem.

Segundo Marquês Filho e Vieira Neto (1999), cada elemento corresponde a um ponto da imagem, um *pixel p*, a partir desta relação tem-se que cada ponto de uma imagem possui quatro (4) vizinhos horizontais e verticais, e quatro (4) vizinhos diagonais. Estes *pixels*, que compõem os quatro (4) vizinhos horizontais e verticais, conjunto $N_4(p)$, e diagonais, conjunto $N_d(p)$, de p, formam a chamada '4-Vizinhança' de p.

Por outro lado, a '8-Vizinhança' é constituída pela união dos vizinhos horizontais e verticais com os vizinhos diagonais, sendo assim, tem-se que a '8-Vizinhança' constitui o conjunto $N_8(p) = N_4(p) \cup N_d(p)$. A Figura 6 ilustra esse conjunto de vizinhanças do ponto p.

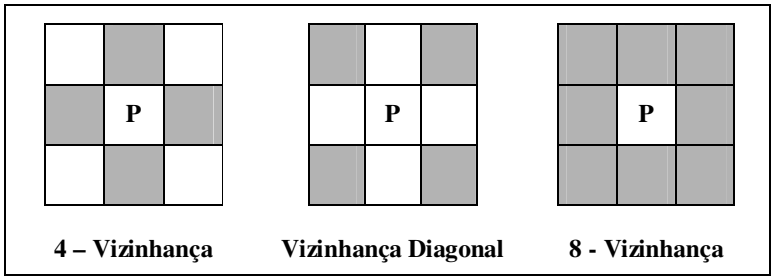


Figura 6 **–** Conceitos de Vizinhanças Filho e Neto (1999).

Conforme Queiroz e Gomes (2006), 4-Vizinhança compreendem os vizinhos mais próximos de um *pixel p* considerado, de coordenadas *(i,j).* São representados pelas coordenadas *(i+1,j), (i-1,j) (i,j+1)* e *(i,j-1).* 8-Vizinhança compreende tanto os vizinhos mais próximos quanto os vizinhos mais distantes do *pixel* considerado. Os vizinhos mais distantes

representam os *pixels* de coordenadas *(i-1,j-1), (i-1,j+1), (i+1,j-1)* e *(i+1,j+1).* A figura 7 mostra as vizinhanças tipicamente utilizadas em operações locais.

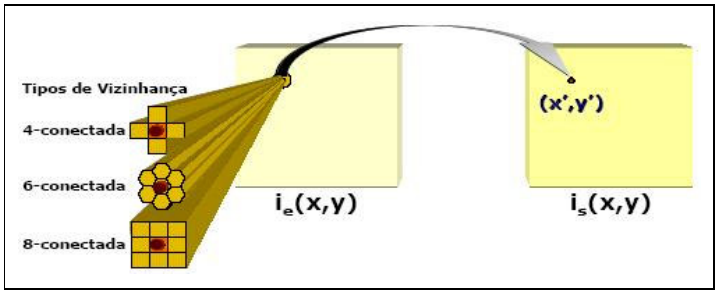


Figura 7 – Tipos de vizinhança em operações locais

É possível processar *grades* de *pixels* hexagonais, neste caso, operações locais envolvem os seis (6) vizinhos mais próximos (6-Vizinhança ou vizinhança *6-conectada*).

As operações locais e pontuais podem ser aplicadas de diversas maneiras em imagens, como expansão do contraste, inversão em escala de cinza, equalização ou modificação através do histograma, e através do uso de filtros. Em geral essas operações são utilizadas para melhorar, modificar ou identificar áreas de interesses na imagem.

2.3 Segmentação

A segmentação de imagens representa um processo fundamental para extração e identificação de objetos ou áreas de interesse da imagem, bem como permite reduzir a informação da mesma a fim de manipular objetos simples de uma cena que geralmente correspondem a linhas ou regiões (grupos de pontos conectados) de interesse.

Conforme Falcão (2003), os métodos de segmentação podem ser classificados como interativos ou automáticos. Assim, o primeiro é executado através da intervenção do usuário, e o segundo não. Ou ainda, podem ser classificados de acordo com a estratégia de representação dos objetos a serem segmentados, são elas: métodos orientados a borda ou orientados a regiões.

O método orientado a bordas detecta inicialmente na imagem os segmentos de borda e depois une os segmentos que pertencem a uma mesma borda. Esta abordagem é denominada *edge-linking*. Outra abordagem, denominada *contornos deformáveis*, parte de um modelo global que deve ser deformado durante a segmentação até aderir completamente à borda do

objeto de interesse.

O método orientado a regiões é dividido em três abordagens relevantes: *classificação por pixels*, *agregação de pixels* e *split-and-merge* (divisão e conquista). A seguir a descrição dos mesmos:

1 – Classificação por pixels: são observadas características da imagem como cor, gradiente, textura, para classificar os *pixels* como pertencentes a uma das *N* possíveis classes ou objetos da imagem.

2 – Agregação por pixels: é baseada no crescimento de regiões conexas a partir de *pixels* sementes e de um critério de parada no crescimento, onde no final cada região represente um objeto.

3 – Split-and-merge (divisão e conquista): Nesta técnica, a imagem é particionada em regiões conexas que podem ser fundidas ou subdivididas em novas regiões conexas, onde no final cada região represente um objeto.

A figura 8 apresenta as estratégias de segmentação segundo Falcão (2003).

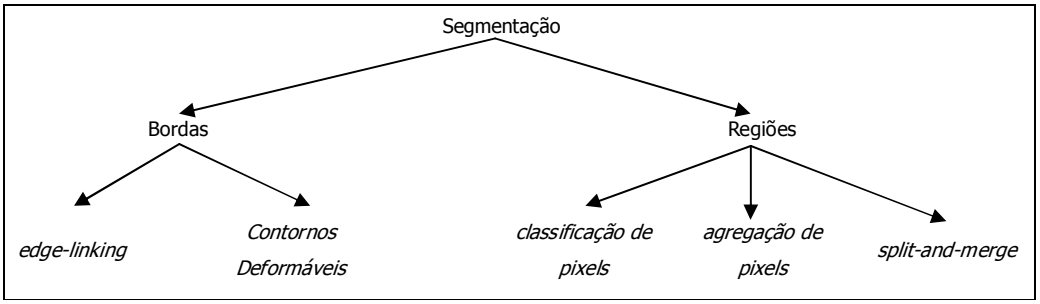


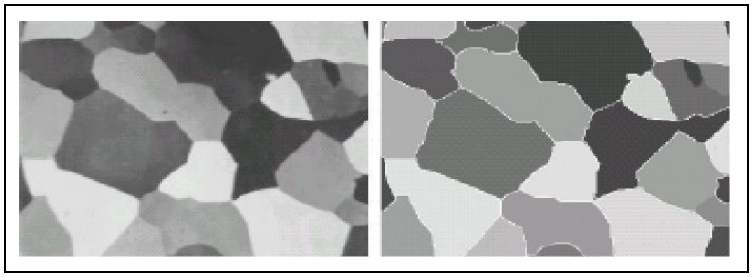
Figura 8 – Estratégias de Segmentação

Segundo Carvalho e Brito (1998)**,** o processo de segmentação consiste em uma divisão ou separação de uma imagem em regiões de atributos similares. Os atributos básicos utilizados é a amplitude e a luminância, isto quando se trata de imagens monocromáticas, e atributos de cores (brilho, contraste, suavização, etc) para imagens coloridas, assim como contornos (bordas) de imagens também são atributos úteis à segmentação.

Conforme Fonseca (2000) entende-se por regiões um conjunto de *pixels* contíguos, que se espalham bidimensionalmente e que apresentam uniformidade em relação a um dado atributo. Atributos das regiões tais como área, forma, parâmetros estatísticos e textura podem ser extraídos e usados posteriormente no processo de análise. Tal particionamento,

basicamente pode ser realizado de três formas: por crescimento de regiões; detecção de bordas, ou a combinação das duas.

A detecção de regiões pode ser feita de dois modos: através do agrupamento de pontos vizinhos com características semelhantes, ou através da determinação das fronteiras (bordas) da região. Ou seja, segundo Laurega (2006) a segmentação de regiões é usada para extrair uma determinada região ou dividir a imagem num conjunto de áreas distintas, cuja definição é o conjunto de pontos ligados onde de qualquer um dos seus pontos pode-se chegar a qualquer ponto dessa mesma região ou por um caminho completamente contido em seu interior. Tais regiões geralmente apresentam características homogêneas que é a continuidade do nível de cinza de seus *pixels*. A figura 9 exibe a segmentação por regiões.



a b

Figura 9 – Segmentação de imagens por regiões: a) original b) segmentada, Laurega (2006).

Conforme Gonzales e Woods (1992) e Grando (2005), os algoritmos de segmentação para as imagens monocromáticas são baseadas em duas propriedades básicas de níveis de cinza: a descontinuidade e similaridade. A descontinuidade refere-se à partição da imagem sendo mudanças bruscas nos níveis de cinza, como na detecção de pontos, linhas e bordas. Já a similaridade baseia-se na limiarização, crescimento de regiões, e divisão e fusão de regiões. O processo de segmentação tem como objetivo diminuir a quantidade de iteração humana, visando reduzir o trabalho e diminuir a subjetividade.

Entretanto a segmentação abrange uma vasta área dentro do processamento de imagens, mas a que cabe ao escopo do trabalho é a segmentação por similaridade baseada na limiarização e a segmentação por descontinuidade baseado na detecção de bordas.

2.3.1 Limiarização (*Thresholding*)

A limiarização consiste em separar a imagem em duas partes, diferenciando a região que representa o objeto de interesse, da região que representa o fundo da imagem. Tal operação geralmente é realizada através do histograma, ou seja, para extrair os objetos do fundo da imagem usa-se a bipartição do histograma, conhecido como a binarização da imagem, conforme Grando (2005).

Esta binarização identifica a diferença de nível de cinza entre os *pixels* através de um valor de limiar. O limiar é um valor referencial que classifica os *pixels*, como branco representando o ***objeto*** e valor igual a um (1), e o restante como preto representando o ***fundo*** e valor igual à zero (0).

Uma imagem com fundo e objeto bem distintos, cujos níveis de cinza possuem uma variação bem definida, ter-se-á um histograma com dois picos bem distintos separados por um "*vale*" de valores relativamente baixos, Facon (2002). Neste caso, conforme Grando (2005) matematicamente a limiarização pode ser definida como:

$$g(x,y) = \begin{cases} 1 & se \quad f(x,y) > T \\ 0 & se \quad f(x,y) \leq T \end{cases}$$

Onde *f(x,y)* é a imagem de entrada, T é o valor de limiar e *g(x,y)* é a imagem de saída, imagem limiarizada. Na fórmula, se a primeira condição for verdadeira onde que $f(x,y) > T$, então este representa o ***objeto***, e o valor do *pixel* é um (1), caso contrário representará o ***fundo*** e o valor do *pixel* será zero (0).

A limiarização pode ser feita manual ou automaticamente. O método manual é baseado no histograma da imagem, que representam os níveis de cinza da mesma, onda a escolha do limiar é feita pelo usuário da aplicação. No método automático, não existe a necessidade de escolha do valor de limiar, pois este deve ser retornado pelo próprio algoritmo. Neste trabalho a aplicação de limiarização será efetuada manualmente com o auxilio do componente *Slider*, que através de uma barra deslizante permite alterar o valor do limiar para aplicação do *threshold* na imagem.

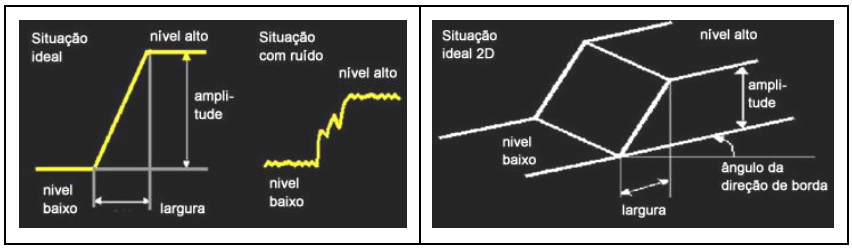
Outra característica de limiarização é quanto sua aplicação, que pode ser local ou global. A local destina-se em dividir a imagem em sub-regiões, cujo valor de limiar é específico para cada uma. Já a global, utiliza um único limiar para toda imagem, sendo o método implementado nesta dissertação.

2.3.2 Detecção de Bordas

Uma borda pode ser definida como a mudança brusca em nível de cinza entre duas regiões relativamente homogêneas, ou seja, a fronteira entre duas regiões cujo nível de cinza encontrado nelas é diferente, Filho e Neto (1999). Cada região é uniforme e homogênea com relação a alguma propriedade da amplitude, tais como tom ou textura, e o valor desta propriedade difere de maneira significante de acordo com a vizinhança de cada região, segundo Mascarenhas e Velasco (1994).

Conforme Seara (1998) e Wangenheim (2006), as bordas são classificadas em unidimensionais e bidimensionais. As unidimensionais são definidas com mudança de intensidade de baixa para alta, sendo que a intensidade do sinal pode ter a interferência de ruídos. A borda bidimensional tem como definição àquelas em que existem descontinuidades, que podem ocorrer ao longo de certas linhas e orientações, sendo que as orientações são características importantes em bordas bidimensionais.

A figura 10 mostra o gráfico de intensidade que representa uma borda bidimensional segundo Wangenheim (2006).



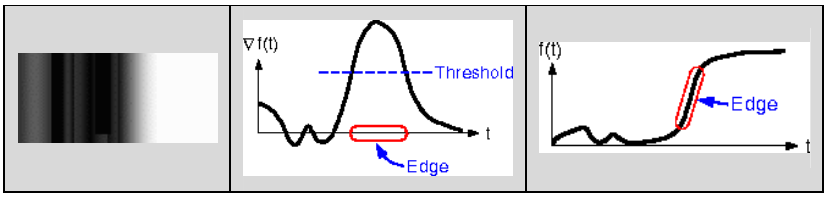
a b

Figura 10 – a) Gráfico de intensidade de uma imagem b) Gráfico de intensidade de uma borda bidimensional.

Para realizar a detecção de bordas existem vários métodos, sendo que uma das operações mais utilizadas é a de diferenciar imagens, o que resulta tanto na detecção de bordas físicas quanto de bordas que não se consegue diferenciar. Ao suavizar a imagem antes de fazer a diferenciação, pode-se diminuir esse efeito indesejado, o que ajuda tanto na detecção quanto na precisão da localização das bordas. Seara (1998). Exemplo de detecção de bordas com a imagem clássica de *Lena* pode ser observado na figura 11.

Figura 11 – Detecção de bordas da imagem clássica de *Lena*

Outra maneira é aplicar operadores de derivação, tais como o gradiente, conforme Santos (2001), a figura 12 apresenta a função *f(t)* representando as variações em nível de cinza. Em (a) é mostrado um espectro típico em nível de cinza de uma borda representado pelo gráfico em (c), e em (b), é apresentado o gráfico da magnitude do gradiente da mesma função. Neste gráfico, observa-se à amplitude alcançada na região da borda, possibilitando assim, o uso de *threshold* (limiarização) o que separa os *pixels* da borda.



a b c

Figura 12 – a) espectro da borda em nível de cinza, b) aplicação do operador gradiente (*threshold*), c) Gráfico resultante após aplicação do gradiente. Santos (2001).

Dentre as diferentes técnicas usadas para detecção de bordas, usualmente na literatura, encontram-se alguns métodos conhecidos como convolução por máscaras (filtros), ou kernels de convolução, através de diferentes operadores de gradiente. A seguir uma abordagem sobre máscaras de convolução e aplicação de operadores na detecção de bordas.

2.3.3 Convolução por Máscaras

A aplicação de operadores ou filtros de convolução resume-se em segmentar a imagem. Segundo Russ (1999) o processo pelo qual se divide a imagem em regiões de interesse, segmentação, é baseado na propriedade do *pixel* que forma a imagem. A intensidade de cada *pixel* pode ser representada com valores que variam no intervalo de 0 a 255, e para

detecta a borda da imagem é necessário verificar mudanças bruscas em nível de cinza.

Conforme Queiroz e Gomes (2006), a máscara (*mask, kernel* ou *template*) de convolução é um arranjo matricial de dimensões inferiores à imagem a ser filtrada, cuja técnica implica em transformações *pixel* a *pixel* da imagem que depende dos valores dos níveis de cinza do *pixel* correspondente e dos *pixels* vizinhos. Está convolução é uma operação local com valores definidos como fatores de ponderação, ou pesos, a serem aplicados sobre os *pixels* da imagem. Esta operação é aplicada progressivamente sobre os *pixels* coluna a coluna e linha a linha, conforme está ilustrado na figura 13.

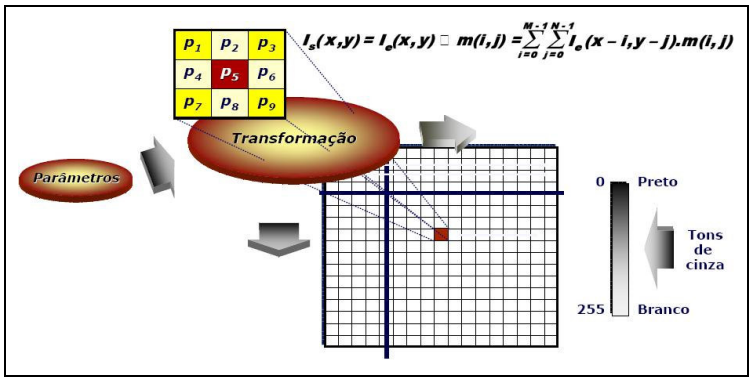


Figura 13 – Representação da aplicação de filtragem por convolução Queiroz e Gomes (2006).

O conceito de convolução por máscara é amplamente utilizado no processamento de imagens, nas quais uma seleção apropriada dos coeficientes que compõem a máscara de convolução torna possível uma grande variedade de operações úteis, tais como e redução de ruído, afinamento e detecção de características da imagem. Isso dado o fato de que através de matrizes e suas multiplicações pode-se representar todas as transformações lineares 2D e 3D, Filho e Neto (1999).

A operação de convolução determina a influência dos *pixels* vizinhos sobre um *pixel* central na matriz de *pixels* de uma imagem, esta operação consiste basicamente na multiplicação de duas matrizes, sendo a primeira, uma matriz *N x N* que corresponde à máscara a ser aplicada, e a segunda, uma matriz composta pelos vetores dos *pixels* de uma imagem. Assim, aplica-se o processo de multiplicação dos valores das duas matrizes no qual a resultante desta multiplicação será a nova imagem transformada como ilustra a figura 14. A operação de convolução por máscara pode ser representada matematicamente por: $C(i,j) = \sum_{m=0}^{(Ma-1)} \sum_{n=0}^{(Na-1)} A(m,n) * B(i-m, j-n)$, onde, $0 \leq Ma + Mb - 1$ *e* $0 \leq j < Na + Nb - 1$,

sendo que *Ma* e *Na* correspondem as dimensões da matriz A e *Mb* e *Nb* correspondem as dimensões da matriz B Mathworks (1994).

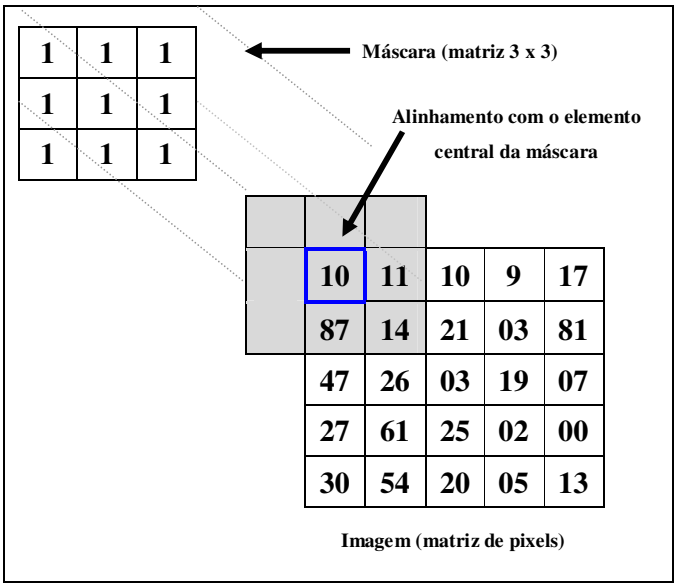


Figura 14 – Conceito de Matriz de Convolução Mathworks (1994).

Além disso, varias transformações podem ser combinadas resultando em uma única matriz, denominada matriz de transformação. Entretanto, operações de convolução podem ser efetuadas com máscara 3 x 3, 5 x 5, 7 x 7, entre outras.

A figura 15 apresenta um exemplo de aplicação da máscara de convolução, onde a figura 15 item (a) representa a matriz correspondente à imagem original, a figura 15 item (b) representa a matriz correspondente à máscara e a figura 15 item (c) representa a matriz correspondente ao resultado da operação de convolução.

5	8	3	4	6	2	3	7
3	2	1	1	9	5	1	0
0	9	2	1	1	9	5	1
4	2	7	2	1	9	0	6
9	7	9	8	0	4	2	4
5	2	1	8	4	1	0	9
1	8	5	4	9	2	3	8
3	7	1	2	3	4	4	6

*

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

=

20	10	2	26	23	6	9	14
18	1	-8	2	7	3	3	-11
14	22	5	-1	9	-2	8	-1
29	21	9	-9	10	12	-9	-9
21	1	16	-1	-3	-4	2	5
15	-9	-3	7	-6	1	17	9
21	9	1	6	-2	-1	23	2
9	-5	-25	-10	-12	-15	-1	-12

a b c

Figura 15 – Operação de convolução com máscara

Inúmeras operações em processamento de imagens são efetuadas a partir do conceito de convolução por máscara. Nos tópicos a seguir, serão apresentados alguns filtros e operadores que utilizam o conceito de convolução.

2.3.4 Operadores e Filtros

Diferentes tipos de filtros são utilizados na segmentação de imagens, e sua aplicação busca resultados diferentes. Dentre os utilizados para a suavização da imagem encontram-se os filtros da média, da mediana e da moda, cuja função é de atenuar variações abruptas nos níveis de cinza da imagem, o que possibilita uma redução de ruído em algumas imagens oriundas de diferentes origens. Nesse trabalho, veremos apenas o filtro da mediana.

Por outro lado, uma grande variedade de técnicas é usada para detectar as bordas de uma imagem, que também fazem uso de filtros conhecidos como máscaras ou *Kernels* (núcleo) de convolução. Uma das técnicas para executar o processo de detecção, consiste de duas convoluções na imagem original. A primeira convolução detecta as bordas no sentido horizontal e a segunda no sentido vertical, formando as bases do subespaço de bordas. A figura 16 demonstra a máscara de tamanho (3 x 3) do operador Prewitt na forma bidimensional.

Operador Prewitt

-1	**-1**	**-1**		**1**	**0**	**-1**
0	**0**	**0**		**1**	**0**	**-1**
1	**1**	**-1**		**1**	**0**	**-1**
Horizontal				**Vertical**		

Figura 16 – Máscara do operador Prewitt

Nesse contexto, os filtros podem ser separados em duas categorias: os filtros da média, da moda e da mediana usados para a suavização de imagens, e os operadores de convolução usados como operadores de gradiente, que realçam e acentuem as regiões de uma imagem com variações abruptas ou significativas em níveis de cinza (filtro para detecção de bordas).

2.3.4.1 Filtro de Mediana

O filtro da mediana, como foi descrito tem a função de remover ruídos de uma imagem. Este filtro é aplicado a cada *pixel* da imagem e cada *pixel* de saída é calculado como uma mediana dos seus *pixels* vizinhos.

Este tipo de filtro preserva os *pixels* de baixa freqüência da imagem, dando o efeito de suavização da mesma. A figura 17 ilustra o *kernel* ou máscara usada pelo filtro de mediana, conforme Felgueiras (2006).

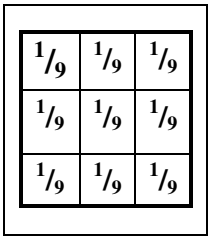


Figura 17 – Máscara para o filtro de Mediana

Ainda existem os filtros Gaussiano e Laplaciano, bastante conhecidos na área de processamento de imagens, onde que o Gaussiano tem o objetivo de remover ruídos, remover detalhes e efetuar a suavização na imagem, semelhante ao filtro de mediana, porém a sua máscara é diferente que permite alterar seu tamanho e valores da mesma.

O filtro laplaciano realça as regiões da imagem onde existe uma diferença brusca da intensidade em nível de cinza, apropriado para a detecção das bordas da imagem. A seguir alguns operadores usados na detecção de bordas.

2.3.4.2 Operador Roberts

O operador Roberts é o algoritmo mais comum para detecção de bordas. Este operador utiliza um par de matrizes 2 x 2 para encontrar os gradientes nas direções x e y da mesma. Para descobrir se o *pixel* avaliado se refere a uma borda, calcula-se o gradiente G através da expressão: $|G| = \sqrt{G_x^2 + G_y^2}$.

Este operador aplica o gradiente na direção de 45º ao invés de calcular a direção de valores dos *pixels* na direção vertical e horizontal, como regularmente fazem outros operadores. A matriz de convolução mostra a máscara utilizada por Roberts:

$$G_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad G_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Para conseguir encontrar a direção do gradiente da borda, sendo que ele é perpendicular à direção da borda deve-se utilizar a seguinte expressão:

$$\alpha = a\tan\left(\frac{G_y}{G_x}\right)$$

Uma desvantagem do operador Roberts é sua assimetria, dependendo da direção, certas bordas são mais realçadas que outras, mesmo tendo igual magnitude, Mascarenhas e Velasco (1994).

2.3.4.3 Operador Sobel

O operador Sobel é mais sofisticado que o operador Roberts, onde cada *pixel* de uma imagem é processado independentemente do resto dos *pixels*, ou seja, esta técnica determina se cada *pixel* presente na imagem pertence ou não a borda, conforme Barreto e Schlemer (2004).

O operador gradiente de Sobel tem a propriedade de realçar linhas verticais e horizontais mais escuras que o fundo, sem realçar pontos isolados. É representado pela seguinte expressão: $G(x,y) = X^2 + Y^2$.

Para que se possa fazer essa determinação, duas máscaras de coeficientes, os quais equivalem às direções X (G_x) e Y (G_y), são utilizadas. Cujos valores são arranjados em uma matriz de convolução de ordem 3 x 3.

$$G_x = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad G_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

A máscara G_x detecta as variações no sentido horizontal e a máscara G_y, no sentido vertical, tornando-se possível à obtenção da magnitude e da direção da borda (seu ângulo) em cada *pixel,* Takeda (2000), como ilustra a figura 18.

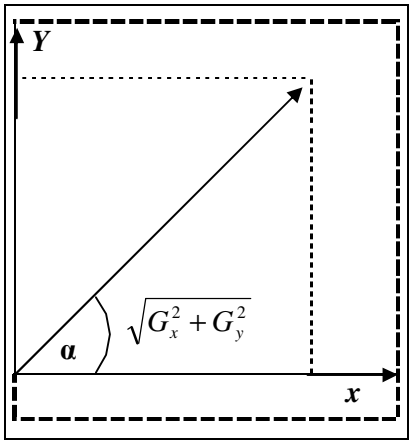


Figura 18 – Magnitude dos gradieantes e ângulo das bordas

A aplicação dos *kernels* no sentido horizontal e vertical proporciona obter o valor da magnitude e a direção das bordas da imagem. A equação $M = \sqrt{G_x^2 + G_y^2}$ é usada para calcular a magnitude de cada *pixel*, onde o valor desta magnitude é atribuído ao respectivo *pixel* na imagem de saída do filtro.

Segundo Fernandez (2004), a figura 19 representa o processo efetuado na aplicação do operador Sobel, onde *dx* representa o gradiente vertical e *dy* representa o gradiente horizontal.

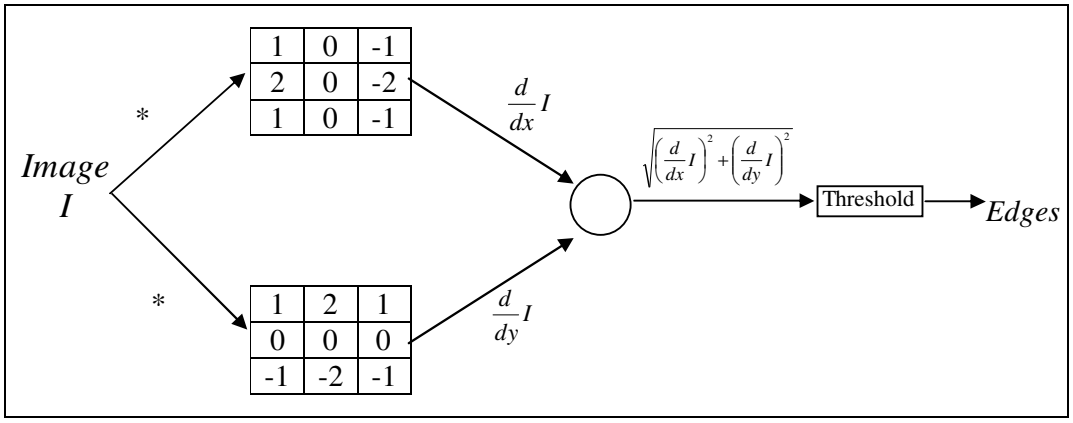


Figura 19 – Processo de aplicação do operador Sobel.

Na forma como as máscaras de Sobel são construídas, baseadas no operador diferencial, apresentam uma resposta nula em regiões homogêneas, regiões onde a derivada é nula. Magnitudes representadas por valores pequenos, indicam que o *pixel* em estudo não pertence a uma borda bem definida (área mais homogênea) segundo Takeda (2000).

Diferentes operadores e filtros de suavização são encontrados na literatura, cuja implementação encontra-se na ferramenta, mas somente os operadores Roberts e Sobel foram descritos nessa dissertação.

2.4 Afinamento de Imagens (*Thinning*)

A técnica de *thinning* ou afinamento é utilizado para reduzir a quantidade de informação na imagem. Ela consiste na operação de média sobre uma imagem quadrada com um limiar (*threshold*) elevado. Muitos autores tratam *thinning* como eixo médio (*medial axis*), esqueletonização ou *thined image*.

Sua aplicabilidade estende-se por diversas áreas, incluindo análise de células e cromossomos, imagens de raio-X, sistemas de visão de autônomos, classificação de impressão digitais, padrões de rachaduras, análise visual de partes industriais, entre outros, França (2001).

A principal característica da técnica de *thinning* é reduzir uma imagem a uma fina linha representativa, o que reduz a quantidade de dados, possibilitando uma análise estrutural simples. Além disso, a redução de uma imagem pode eliminar algumas distorções no contorno enquanto mantém as propriedades geométricas e topológicas. Praticamente, esta redução transforma a imagem a um *pixel* de largura, permitindo que algoritmos de vetorização, usados em tarefas de reconhecimento de padrões, possam trabalhar com os *pixels* da imagem.

Assim, o algoritmo de *thinnning* precisa compactar dados, manter propriedades significativas dos padrões, eliminarem ruídos locais sem introduzir distorções. Tais algoritmos são classificados como seqüenciais ou paralelos e agem excluindo de forma sucessiva as camadas da extremidade, borda, até que um linha ou esqueleto permaneça. A exclusão de um ponto *p* depende dos *pixels* vizinhos e de acordo como estes são examinados, aplica-se um dos tipos de algoritmo.

Nos algoritmos seqüenciais, a exclusão é feita examinando os *pixels* em uma seqüência fixa a cada iteração, e a exclusão de *p* na *n-ésima* iteração depende de todas as operações que tenham sido realizadas até o momento, ou seja, do resultado da *(n-1)-ésima* iteração, assim como os *pixels* já processados na *n-ésima* iteração. Tais algoritmos geram "esqueletos" melhores, observando a conectividade e conservação da topologia, porém

exigem muito tempo de processamento.

De outra forma, nos algoritmos paralelos, a exclusão na *n-ésima* iteração depende apenas dos *pixels* da iteração *n-1*, assim todos os *pixels* podem ser analisados independentemente de forma paralela a cada iteração. Tais Algoritmos podem originar "esqueletos" com falhas, em contra partida consomem pouco tempo de processamento.

A seguir, serão analisados os algoritmos seqüências e o algoritmo paralelo Zhang-Suen, cujo algoritmo foi implementado neste trabalho.

2.4.1 Algoritmos Seqüenciais

Como mencionado anteriormente, uma das maneiras de afinamento da imagem ou o esqueleto de uma região podem ser definidos através da transformação do Eixo Médio (*Medial Axis Transformation - MAT*), proposta por Blum (1976). O Eixo Médio MAT de uma região *R* com borda *B* é definido: para cada ponto *p* pertencente a *R*, encontra-se o vizinho mais próximo em *B*. Se *p* tem mais que um vizinho ele pertence ao eixo médio de *R*.

A implementação do MAT envolve o cálculo da distância de todos os pontos interiores de uma imagem para todos os pontos da borda da imagem, requerendo um grande esforço computacional. Os algoritmos procuram excluir pontos de uma dada região, desde que sua remoção não remova pontos terminais, não interrompa conexões e não cause erosão excessiva na região (borda).

O algoritmo MAT consiste de sucessivas aplicações ao contorno da imagem de duas regras básicas, onde pontos de contornos são quaisquer pontos com o valor 1 e que tenha ao menos um dos oito (8) vizinhos iguais a 0 no exemplo da figura 20. Assume-se que os pontos da imagem (ou borda) possuem valor igual a 1 (um) e que os pontos do fundo valor igual a 0 (zero). A seguir as regras, França (2001), Brillinger (1998):

Regra 1ª: define que um ponto $p1$ é marcado para exclusão se todas as quatro condições a seguir são satisfeitas:

a) $2 \leq NN(p1) \leq 6$;

b) $CRN(p1) = 1$;

c) $p2 ; p4 ; p6 = 0$;

d) $p4 ; p6 ; p8 = 0$;

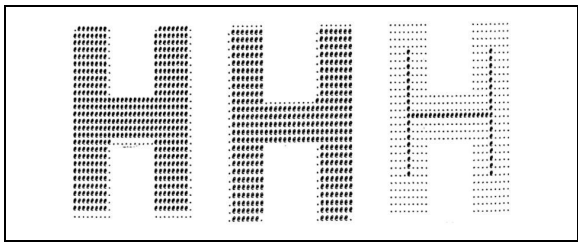
Regra 2ª: as condições (a) e (b) continuam as mesmas, porém a condição (c) e (d) mudam:

(c') p2 ; p4 ; p8 = 0;

(d') p2 ; p6 ; p8 = 0;

Na primeira regra, passo 1 é aplicado a todo ponto da borda da região considerada. Se uma ou mais condições são violadas (a – d), o valor do ponto não é mudado. Do contrário, se todas as condições são satisfeitas, o ponto é marcado para exclusão. Entretanto, os pontos serão excluídos, somente quando todos os pontos da borda forem processados. Após o passo 1 ter sido aplicado em todos os pontos, aqueles que foram marcados são excluídos, seu valor passa a ser zero (0). Em seguida é executado a 2ª regra, ou passo 2. Este é aplicado na imagem da mesma forma que o passo 1.

Tal procedimento é aplicado interativamente até que não exista mais nenhum ponto a ser excluído e, quando terminar a execução do algoritmo, o esqueleto da região (afinamento da região) será originado. A figura 20 mostra o resultado da aplicação do algoritmo.



a b c

Figura 20 – a) Resultado do passo 1 da MAT durante a primeira iteração; b) resultado do passo 2; c) resultado final França (2001) Zhang e Suen (1984).

2.4.2 Algoritmo Paralelo Zhang-Suen

O algoritmo paralelo proposto por Zhang e Suen resulta de operações morfológicas utilizada para remover os *pixels* marcados para remoção em imagens binárias, e está dividido em duas iterações, que resumidas indicam: um ponto *p* é apagado se as seguintes condições são verdadeiras:

1º) Condição para a primeira iteração:

- Conectividade igual a um (1);
- No mínimo existem dois (2) vizinhos pretos e não mais do que seis (6);

- No mínimo um dos pontos *p(i-1,j), p(i,j+1), p(i+1,j)* são brancos;
- No mínimo um dos pontos *p(i,j+1), p(i+1,j), p(i,j-1)* são brancos;

2º) Condição para a segunda iteração:

- Conectividade igual a um (1);
- No mínimo existem dois (2) vizinhos pretos e não mais do que seis (6);
- No mínimo um dos pontos *p(i-1,j), p(i,j+1) e p(i,j-1)* são brancos;
- No mínimo um dos pontos *p(i-1,j), p(i+1,j) e p(i,j-1)* são brancos;

A figura 21 mostra a tabela representando a vizinhança do ponto *p*1 e a representação da condição da primeira iteração para a remoção dos pontos.

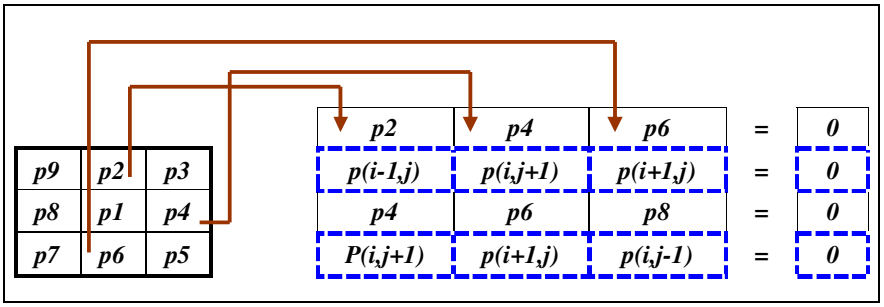


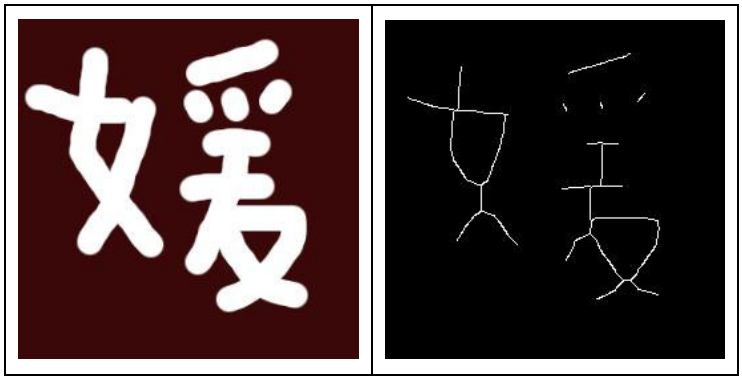
Figura 21 – Representação da remoção dos pontos para a primeira iteração do algoritmo de Zhang e Suen.

Portanto, na primeira execução (iteração 01) serão excluídos 4 *pixels* das bordas sul, leste e noroeste e, na segunda execução (iteração 02) serão excluídos os *pixels* nas posições opostas, norte, oeste e sudeste.

Um dos problemas encontrados nesse algoritmo são os quadrados 2x2, que são eliminados do "esqueleto". Para evitar isso, é necessário mudar a primeira condição para $3 \leq NN(p1) \leq 6$, porém essa solução gera outro problema, deixando de eliminar *pixels* que deveriam ser excluídos.

O algoritmo paralelo de Zhang e Suen teve seus estudos iniciais por volta de 1984, cujo trabalho obteve resultados superiores aos outros de mesma época. Entretanto muitas melhorias propostas por pesquisadores e pelo próprio Zhang melhoraram ainda mais o algoritmo, seus resultados são comparados ainda hoje, segundo França (2001).

A figura 22 mostra a aplicação do algoritmo paralelo de Zhang e Suen (1984) adaptado por Cheok Yan Cheng. Cujos trabalhos relacionados podem ser encontrados em Cheng (2004).



a b

Figura 22 – Aplicação do algoritmo paralelo de afinamento de Zhang e Suen, a) figura original, b) após aplicação do algoritmo.

Para que os resultados dos algoritmos de esqueletonização ou afinamento sejam satisfatórios são necessários que a exclusão de forma sucessiva das diversas camadas das extremidades (*pixels* das bordas) de um padrão seja efetuada até que apenas um esqueleto permaneça.

2.4.3 Algoritmo de Afinamento de Holt

O algoritmo de Holt não envolve nenhuma iteração, como usado no algoritmo de Zhang e Suen. No lugar delas, são usadas expressões lógicas numa vizinhança 3 x 3 sobre o ponto em questão. Conforme Facon (2001), a primeira iteração pode ser substituída pela expressão lógica:

$Vp(P) \wedge (\sim borda(P) \vee (Vp(L) \wedge Vp(S) \wedge (Vp(N) \vee Vp(O))))$ e,

A segunda iteração pode ser substituída por:

$Vp(P) \wedge (\sim borda(P) \vee (Vp(O) \wedge Vp(N) \wedge (Vp(S) \vee Vp(L))))$.

Cuja aplicação no ponto *P* corresponde:

- Se o resultado das expressões for falso, o ponto *P* é removido, do contrário ele não é removido;
- A função Vp(), representa o valor do ponto *P*. Se *P* corresponde a um objeto preto seu resultado é verdadeiro, se for branco o resultado é falso.
- A função borda(), corresponde a verdadeiro se o ponto *P* estiver na borda do objeto, e falso, caso contrário;

- As letras N, S, L, O, (Norte, Sul, Leste, Oeste), correspondem os pontos vizinhos de *P*. A figura 23 mostra o esquema de vizinhança usada para o algoritmo de Holt, semelhante ao algoritmo de Zhang e Suen.

NO	***N***	***NE***
O	***P***	***L***
SO	***S***	***SE***

Figura 23 – Representação da vizinhança para o algoritmo de Holt

Conforme Facon (2001), Holt combinou as duas expressões lógicas descritas acima e obteve uma nova expressão, como segue:

$$Vp(P) \wedge (\sim borda(P) \vee (borda(L) \wedge Vp(N) \wedge Vp(S)) \vee$$

$$\vee (borda(S) \wedge Vp(O) \wedge Vp(L)) \vee (borda(L) \wedge borda(SE) \wedge borda(S)))$$

Essa nova expressão também foi melhorada por (FACON, 2001), que criou a técnica de remoção de escada (*staircase removal*). Esta técnica consiste da remoção de metade dos pontos cuja forma é semelhante a uma escada e sua remoção não afetam o formato e a conectividade do objeto. A figura 24 mostra o resultado da aplicação do algoritmo normal de Holt e a aplicação de Holt melhorado com a técnica de remoção de escada.

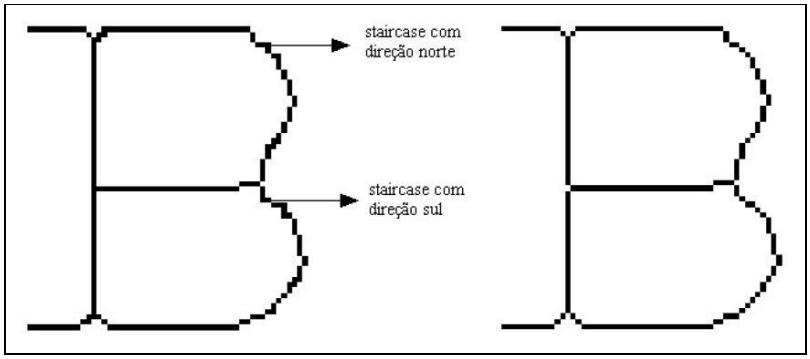


Figura 24 – Resultado da aplicação do algoritmo de Holt, à esquerda, Algoritmo de Holt com "staircase removal", e a direita, o Algoritmo de Holt, Facon (2001).

Na literatura são encontrados diferentes tipos de algoritmos de afinamento da imagem. Os algoritmos de Zhang e Suen, e Holt vistos anteriormente são algoritmos de execução rápida, fácil de implementação e com bons resultados. Comparando-os observa-se que o

algoritmo de Holt com “remoção de escada” supera o resultado do algoritmo de Zhang e Suen sendo mais rápido e simples de implementar.

2.5 Manufatura Integrada por Computador

As ferramentas CAE/CAD/CAM formam o conjunto de tecnologias computacionais para automação de todo o sistema de manufatura, denominado CIM (Manufatura Integrada por Computador). A utilização dos sistemas CIM visa à automação de todos os procedimentos e etapas envolvidas na manufatura, através de sistemas elétricos, eletrônicos e computacionais, ou seja, são usados na modelagem geométrica do produto e durante todos os estágios da produção para confecção do produto final.

Conforme Oliveira e Rubio (2005), os sistemas CIM envolvem o uso de uma série de tecnologias que trazem ferramentas de auxílio às atividades de planejamento e implementação do sistema de produção, são elas:

- **CAPP** - *Computer Aided Process Planning* - Planejamento do Processo Assistido por Computador;
- **CAE** - *Computer Aided Engineering* - Engenharia Assistida por Computador;
- **CAD** - *Computer Aided Design* - Projeto Assistido por Computador;
- **CAM** - *Computer Aided Manufacturing* - Manufatura Assistida por Computador.

Assim, o CIM integra as informações contidas nos projetos CAE permitindo que sejam entendidas pelos sistemas CAD, que deve repassar os desenhos para o CAPP, que irá comunicar seus planos com o sistema CAM, para serem executados.

Na Engenharia Mecânica, o uso de imagens cresce principalmente em projetos de manufatura (sistemas CAD/CAM) e em aplicações de equipamentos tipo Comando Numérico Computadorizado (CNC). Com isso, muitas metodologias são desenvolvidas para geração automática de programas CNC que usam partes de uma imagem, presentes inicialmente em sistemas CAD conforme Silva et. al. (2005).

2.5.1 Comando Numérico, Programação em Máquina Ferramenta

Conforme Costa e Pereira (2006), em processos de usinagem, a tecnologia CNC pode ser interpretada como um sistema de informação que transforma uma descrição geométrica do componente a ser usinado, a partir de uma entrada simbólica da mesma no controle de posições e velocidade de um ou mais servomotores. Isto compreende o comando de movimento de aproximações, avanços e recuos de uma ferramenta de corte.

O Controle Numérico Computadorizado (CNC) é um equipamento apropriado para realização de processos de usinagem. Geralmente, usa-se um equipamento eletrônico, como um computador dedicado, que recebe informações da forma em que a máquina vai realizar uma operação por meio de linguagem própria, denominado programa CN. O CN processa as informações e aciona os motores da máquina ferramenta através de impulsos elétricos para executar os movimentos desejados com todas as características de usinagem, realizando a operação na seqüência programada sem a intervenção do operador.

Conforme Pereira (2003), a tecnologia CN e CNC basicamente executam a mesma tarefa, ou seja, trabalham com a manipulação de dados com a finalidade de usinar uma peça. Entretanto a diferença encontra-se na seguinte denominação:

O sistema CN utiliza funções lógicas fixas, construídas dentro de uma unidade de comando e não podem ser mudadas pelo programador ou operador da máquina. Assim o sistema pode interpretar o programa, mas não pode efetuar nenhuma mudança ou alteração nele para satisfazer uma condição diferente daquelas pré-definidas na unidade de comando.

Diferentemente, a tecnologia CNC, utiliza um microprocessador interno que é capaz de armazenar em seus registros de memória as variáveis de rotinas que manipula funções lógicas. Isto permite que o programador ou operador possa modificar o programa na própria máquina cujos resultados podem ser percebidos instantaneamente. Esta flexibilidade é o ponto chave que traz vantagens significativas e contribui para o aumento do uso dessa tecnologia nos processos de manufatura.

2.5.2 Projeto Assistido por Computador (CAD)

Os sistemas CAD permitem modelar produtos e componentes, detalhar suas formas, tanto em duas 2D, quanto em 3D, auxiliando nos projetos de engenharia. Eles facilitam a criação e manipulação de informações de um projeto, sistematizando os dados envolvidos e possibilitando a rápida reutilização das informações.

A modelagem de uma peça, por exemplo, pode ser efetuada utilizando diferentes dispositivos de entrada, como: mouse, teclado, caneta luminosa, scanner, digitalizadores tridimensionais, entre outros. Bem como a aquisição de imagem digital, através de máquinas fotográficas, como objeto de estudo desse trabalho. A figura 25 mostra o uso de digitalizadores tridimensionais.

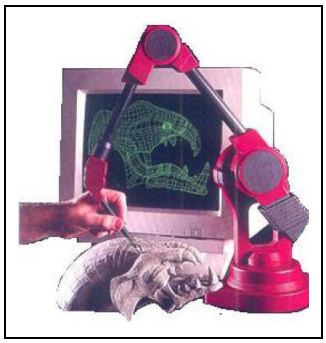


Figura 25 – Digitalizador tridimensional (Fonte: Techno Isel e Maxnc - USA). Oliveira e Rubio (2005).

Os Digitalizadores Tridimensionais são scanners 3D, com capacidade de obter às coordenadas tridimensionais de um objeto sólido real. A aquisição dos dados pode ser feita num modelo reduzido ou até mesmo em tamanho normal. A partir dos dados obtidos pelos digitalizadores é possível trabalhar as formas do sólido diretamente num software de CAD 3D, Oliveira e Rubio (2005).

O uso de digitalizadores tridimensionais e a aquisição da imagem através de fotografia digital fazem referência às técnicas de engenharia reversa.

Esta tecnologia consiste na criação de um modelo tridimensional, ou bidimensional, no computador a partir de um objeto do mundo real já existente. Além disso, a engenharia reversa permite a geração de desenhos, alterações e acréscimo de formas em objetos de modelagem, como em peças que saem defeituosas de moldes de fundição usados em projetos de produção.

2.5.3 Engenharia Reversa

A técnica de engenharia reversa no contexto dessa dissertação consiste na criação de um novo modelo de peça a partir de um objeto do mundo real já existente. Assim, esta técnica pode ser resumida em seis passos:

- Ter o objeto disponível para modelagem;
- Capturar o objeto através de câmaras digitais;
- Gerar as coordenadas do objeto através de técnicas de processamento de imagem;
- Envio dos pontos ou importação das coordenadas ao CAD;
- O CAD reúne os pontos e cria linhas para serem utilizadas na elaboração de superfícies do objeto, ou seja, calcula-se a trajetória dos pontos para criação das linhas e geração do código CNC;
- Envio do código ao CAM para serem executados em ações da máquina-ferramenta.

Um processo de engenharia reversa é adequado quando ocorre a seguinte situação: determinada peça de uma máquina falha e não se possui outra disponível para reposição imediata. Então, a peça quebrada é entregue a um operador de máquina-ferramenta para reproduzir a nova peça. Para fabricar esta peça, o operador deve efetuar a medição da mesma, a fim de fazer um levantamento dimensional dela e descobrir suas características geométricas. Desenhar a peça ou fazer um esboço servindo como base para realizar as operações de usinagem da mesma. O processo de reprodução da peça, através de medições físicas para criar dados técnicos da mesma, de maneira que se possa copiá-la, é denominado de engenharia reversa, segundo Ferneda (1999).

Conforme Chikofsky e Cross (1990), o termo “engenharia reversa” teve origem na análise de hardware por ser comum criar novos projetos a partir de produtos finais. Em geral, o objetivo da engenharia reversa na área de hardware é duplicar o sistema. Na área de software seu objetivo é obter entendimento suficiente do sistema e sua estrutura para apoiar a sua manutenção, sua melhoria ou a sua substituição, ou seja, criar visões do sistema em diferentes níveis de abstração a fim de facilitar o seu entendimento e manutenção.

Dessa forma, diversas abordagens sobre diferentes definições contemplam as atividades de engenharia reversa. Algumas delas são descritas a seguir:

- Engenharia reversa é o processo de analisar um software, criando representações do mesmo em níveis mais altos de abstração, cuja produção de informações possa aumentar o conhecimento geral do mesmo. Pressman (2002);
- Engenharia reversa é o processo do exame e compreensão do sistema existente, para recapturar ou recriar os requisitos atualmente implementados pelo sistema, apresentando-os em um nível mais alto de abstração. Costa (1997);
- A engenharia de reserva serve para entender um sistema de software com o objetivo de facilitar atividades como: expansão, correção, documentação, re-projeto ou re-programação em outra linguagem de programação. Rugaber (1992).
- A engenharia reversa consiste em produzir novas peças, produtos ou ferramentas a partir de modelos ou componentes existentes. Dickin (1996).
- A engenharia reversa é o processo de levantar dimensões, com rapidez e exatidão, determinar padrões geométricos tais como áreas e volumes além de definir as tolerâncias de um modelo existente. Daschbach (1995).

A aplicação da engenharia reversa é bastante utilizada na área industrial. Devido a grande competitividade entre empresas metalúrgicas, a busca pela qualidade e baixo custo dos preços na fabricação de produtos, esta técnica é usada, pois permite criar novos produtos, copiar modelos existentes e corrigir ou melhorar esses modelos de maneira rápida e com baixo custo.

Conforme Lima (2003), a engenharia reversa aplicada à indústria esta relacionada aos seguintes processos de fabricação:

1. Criação de um novo produto: É comum na indústria, a elaboração de projetos ou modelos físicos baseados nas funcionalidades e necessidades que necessitam serem implementadas. A criação de um novo produto ocorre quando a elaboração de um modelo inicia-se a partir de um objeto existente. Assim, criando-se algumas partes complexas, pode-se construir um modelo CAD através do método da Engenharia Reversa.

2. Cópia de um modelo existente: Em alguns casos, não existem desenhos ou quaisquer informações sobre um modelo ou peça metalúrgica. Como exemplo, na indústria automobilística, algumas peças de carros tiveram seus ferramentais construídos artesanalmente (sem nenhuma documentação) e continuam produzindo peças até hoje. A Engenharia Reversa pode copiar todas as características do molde, auxiliando na confecção do novo modelo.

3. Correção de um modelo danificado: Parte-se de um modelo ou peça está danificado, a reconstrução deste não deve conter seus erros. Usando ainda o exemplo dos ferramentais antigos, com o passar do tempo eles sofrem desgastes ou até mesmo quebras. O emprego da Engenharia Reversa facilita sobremaneira a confecção do ferramental substituto.

4. Melhorias das formas de um modelo: O conceito de um produto baseado em funcionalidades e aspectos estéticos pode ser finalizado utilizando materiais leves como madeira ou resinas para fabricá-los. Neste processo, o projetista não precisa desperdiçar tempo em criar o modelo com alta precisão, pois este pode ser melhorado na criação do modelo CAD. Contudo, os sistemas de Engenharia Reversa devem ser capazes de deduzir algumas características como simetria, paralelismo e perpendicularidade.

Observa-se que o processo de Engenharia Reversa caracteriza-se pela reprodução de um modelo físico, para que este possa transformar-se em um modelo digital. No processo convencional de engenharia, cria-se primeiramente o modelo virtual, para que se possa então, confeccionar produtos correspondentes ao modelo físico. Dessa forma, na engenharia reversa, o processo ocorre de trás para frente, ou seja, o modelo físico já existe e necessita-se do modelo virtual para que as etapas da engenharia possam ser formuladas e pode ser utilizado em diversas aplicações.

Nesse contexto, a engenharia reversa é aplica em diferentes áreas do conhecimento. Aplicada ao software, permite uma representação do mesmo em um nível mais alto de abstração, buscando um melhor entendimento de suas funcionalidades para futuras alterações. Aplicado a uma peça metalúrgica, possibilita identificar suas formas geométricas e reprodução da mesma. No caso desse trabalho, o objetivo é utilizar a engenharia reversa para extrair as características do objeto (peça) através do processamento de imagens e transformá-las em dados para serem usados em aplicativos como o CAD.

2.5.4 Manufatura Assistida por Computador (CAM)

A tecnologia CAM envolve a programação de máquinas de comando numérico (CNC) com o auxilio do computador, e apresenta o conjunto chamado CAD/CAM que representa os módulos de programação CNC em sistemas CAM.

O programa CNC é escrito em uma sentença de comandos, onde cada comando é

composto de palavras e cada qual tem seu endereço de letras e valores numéricos. Tais comandos são utilizados para o cálculo do caminho da ferramenta. Os conjuntos de comandos criam uma sentença de palavras, que informam a máquina CNC o que se deseja fazer com o bloco de comandos, ou seja, a sentença de comandos indica o que fazer para construir determinada peça.

Usualmente o computador representa à unidade controladora, que tem a função de ler e interpretar o programa de instruções e convertê-lo em ações da máquina-ferramenta. Entre máquinas-ferramentas que utilizam à tecnologia CAD/CAM temos: tornos, fresadoras, furadeiras, cortadeiras, centros de usinagem, retíficas, entre outras.

Existem diferentes métodos de programação de máquinas CNC auxiliado pelos sistemas CAD/CAM, o qual consiste em controlar a máquina-ferramenta com as informações pré-programadas ou codificadas. Desde a criação (desenho) da peça pelo CAD, ou a importação das entidades geométricas para posteriormente ser enviado à máquina-ferramenta.

2.5.5 Ferramentas de Processamento de Imagens e Geração de Programas CNC

A tecnologia CAD/CAM integra a modelagem do produto, sendo responsável pela prototipação de novos materiais, pois permite o detalhamento de suas formas tanto em duas quanto em três dimensões.

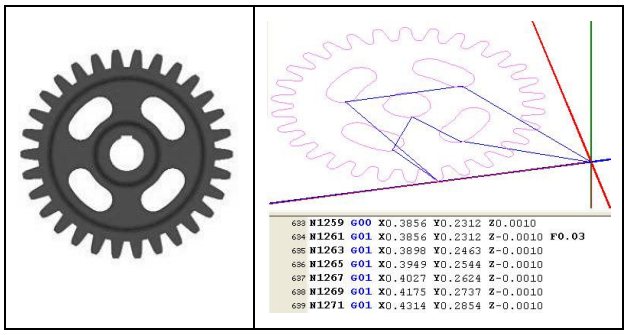
Na tentativa de automatizar determinados processos de fabricação, ferramentas para geração automática de código propõem modelos para geração do programas CNC a partir de imagens digitais. Dentre as quais, a aplicação: “Processamento de imagens aplicado à programação CNC em processos de furação”, apresenta a implementação de uma ferramenta para segmentação de imagens desenvolvida em linguagem IDL (*Interactive Data Language – IDL® 6.0 – Resersh Systems Inc.*), conforme Silva et. al. (2005). Tais imagens são peças de veículos, máquinas agrícolas, entre outras, que necessitam de processos de furação em sua fabricação.

Nesse contexto, alguns trabalhos envolvendo a aplicação de imagens e a geração de código CNC podem ser encontrados na literatura. Exemplo disso, são os softwares **TracTrix™** (Trix, 2008), **Vextractor®** (VextraSoft, 2007), **DeskCNC®** e **Scan2CAD®**, sendo estes dois últimos descritos a seguir.

O *DeskCNC* (IMService, 2003) é um software baseado em CAM para Windows que

possibilita criar e simular programas CNC para processos de furação, usinagem 3D, entre outros. O módulo "*CAM Software"* permite digitalizar imagens 2D e 3D, abrir imagens de diferentes formatos (JPEG, GIF, BITMAP, TIFF, entre outros), carregar um arquivo CN, possui diversos filtros e funções de modificação, como equalização, contraste, detecção de contornos e toda manipulação da imagem até a gravação do vetor *raster* e conversão diretamente para o código CNC.

A figura 26 ilustra um exemplo da aplicação do *DeskCNC* na sua versão *Trial*, com visualização 3D, e abaixo a geração do código CNC para a peça de exemplo.



a b

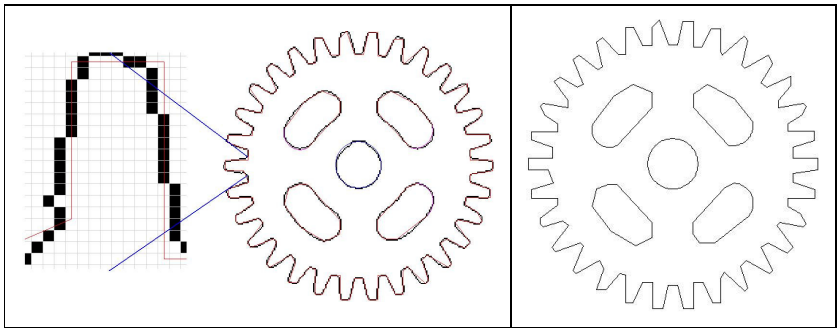
Figura 26 – a) Imagem Original; b) Contornos da imagem no editor CNC com seu código gerado abaixo.

O software *DeskCNC* está dividido em três módulos. O módulo CAM, que permite carregar e manipular a imagem, representado pela figura 26 item (a), o módulo editor CNC, que permite visualizar a imagem em 3D e gerar ou criar programas CNC, representado pela figura 26 item (b) e o Módulo que fornece mecanismo para enviar o programa para a máquina ferramenta.

Outro software, o *Scan2CAD* permite converter arquivos digitalizados ou imagens de diferentes formatos, em arquivos próprios para sistemas CAD, além de salvar as imagens nos formatos de arquivos de imagens padronizados e arquivos para visualizações.

Assim, com o *Scan2CAD*, pode-se carregar uma imagem, efetuar o processamento, como conversão em nível de cinza, geração do contorno ou detecção das bordas, vetorização da imagem e exportação do vetor (imagem) para o formato *dxf* do CAD.

Para exemplificar o funcionamento do *Scan2CAD*, foi efetuado processamento na imagem através de *threshold* e detecção de contornos, depois executado a vetorização e exportação para o Autocad. A figura 27 ilustra um exemplo do uso do *Scan2CAD*.



a b

Figura 27 – a) imagem segmentada e vetorizada pelo *Scan2CAD*, b) imagem resultante exportada no formato *dxf* e visualizada no CAD.

A vetorização da imagem não preserva o formato da mesma, isto é observado na figura 27 item (b), onde os contornos da imagem não são arredondados, porém é uma das opções disponíveis no *Scan2CAD* para conversão de uma imagem em arquivo reconhecido pelo CAD.

Entretanto, a tecnologia CAD/CAM possui uma vasta área de aplicação, onde autores elaboraram trabalhos que hoje contribuem para o avanço da tecnologia CNC, um desses trabalhos pode ser visto em Álvares et. al. (2005), cujo assunto é baseado na modelagem de peças pela Internet através da tele-operação de equipamentos CNC, buscando uma metodologia de integração remota de manufatura.

2.6 Técnicas de Detecção de Bordas *Multi-Flash*

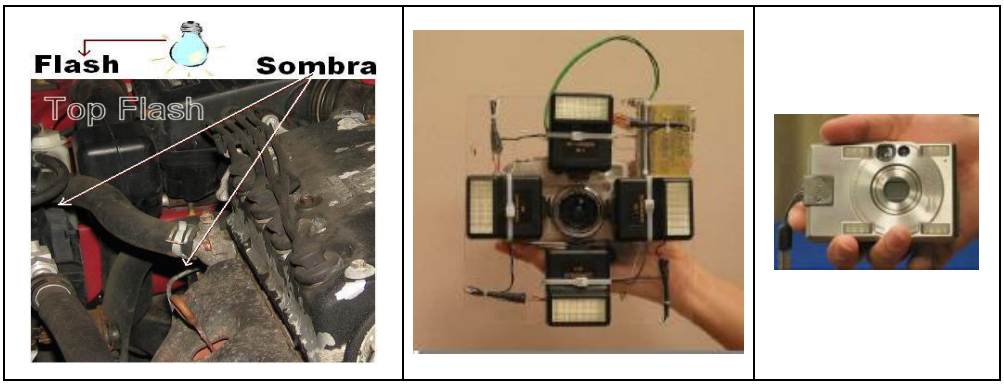
Para conseguir bons resultados no processamento de imagens é muito importante a forma de aquisição da mesma. Dessa forma, duas maneiras de aquisição foram empregadas. A primeira trata da aquisição de uma imagem (foto da peça) através de uma máquina fotográfica digital de maneira convencional, observando a iluminação, para não ocorrer à incidência de sombras inibindo distorções e ruídos no processamento aplicado posteriormente.

A segunda maneira é a aplicação da técnica de *Multi-flash* ou Detecção de Bordas de Profundidade usando Imagens *Multi-flash*, proposta por Ramesh et al (2004). Esta técnica é empregada na aquisição da imagem quando o equipamento fotográfico aplica quatro (4) *flashes* na mesma, uma na parte superior, outra no lado direito, outra na parte inferior e a última no lado esquerdo, respectivamente gerando quatro (4) imagens diferentes com realce

ou sombras em suas extremidades (bordas). O algoritmo utilizado por Ramesh gera como resultado três (3) imagens, uma ambiente, outra com o mapa de profundidade e a última com as bordas detectadas.

Para adquirir as imagens, Ramesh utilizou uma câmara com vários *flashes* estrategicamente posicionados para criar as sombras ao longo das descontinuidades da imagem, ou seja, o limite entre a imagem de interesse e a parte de fundo da cena.

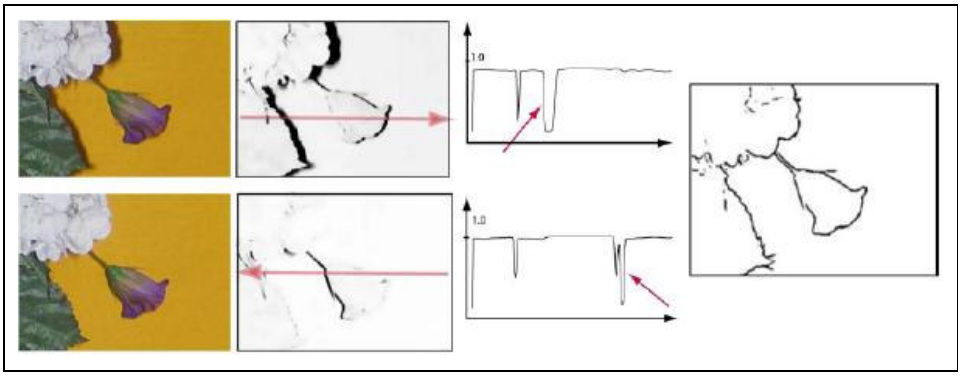
A figura 28 item (a) ilustra a aplicação do *flash* no lado direito da imagem proposto por Ramesh, e a figura 28 item (b) a câmara fotográfica com 4 *flashes*.



a b

Figura 28 – a) Aplicação do flash, b) Câmara *multi-flash* por Ramesh et al (2004).

Conforme observado a figura 28 item (a), exibe a aplicação do *flash* que ilumina a cena no lado superior da imagem para gerar sombras nas descontinuidades da mesma. Em (b), a câmara *Multi-flash* para gerar as quatro imagens. Para obter o resultado, o método consiste na observação da geometria *epipolar* da sombra. A idéia de geometria *epipolar*, por exemplo, consiste na utilização de duas imagens correspondentes cuja visão é diferente do objeto, ou seja, diferentes posições da câmara. Assim, dado duas imagens A e B, a geometria *epipolar* usa informações de projeção da câmara para definir dado um ponto na imagem A, uma região onde o ponto correspondente na imagem B pode ser encontrado, sendo esta região uma reta. A figura 29 ilustra o exemplo da geometria *epipolar* aplicado na detecção de bordas.



a b c d

Figura 29 – Detectando bordas de profundidade a) Imagem de cima flash da esquerda, abaixo flash da direita, b) Setas para verificar ponto *epipolar*, c) Raio ou ponto *epipolar*, cujas setas indicam transição negativa, d) Bordas detectadas.

A figura 29 item (a) exibe as imagens originais, onde na imagem superior foi aplicado o *flash* da esquerda (observa-se que gerou sombras ao lado da imagem), na imagem inferior foi aplicado o flash na direita. Em (b), os valores onde as áreas são iluminadas pelo *flash* são próximos de um (1.0), e próximo de zero (0) nas regiões de sombra. Em (c), é mostrado a transição negativa, juntamente com o raio *epipolar*, ponto correspondente nas duas imagens, onde caracteriza a borda de profundidade e (d) o resultado da detecção Ramesh et al (2004).

Conforme Martinez (2005), um algoritmo básico para implementar a detecção de bordas de profundidade foi desenvolvido usando o Matlab, cuja intensidade dos *pixels* foi calculada usando o operador gradiente *Sobel*, porém pode ser utilizado qualquer um dos operadores, *Prewitt*, *Roberts*, entre outros. Tal algoritmo possui as seguintes etapas:

1 – Capturar uma imagem ambiente denominada $\mathbf{I_0}$;

2 – Capturar **n** imagens com fonte luminosa ou *flash* em $\mathbf{I_{k+}}$ para **n** = 1, 2, 3, 4;

3 – Processar $\mathbf{I_k = I_{k+} - I_0}$;

4 – Calcular a intensidade máxima da imagem. Para todos *pixel* de **x**, $\mathbf{I_{max} = max_k (I_k(x))}$ onde **k** = 1, 2, 3, 4;

5 – Para cada *flash* de **k**, calcular a parte da imagem $\mathbf{R_k}$. $\mathbf{R_k (x) = I_k(x) / I_{max}(x)}$;

6 – Para cada parte da imagem $\mathbf{R_k}$, onde cruzam (atravessam) raios *epipolares* do raio *epipolar* $\mathbf{e_k}$, localizar os *pixels* **y** marcados com transição negativa e marcá-los (*pixels* **y**) como *pixels* da borda de profundidade.

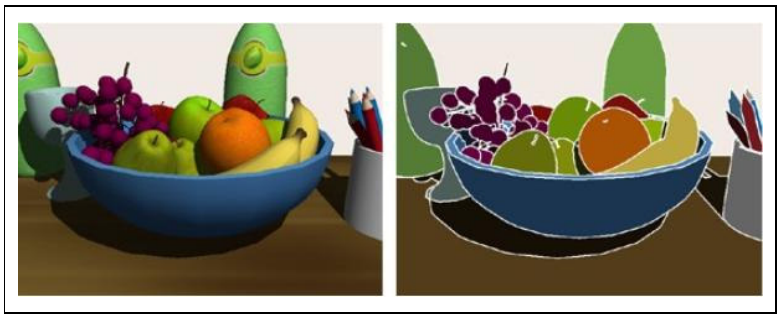
As aplicações desses métodos podem ser usadas em técnicas de ilustrações, desenho animado, *cartoons* e na geração de imagens parecidas como as desenhadas à mão. Facilita a geração de imagens com aparência de *cartoons* a partir de uma imagem real, ou a partir de

uma imagem tridimensional, segundo Marroquim (2007).

O algoritmo proposto por Ramesh Raskar, consome um pouco de processamento computacional, porém é de fácil implementação que permite classificar bordas de profundidade e bordas normais, explorando a relação *epipolar* entre luzes e sombras para extrair as características geométricas das imagens na cena.

Outros autores desenvolveram trabalhos na detecção de silhuetas em imagens, conforme Jardim (2007). Na detecção da silhueta, definem-se as curvas da imagem possibilitando a interpretação da geometria da figura, onde muitos estudos estão sendo realizados, principalmente na área de *Non-photorealistic Rendering*, segundo Ramesh et. Al (2004) e Hertzmann (1999), cuja aplicação pode ser dividida em dois grupos: O artístico, através de estilos de linha, *cartoon* e animação. E o ilustrativo, através de peças, plantas, mapas e ilustrações médicas.

Ainda na área de *Non-photorealistic Rendering*, pesquisas exploram a segmentação baseada em modelos 3D, que pode ser encontrado em Kolliopoloulos et al (2006). A partir de uma pintura, é efetuado o particionamento da mesma, ou seja, a imagem é dividida em regiões permitindo separar os objetos da cena, ideal para a detecção de contornos. A figura 30 ilustra o trabalho realizado, onde em (a) é mostrado a imagem 3D da cena e em (b) é mostrada a imagem segmentada através da *renderização* da cena.



a b

Figura 30 – a) imagem original, b) imagem segmentada Kolliopoulos et al (2006).

Dessa forma, observam-se pesquisas na segmentação de imagem, detecção de silhuetas, *Non-photorealistic Rendering,* estas aplicadas em imagens 2D e 3D, sendo efetuadas por diferentes autores na área de processamento de imagens.

A seguir, o terceiro capítulo aborda a metodologia empregada na implementação desse trabalho.

3 METODOLOGIA

Na implementação do protótipo *JImage Automation*, diversas tecnologias foram utilizadas, além da preparação do ambiente para aquisição das imagens, tanto de uma única imagem, método convencional, quanto às usadas pela técnica *multi-flash*.

Após o processamento da imagem, é efetuada a importação do arquivo texto contendo as coordenadas da peça, para o cálculo da trajetória da ferramenta e geração do código para máquinas CNC. Neste processo é utilizada a linguagem LISP. O sistema CAM terá a função de ler e interpretar o programa de instrução e converter em ações da máquina-ferramenta.

A seguir, serão descritos os materiais, ferramentas e métodos utilizados na elaboração desse trabalho.

3.1 Linguagem de Programação Java e Biblioteca JAI

A linguagem de programação adotada para o desenvolvimento do *JImage Automation* foi o Java, por ser uma linguagem independente de plataforma e principalmente por possuir a biblioteca JAI, a qual provê suporte ao processamento de imagens complexas. Entretanto uma interface foi criada através do ambiente Eclipse™ 3.3.0, juntamente com o compilador Java JDK 1.6.0.03 e a biblioteca de processamento de imagens JAI 1.1.2.01 – *Java Advanced Image*.

O Java possui o API Swing, uma biblioteca que fornece um conjunto de classes para o desenvolvimento de interfaces gráficas. Essas classes são componentes de alto nível que possibilitam uma melhor compatibilidade entre os sistemas onde o Java está sendo executado. Nas primeiras versões, o Java possuía a biblioteca AWT (*Abstract Windows Toolkit*) como ferramenta de programação gráfica, com alguns problemas de compatibilidade. Com o surgimento do Swing esse problema foi solucionado, já que o próprio Swing procura desenhar todos os componentes gráficos na aplicação tirando essa tarefa do sistema operacional, como ocorria com a AWT. Também resolveu a falta de portabilidade existente na AWT (GUJ, 2006).

Entre diversas operações, Swing possui suporte a Java-2D, que são operações gráficas

2D, imageamento e impressão, transformações de coordenadas, conversão de modelos de cores, operadores de exibição de imagens. Ainda, possui componentes gráficos para a construção de GUIs como: *JFrame, JPanel, JMenu, JButton, JRadioButton, JCheckBox, JRAdioButton, JFileChooser*, e muitos outros.

Assim, a biblioteca Swing é voltada para aplicações genéricas e construção de interfaces gráficas ao usuário, permitindo a construção de botões, menus, gerenciamento de janelas e tratamento de eventos. Porém, para executar as transformações em determinadas imagens é utilizada a biblioteca JAI.

A API ou biblioteca JAI – *Java Advanced Imaging*, é uma biblioteca padrão para o processamento de imagens, composta por um conjunto de classes Java que permitem realizar operações em imagens digitais. Além disso, a biblioteca JAI contém um conjunto de rotinas que permitem o rápido desenvolvimento de aplicações voltadas para o processamento de imagens com eficiência e alta performance.

A JAI, por ser baseada em Java, é uma biblioteca flexível, extensível, eficiente, orientada a objetos e possui independência de plataforma. Suas operações variam desde a aplicação de convoluções, criação de histogramas da imagem, inversão de cores da imagem, detecção de bordas com o operador de gradiente, entre outras. Tais operações já estão definidas na JAI e prontas para serem usadas (SUN 1999).

Conforme Jankowski (2002), a JAI possui aproximadamente 80 operadores com suporte nativo e performance otimizada para o processamento de imagens. Em Sun (1999), podem-se encontrar tais operadores separados por categoria, que incluem: a categoria de operadores geométricos, de área, pontuais, estatísticos, de arquivos, categoria de operadores de extração de bordas, entre outros.

Sendo assim, JAI é uma poderosa biblioteca que possui suporte ao tratamento de imagens complexas desenvolvida para programadores usarem ou adicionarem rotinas facilmente em suas aplicações de processamento de imagens.

3.2 Matlab®

Segundo Zubelli et al (2000), o Matlab® vem de *Matrix Laboratory*, foi desenvolvido originalmente para tratamento de vetores e matrizes. Atualmente o Matlab® possui uma biblioteca que abrange funções matemáticas com geração de gráficos e manipulação de dados.

Além disso, possui uma vasta coleção de bibliotecas denominadas *toolboxes*, tais como: equações diferenciais, estatística, finanças, processamento de sinais e processamento de imagens, esta última é a parte relevante nesta dissertação.

Para executar as funções desenvolvidas por Ramesh, é necessária à instalação do Matlab®. Neste caso, foi instalado o Matlab® versão 7.0.0.19920 (R14). Porém não há necessidade de nenhuma configuração especial ou execução do Matlab®, somente para que a ferramenta *JImage Automation* execute as funções Matlab®, as imagens e as funções deverão estar no caminho ou diretório de trabalho (*workspace*) do Matlab®. Assim a ferramenta irá ativar o Matlab® estabelecendo uma conexão com o mesmo, chamar as funções Matlab® que serão executadas e retornarão a imagem resultado. As imagens deverão ser nomeadas como: *flashSuperior.jpg*, *flashDireito.jpg*, *flashInferior.jpg* e *flashEsquerdo.jpg*, pois a função desenvolvida esta preparada para receber as imagens com essa denominação.

Assim, ao utilizar dois ambientes diferentes, necessita-se de algum tipo de ferramenta que faça a integração entre eles. Esta integração é possível com a biblioteca JMatlink 1.3.0, cujos métodos permitem a conexão do Matlab® com a plataforma Java.

3.3 Integrando Matlab e Java

Uma característica importante do Matlab® é a possibilidade que o ambiente e a linguagem de programação proporcionam ao permitir que o usuário crie suas próprias funções, como é o caso da função *FindDepthEdges.m* e *thresholding.m*, que efetuam a detecção de bordas de profundidade implementadas por Ramesh et al (2004) e adaptada para serem usadas nessa dissertação.

No Matlab®, encontram-se tipos de dados equivalentes aos da linguagem C, como *char* (string), *array* (vetores, matrizes) e *structs*, onde informações adicionais podem ser encontradas no *Help* , arquivo de ajuda do Matlab®. Além disso, programas escritos em C e Fortran podem ser chamados dentro do ambiente Matlab®, como os comandos nativos, tais arquivos são denominados *mx* e *mex*.

Esta facilidade, de incorporar recursos de diferentes linguagens, também está presente no Java, o que permite criar métodos Java invocando funções Matlab®, ou a partir deste incorporar recursos do Java, ou seja, possui interface para criar classes e objetos escritos em Java e chamar métodos associados a esses objetos, segundo Zubelli et al (2000).

A aplicação da técnica proposta por Ramesh está implementada neste trabalho, cuja codificação em Matlab® foi adaptada e inserida na ferramenta *JImage Automation*. Porém, para executar as funções desenvolvidas em Matlab® foi necessário a utilização da biblioteca *JMatlink.* Conforme Müller (2005), *JMatlink* é uma biblioteca que usa métodos nativos Java possibilitando conectar o Matlab® através da plataforma Java, ou seja, permite a integração entre o Java e o Matlab®. As funções da biblioteca JMatlink, descritas a seguir foram utilizadas na implementação da ferramenta:

1 – Função ***engOpen()***: Abre uma conexão Matlab®, ou seja, ativa o Matlab®;

2 – Função ***engEvalString()***: Avalia uma expressão em Matlab®, usado para identificar a função criada no Matlab®;

3 – Função ***engGetFigura()***: Retorna uma imagem do Matlab®, após executar a função do Matlab® o resultado é retornado a aplicação Java.

4 – Função ***engClose()***: Fecha a conexão Matlab®.

Existem muitas outras funções desenvolvidas na biblioteca *JMatlink*. Entretanto, foi necessário apenas as funções descritas acima para integrar as duas tecnologias e obter os resultados da detecção por profundidade usando a técnica de aquisição de imagens *Multi-flash*.

3.4 Autocad

As ferramentas descritas acima foram utilizadas para o desenvolvimento do sistema *JImage Automation*, que contemplam a parte de processamento até a geração das coordenadas espaciais da imagem (peça). Para geração do programa CNC foi necessário à utilização do Autocad® 2007 e da linguagem de programação Autolisp. O Autocad® importa as coordenadas espaciais geradas pelo *JImage Automation*, efetua o cálculo do caminho da ferramenta e gera o código CNC, através da linguagem de programação AutoLisp.

Após a geração das coordenadas e importação para o CAD, é efetuado o cálculo do caminho da ferramenta, que indica a trajetória que a ferramenta deve percorrer para a construção da peça. Este cálculo é realizado observando duas regras básicas. A conectividade dos pontos e a sua direção ou sentido. O cálculo da trajetória é efetuado através da conexão entre dois pontos pelas retas traçadas entre eles, e a junção de todas as retas, irá formar o

caminho da ferramenta, o qual deverá ser percorrido para a confecção da peça. Este processo pode ser visto na seção 4.7.

3.5 Aquisição de Imagens

Como descrito anteriormente, esse trabalho apresenta duas maneiras diferentes de processamento de imagens para a geração das coordenadas. Primeiro, usando a aquisição de uma única imagem e, segundo, a aquisição de quatro imagens para aplicação da técnica *Multi-flash*.

A seguir, os seguintes passos apresentam o processo de aquisição de uma única imagem pelo qual ela deverá passar na geração das coordenadas espaciais.

1 – Aquisição da imagem via fotografia digital observando a iluminação;

2 – Converter a imagem em nível de cinza, se necessário aplicar filtros de suavização;

3 – Aplicar a detecção de bordas com um dos operadores implementados;

4 – Caso necessário aplicar *a* limiarização definindo um limiar transformando a imagem em binária, cujos valores são 0 para imagem de fundo, representado pela cor preta e 1 para as bordas, representado pela cor branca;

5 – Efetuar o *thinning*, afinamento da borda para gerar somente os pontos que definem a borda, transformando-os a um *pixel* de largura;

6 – Gerar as coordenadas espaciais para importação ao sistema CAD.

Para a técnica *Multi-flash* os seguintes passos devem ser seguidos:

1 – Adquirir quatro imagens com flashes, na parte superior, direita, inferior, esquerda e colocá-las no diretório de trabalho do Matlab® (*workspace*);

2 – Abrir a conexão com o Matlab®, ou seja, ativar a interface com o Matlab®;

3 – Detectar as bordas usando as funções Matlab®; isto é, usar a ferramenta desenvolvida em Java para chamar a função Matlab®, um arquivo *M-file*, que irá carregar as quatro imagens e retornar a imagem com as bordas detectadas para a aplicação Java;

4 – Se necessário aplicar o afinamento (*thinning*);

5 – Gerar as coordenadas espaciais para importação ao sistema CAD.

Para aquisição das imagens foi utilizada a câmara digital SONY® DSC–S40 de 4.1 *Mega Pixels*, contemplando à parte de processamento da imagem no *JImage Automation*.

Na aquisição das imagens aplicadas à técnica *Multi-flash*, foi elaborado uma caixa para fixar a câmara digital no centro e próximos a ela os orifícios para iluminação, buscando simular o efeito dos flashes conseguido pela máquina *multi-flash*. A figura 31 mostra o material utilizado para a simulação *multi-flash*.

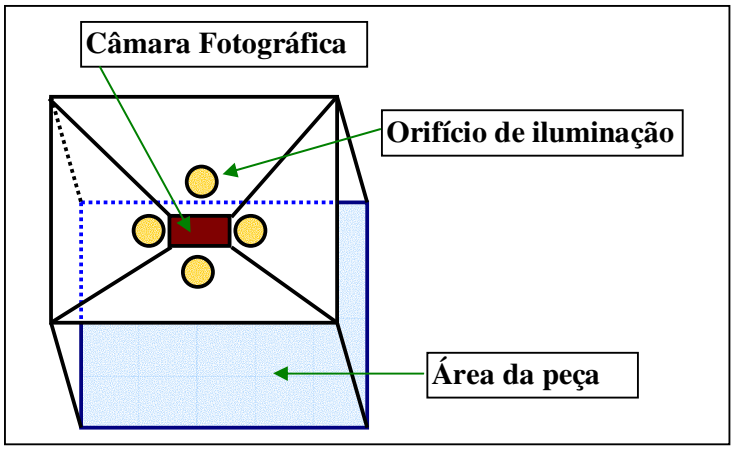


Figura 31 – desenho ilustrando material utilizado para aquisição de imagens *multi-flash*

Com esse material, conseguiu-se o efeito no sentido superior, inferior, direito e esquerdo, entretanto a utilização da máquina *multi-flash* melhora consideravelmente os resultados da aquisição.

O diagrama apresentado na figura 32 ilustra todo o processo que representam a aplicação da engenharia reversa desde a aquisição da imagem identificando o modelo real até a geração do programa CNC no CAD, para a técnica convencional e a *Multi-flash*.

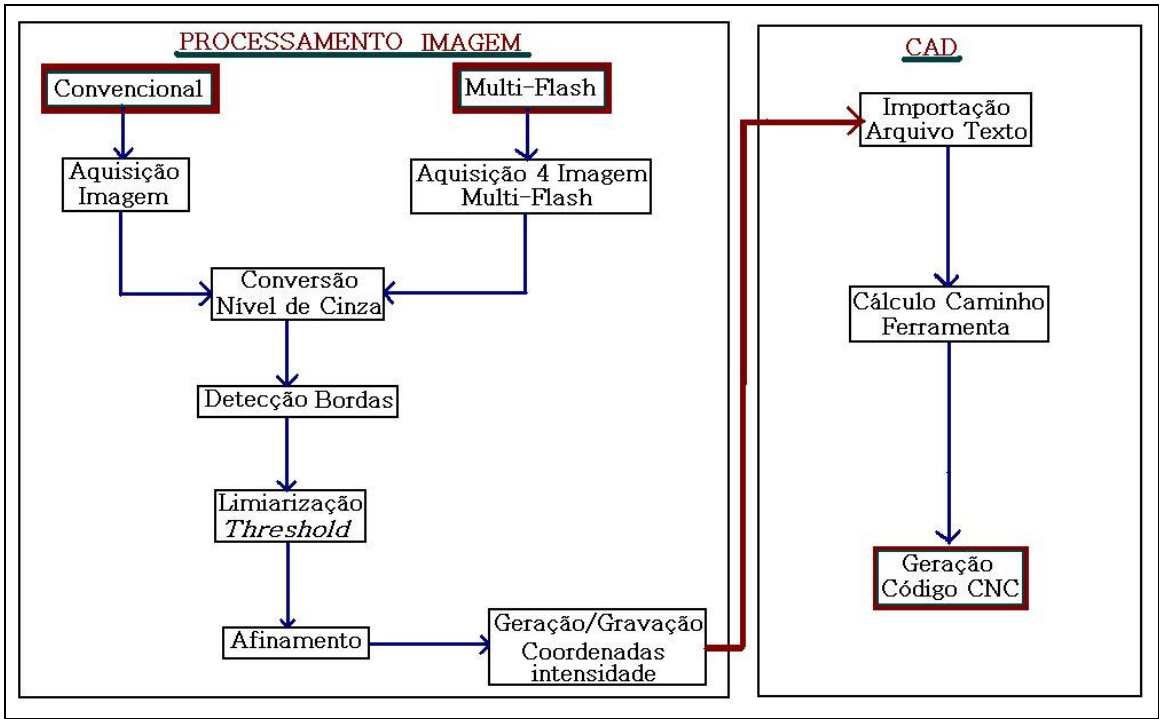


Figura 32 – diagrama esquematizando o processo de segmentação de imagens e geração do código CNC

Nos próximos capítulos serão apresentados à implementação e aplicação do *JImage Automation*, buscando avaliar a ferramenta desenvolvida e a discussão dos resultados obtidos.

4 A FERRAMENTA JIMAGE AUTOMATION

Esta secção descreve a ferramenta *JImage Automation* apresentando as opções disponíveis ao usuário, as operações e o funcionamento quanto ao processamento de imagens identificando a interface que auxilia o usuário na operação da mesma.

Para alcançar os objetivos propostos, uma interface foi desenvolvida a fim de permitir operações em imagens. Na construção dessa interface, foram utilizadas algumas ferramentas descritas no capítulo anterior. A interface pode ser vista na figura 33, onde exibe a janela interna com uma imagem de exemplo em edição. Esta imagem será usada somente para ilustração de exemplo durante esse capítulo, entretanto imagens reais de fotografia de peças serão utilizadas no capítulo de resultados e avaliação da ferramenta.

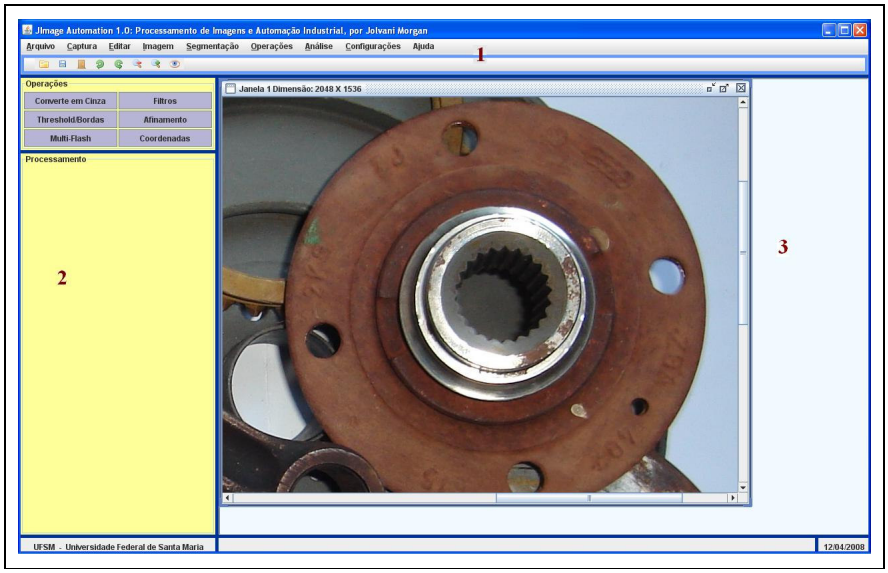


Figura 33 – Interface do *JImage Automation*

A interface do *JImage Automation* pode ser dividida em três partes. Primeira (1), a parte superior com os menus de opções, segunda (2), a parte do painel de *"Operações e Processamento"*, onde no painel *processamento* é destinado às janelas internas quando selecionado um dos menus do painel *Operações*, e terceira (3), a parte de área de trabalho onde permite que as imagens sejam abertas também em janelas internas com a classe *JInternalFrame* do Java.

Nesta interface, as opções de menu e barra de ícones contemplam operações básicas de abrir e salvar uma imagem, bem como todo o processamento necessário para efetuar operações de segmentação em imagens. Porém, para facilitar o usuário, o sistema possui dois painéis laterais, o primeiro, denominado *"Operações"*, possui o menu de opções para executar operações de processamento de imagens. O segundo painel *"Processamento"* é responsável pelas janelas internas que serão abertas quando for selecionado um dos menus de opções no painel de "*Operações"*.

Este painel lateral pode ser configurado pelo usuário, quando marcada a opção *"Operações em Imagens"* o painel fica visível, quando desmarcando as operações em imagens ficam desabilitadas. A figura 34 exibe esta opção.

Figura 34 – Controle do Painel Lateral, *Operações em Imagens*.

O menu *"Operações"* e o painel *"Processamento"* trabalham, na maioria dos casos em conjunto, quando for selecionada uma opção do menu "Operações", por exemplo, *Threshold/Bordas* uma nova janela interna é criada dentro do painel *"Processamento"* com as opções para efetuar a detecção de bordas e limiarização na imagem em edição. A seguir o funcionamento do painel lateral esquerdo.

O menu descrito a seguir faz parte do processamento da imagem encontrado também no menu superior da ferramenta. A figura 35 ilustra o menu de operações em imagens e a seguir a descrição de cada um deles.

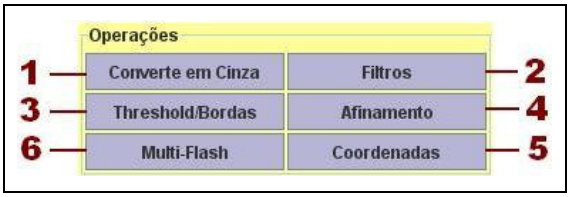


Figura 35 – Menu de operações em imagens

4.1 Operação de Conversão em Nível de Cinza

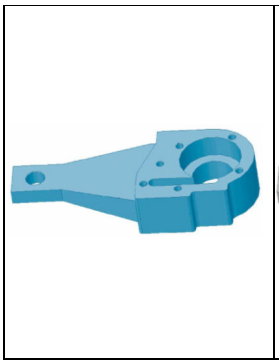
Para executar operação de conversão em nível de cinza na imagem deve-se verificar como é feita a representação da mesma no computador. Entretanto, a representação computacional de uma imagem pode ser vista como uma matriz de inteiros, cujos valores podem ser tratados numericamente, efetuando operações lógicas e aritméticas, tais como, *pixel* a *pixel* ou por vizinhança. Miranda (2006).

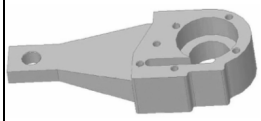
Dessa forma, para transformar uma imagem colorida em nível de cinza, é necessário obter as primitivas, vermelho, verde e azul, da escala RGB da imagem e transformar em um único valor de cinza. Este valor varia entre o preto com a menor intensidade, representado pelo valor 0 e branco com a maior intensidade, representando pelo valor 255.

Com a ajuda da biblioteca JAI, utiliza-se uma matriz com valores de 0,114, 0,587 e 0,299, que representam a adição 30% de vermelho, 59% de verde e 11% de azul, para obter o nível de cinza correspondente. Os valores da matriz são multiplicados aos valores RGB da imagem retornando o valor do tom de cinza correspondente, conforme a seguinte fórmula:

```
valorCinza = (R * 0,114) + (G * 0,587) + (B * 0,299)
```

Onde R representa a intensidade de cor em tons de vermelho (*Red*), G representa a intensidade de cor em tons de verde (*Green*) e, B representa a intensidade de cor em tons de azul (*Blue*). Dessa forma, a imagem será percorrida e a cada valor R+G+B, será gerado o valor de cinza correspondente. A figura 36 ilustra a aplicação de nível de cinza nas imagens coloridas.





```
Planar Image ResultImg = img;

double b = (double) brightness;
double[][] matrix = {
                    {0.114,0.587,0.299, b}
                    };
if (img != null) {
    ParameterBlock pb = new ParameterBlock();
    pb.add(matrix);
    JAI.create("bnadcombine", pb, null);
 }
```

a b c

Figura 36 – a) Imagem original em b) Imagem em nível de cinza c) Parte do código que transforma a imagem.

No código da figura 36 item (c) é aplicado o vetor *matrix* com os valores de intensidade de cada cor que será multiplicado a cada *pixel* da imagem original e o resultado será atribuído a variável do tipo *PlanarImage* denominada *ResultImg* que retornará a imagem em nível de cinza, como exibe a figura 36 item (b).

4.2 Aplicação de Filtros de Suavização

Quando necessária à aplicação de suavização em imagens, recorre-se ao uso de filtros. Para o filtro de mediana, a biblioteca JAI, possui um operador próprio. O trecho de código a seguir exibe a aplicação de suavização utilizando o filtro de mediana através do operador *MedianFilter* da biblioteca JAI.

```
ResultImage = (PlanarImage)JAI.create("MedianFilter", this.src);
```

A variável *ResultImage* armazena o resultado da suavização aplicado na variável "*src*" do tipo *PlanarImage* retornado pela JAI. Com isso, a suavização busca melhorar aspectos de iluminação e distorções na imagem, como remover ruídos e detalhes que não sejam interessantes para a parte em estudo. No *JImage Automation,* outros filtros foram usados através da aplicação de máscaras.

Entretanto, aplicar a técnica de suavização depende da imagem, ou de sua qualidade visível. Sua aplicação também depende do operador utilizado e do tipo de característica que deseja extrair, não existindo uma fórmula para todas as imagens, cada caso é um caso, e deve ser analisado pelo utilizador, que deve verificar a necessidade de aplicação, ou não, da suavização na imagem. A não necessidade de suavização pode ser eliminada na hora da aquisição da imagem, através de cuidados com a iluminação e distorções.

4.3 Limiarização (*thresholding*) e detecção de Bordas

Ao selecionar o menu de *threshold e detecção de bordas*, assim como no menu de filtros, abre-se uma janela interna no painel de *"Processamento"*, possibilitando que o usuário selecione o tamanho da máscara e o operador desejado para a detecção de bordas.

Entretanto, para executar a extração das bordas utiliza-se de operadores de convolução através da técnica de convolução por máscaras. Neste trabalho foram disponibilizados vários operadores, são eles: Frei-and-Chen, Kirsch, Prewitt, Roberts, Sobel e o operador Gradiente. As máscaras utilizadas foram 3 x 3, 5 x 5, 7 x 7 e 9 x 9. Porém o código não está implementado para todas as máscaras. A figura 37 apresenta a janela interna com as opções.

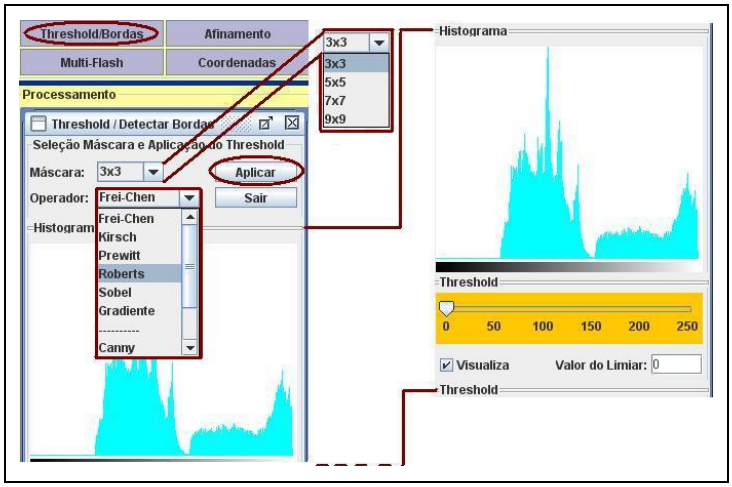


Figura 37 – Janela Interna para aplicação de Threshold e detecção de bordas

Como visto anteriormente, a maioria dos operadores detecta as bordas através de duas convoluções, uma horizontal e outra vertical. Porém, no operador Roberts o Gradiente é aplicado na direção de 45º graus. Para exemplificar a aplicação de máscaras e o uso de operadores a figura 38 apresenta o trecho de código que implementa o método de detecção de bordas ***Frei-and-Chen***, no sentido horizontal e vertical com máscara, de tamanho 3x3.

```
public PlanarImage DetectarBordas(PlanarImage image, int mascara) {
       this.src = image;
      PlanarImage dst = null;
     if (mascara == 3) {
           float[] freichen_h_data = { 1.0F,    0.0F, -1.0F,
                                       1.414F,  0.0F, -1.414F,
                                       1.0F     0.0F, -1.0F};

           float[] freichen_v_data = { -1.0F, -1.414F, -1.0F,
                                        0.0F,   0.0F,    0.0F,
                                        1.0F,   1.414F,  1.0F};

            KernelJAI Kern_h = new KernelJAI(3,3,freichen_h_data);
            KernelJAI Kern_v = new KernelJAI(3,3,freichen_v_data);

            dst = (PlanarImage)JAI.create("gradientmagnitude", this.src,
                                          kern_h, kern_v);
         }
    return dst;
}
```

Figura 38 – Parte do código para aplicação da máscara de Frei-and-Chen.

Para todos os operadores a aplicação de convolução por máscara segue a mesma codificação, entretanto a única alteração são os valores e o tamanho da máscara. Além disso, às máscaras utilizadas neste trabalho possuem tamanhos ímpares (3x3, 5x5, 7x7...) para garantir uma aplicação simétrica da mesma sobre a imagem conforme. A figura 39 apresenta as seguintes máscaras 3 x 3 e 5 x 5, onde foram usadas para o operador Frei-and-Chen:

-1	**-1,414**	**-1**
0	**0**	**0**
1	**1,414**	**1**

a

1	**0**	**-1**
1,414	**0**	**-1,414**
1	**0**	**1**

b

-3	**-2**	**-2,828**	**-2**	**-3**
-2	**-1**	**-1,414**	**-1**	**-2**
0	**0**	**0**	**0**	**0**
2	**1**	**1,414**	**1**	**2**
3	**2**	**2,828**	**2**	**3**

c

3	**2**	**0**	**-2**	**-3**
2	**1**	**0**	**-1**	**-2**
2,828	**1,414**	**0**	**-1,414**	**-2,828**
2	**1**	**0**	**-1**	**-2**
3	**2**	**0**	**-2**	**-3**

d

Figura 39 – Máscara Frei end Chen a) 3x3 horizontal, b) 3x3 vertical, c) 5x5 horizontal e d) 5x5 vertical

O resultado da aplicação de detecção de bordas pode ser visto na avaliação da ferramenta do capítulo seguinte.

Na mesma janela para detectar bordas, da figura 37, está implementado o *thresholding* (limarização), juntamente com o histograma da imagem. O histograma estabelece a distribuição dos *pixels* em nível de cinza, e abaixo dele, a barra de limiar ou *threshold* criada através do componente *JSlider*. Esta barra varia de 0 a 255, indicando que o valor de limiar que a imagem pode assumir fica entre essa faixa.

A utilização do *threshold* transforma a imagem em binária, cujos *pixels* terão valores 0 e 1. Sua aplicação é efetuada manualmente e a seleção do limiar ocorre ao deslizar o componente *JSlider*, onde conforme o valor do limiar, irá estabelecer novos parâmetros para separar o objeto do fundo da imagem. Tal aplicação também se caracteriza em uma operação global, onde se aplica um único limiar para toda imagem. O trecho de código a seguir, figura 40, exibe a aplicação de *threshold* com o uso da biblioteca JAI.

```
this.imagem = img;
// Binarizes the original image.
ParameterBlock pb = new ParameterBlock();
pb.addSource(imagem);
pb.add(1.0*threshold);
// Creates a new thresholded image
thresholdedImage = JAI.create("binarize", pb);

display.set(thresholdedImage);
```

Figura 40 – Trecho de código da binarização de imagem

A variável *thresholdedImage* retorna a imagem binarizada (limiarizada), e a cada mudança de valor de limiar efetuado pelo usuário ao movimentar o *JSlider* retorna uma nova imagem binarizada.

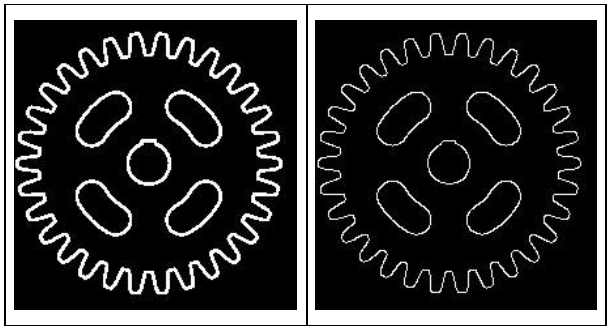
4.4 Afinamento (*Thinning*)

Após executar a detecção de bordas e o *threshold*, se necessário, aplica-se a técnica de afinamento, buscando reduzir a borda extraída ao mínimo possível, preferencialmente a um *pixel* na imagem.

Para o afinamento foi utilizado a algoritmo paralelo por Zhang e Suen (1984), que utiliza operações para remover os *pixels* marcados para remoção através de duas iterações, como visto no capítulo 2.

A aplicação do afinamento busca reduzir a informação da imagem, para que quando efetuado a geração das coordenadas espaciais para importação no CAD, estas gerem somente os pontos necessários para o cálculo da trajetória da ferramenta.

A figura 41 item (a) apresenta a detecção de bordas da peça. A figura 41 item (b) exibe o resultado da aplicação do algoritmo de Zhang e Suen. O algoritmo pode ser encontrado em Cheng (2004), que foi adaptado e inserido neste trabalho.



a b

Figura 41 (4.8) – Aplicação do Algoritmo de Zhang e Suen

Para a figura 41 item (a), foi aplicado o operador Sobel na detecção de bordas com máscara 3 x 3, e executado o *threshold* com limiar de 129, a seguir a figura 41 item (b) foi aplicado o algoritmo de afinamento. Observa-se uma significativa diferença na redução das bordas após o afinamento.

Entretanto, a utilização do algoritmo de afinamento irá reduzir o processamento no Autocad para definir as retas que formam o contorno da peça, cuja descrição encontra-se na seção 4.7.

4.5 Extração das Coordenadas Espaciais

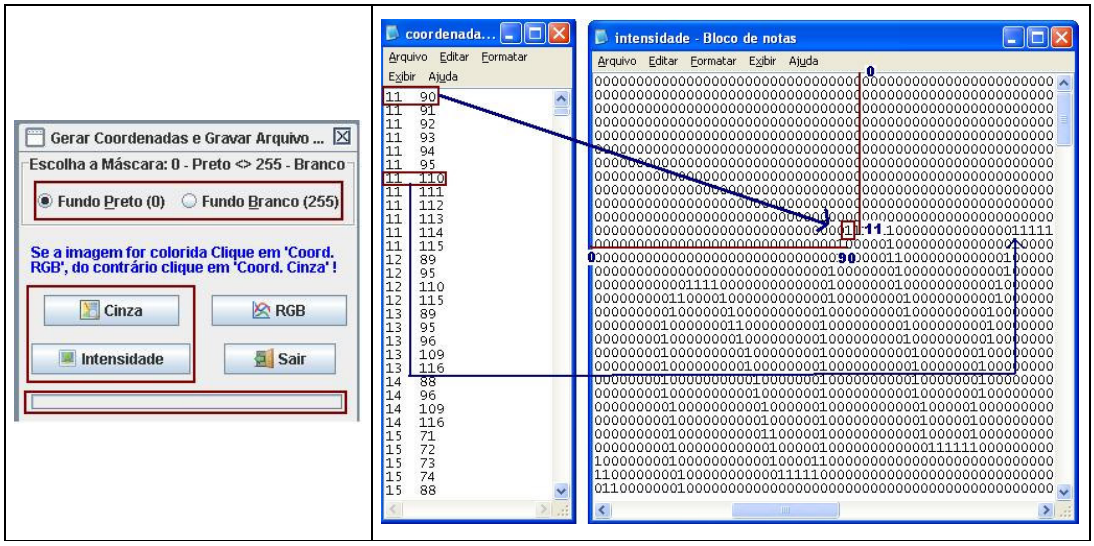
Após aplicação da detecção de bordas e do afinamento, o último passo executado pelo *JImage Automation* é a extração das características da imagem, através das coordenadas espaciais ou da intensidade dos *pixels* da imagem.

Tanto a extração das coordenadas como a intensidade dos *pixels*, o algoritmo percorre a imagem *pixel* a *pixel* e cria um arquivo texto com os valores que representam os *pixels* da imagem. Porém, na detecção das coordenadas, são identificadas aquelas que representam a

borda da imagem, ou seja, os *pixels* com valores igual a um (1), definidos anteriormente pelo *threshold* e ou afinamento. Na extração da intensidade, o arquivo texto gerado conterá todos os *pixels* da imagem, se esta for binária este arquivo conterá *pixels* com valores 0 e 1, definindo o objeto (bordas) e o fundo da imagem.

Além disso, a extração da intensidade poderá ser efetuada a qualquer momento, com a imagem original, ela transformada em nível de cinza, após aplicação da detecção de bordas e outras opções.

A figura 42 apresenta a janela para a geração de coordenadas e intensidade, e o resultado desse processo.



a b c

Figura 42 – a) Janela Interna para gerar as coordenadas, b) Geração das coordenadas, c) Geração da intensidade

A janela interna de coordenadas figura 42 item (a), permite selecionar o fundo preto na maioria dos casos e o fundo branco quando a imagem for invertida pelo usuário. Estando a imagem em nível de cinza utiliza-se o botão "cinza" para gerar as coordenadas, figura 42 item (b), caso contrário o botão "RGB". A intensidade será gerada tanto para imagens coloridas como para níveis de cinza. Na figura 42 item (c), é ilustrado a geração de intensidade na imagem em nível de cinza, cujo formato é parecido com a imagem original (figura 41). Ainda na figura 42 item (a) e (b), a comparação entre os arquivos textos de coordenadas e intensidade, comprovando que as coordenadas do arquivo texto de "*coordenadas*" correspondem realmente à linha e coluna do arquivo texto de "*intensidade*", cujos pontos com valores iguais a um (1) indicam a borda da imagem (coordenada 11, 90 = intensidade 1).

O código abaixo, figura 43, ilustra parte da operação de geração das coordenadas.

```
//Get the number of bands on the image.
sampleModel sm = imagem.getSampleModel();
nbands = sm.getNumBands();
int[] vetPixel = new int[nbands];
RandomIter iterator = RandomIterFactory.create(imagem, null);
    try {
        for(int h=0;h<height;h++){
            for(int  w=0;w<width;w++){
             iterator.getPixel(w,h, vetPixel);
             for(int band=0; band<nbands; band++){
                  if (pixel[band] != valor && (band == 0))
                    gravarCoordenadas(out, w, h);
                    vetor[control] = vetPixel[0];
                  }//fim for band
             }//for w
           control++;
         } //fim f
```

Figura 43 – Código para geração das coordenadas espaciais

O código apresenta a chamada ao método *gravarCoordenadas(...)*, responsável pela gravação das coordenadas ao arquivo texto. O uso da JAI auxilia durante a manipulação da imagem com o método *getNumBands()* que retorna o número de bandas da imagem, e criando uma variável *iterator* responsável pela inserção do *pixel* de coordenada *x,y* representadas por *h,w* no vetor *vetPixel[]* através do método *getPixel* representado pelo comando *iterator.getPixel(w,h, vetPixel);*.

As coordenadas serão geradas e gravadas em arquivo texto, permitindo que o usuário atribua um nome para este arquivo. E finalmente importadas ao sistema CAD.

4.6 Aplicação da Técnica Multi-flash

A técnica *Multi-flash* é independente das outras operações. Quando esta técnica é aplicada, todo o processamento é executado no Matlab®, ficando para o sistema *JImage Automation* o afinamento e geração das coordenadas.

Para alcançar os resultados, a parte relevante da técnica é a forma de aquisição da imagem. Entretanto, para obter o efeito conseguido por Ramesh, foi previamente elabora uma caixa com orifícios para que a aplicação de luzes substituísse os flashes, conforme visto no item 3.5 do capítulo metodologia.

Após a aquisição, é efetuado o processamento no Matlab® integrando as duas tecnologias, como mostra a figura 44, que ilustra a aplicação da técnica *Multi-flash* no *JImage Automation* e os métodos das chamadas ao Matlab®.


```
JMatLink engine;

engine.engOpen();
```

```
engine.engEvalString("FindDepthEdges");
Image image = engine.engGetFigure(1, 500, 400);
PlanarImage img = JAI.create("AWTImage",image);
```

```
engine.engClose();
```

```
engine.engCloseAll();
```

Figura 44 – Menu de Aplicação *multi-flash* e chamada aos métodos no Matlab®

JMatLink é a classe principal da biblioteca JMatLink 1.3.0, e é instanciado para permitir o acesso ao Matlab® *(JMatLink engine)*, o código fonte, exemplos e a documentação da biblioteca podem ser encontrados em Muller (2005).

A parte principal da implementação Java se encontra no menu ***Bordas – Técnica Multi-Flash***, cuja função é enviar uma variável do tipo *string* com o nome da função, sem a extensão, que será executada no Matlab®, através do método *engine.engEvalString(...)*.

A função *FindDepthEdges.m* enviada ao Matlab® pelo método anterior, irá ler as imagens adquiridas com os flashes e efetuar o cálculo das bordas de profundidade gerando uma nova imagem com a detecção das bordas. No método implementado por Ramesh, foi utilizado o operador Sobel. Porém pode ser utilizado qualquer outro operador para a detecção de bordas no Matlab®.

O resultado da função *FindDepthEdges.m* (a imagem) será retornado ao Java pelo método *engGetFigure()*, que abre a figura numa janela de tamanho 500 por 400. No Java é criada uma variável do tipo *Image* para receber a imagem do Matlab®. O método correspondente é *Image image = engine.engGetFigure(1, 500, 400)*. Finalmente a variável *image* resultado, do tipo *Image*, será convertida para uma imagem do tipo *PlanarImage* pela JAI, através do método *PlanarImage img = JAI.create("AWTImage", image)*. A variável *img*, então, é aberta pela aplicação *JImage Automation* para gerar as coordenadas.

O código fonte em Matlab®, resultados e artigos podem ser encontrados em Ramesh et al (2004) e Martinez (2005).

Após finalizar a aplicação da técnica *Multi-flash*, é efetuado o afinamento e geração das coordenadas espaciais da imagem.

Terminado o processamento com a imagem, o arquivo texto gerado que contém as coordenadas é importado pelo CAD. As coordenadas desse arquivo identificam os pontos onde é efetuado o cálculo das retas que formam as entidades geométricas, ou seja, a geometria ou perfil da imagem (peça), que definem o caminho da ferramenta para geração do código CNC. Esse processo será descrito na seção seguinte.

4.7 Cálculo da Trajetória e Geração do Programa CNC

O processamento dos arquivos texto gerados pelo sistema *JImage Automation* é realizado em plataforma CAD na forma de aplicativo. A implementação é feita na linguagem AutoLisp, a qual permite explorar recursos do AutoCAD para criar e editar entidades de desenho, executar comandos do CAD e elaborar menus para adicioná-los ao CAD.

O arquivo texto é inicialmente importado no CAD, através de funções específicas da linguagem de programação. Os dados contidos em arquivos desse tipo representam *pixels* da imagem da peça processada no *JImage Automation*. Para identificar a trajetória da ferramenta no contorno definido, esses dados são convertidos em entidades ponto do CAD. A metodologia adotada é a de ajustar esses pontos a segmentos de retas através de duas regras: a sua conectividade e sua direção.

Para iniciar o processamento de identificação do caminho da ferramenta, primeiro é necessário especificar manualmente o ponto inicial. Em seguida, o ponto para estabelecer a direção que deverá ser efetuado este processamento. Nesse sentido, o usuário define o ponto que dará inicio a trajetória da ferramenta no programa CNC, e estabelece o sentido de deslocamento, para o caso de uma geometria fechada.

A técnica por conectividade usa o conceito de vizinhança, pois cada *pixel* da imagem poderá estar conectado com até outros oito pontos, figura 45. Este número de pontos na vizinhança estabelece o valor de sua conectividade. Nesta técnica, esse valor é um dos parâmetros para estabelecer o caminho da ferramenta.

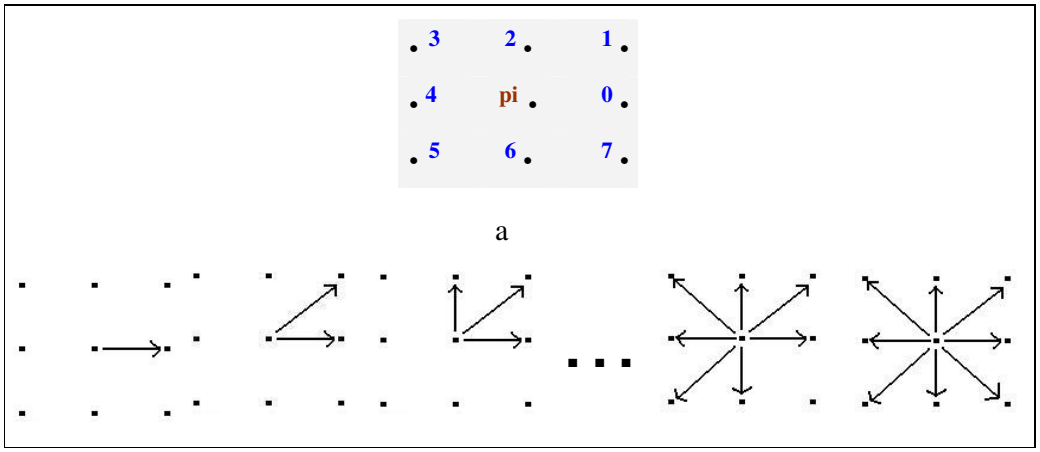
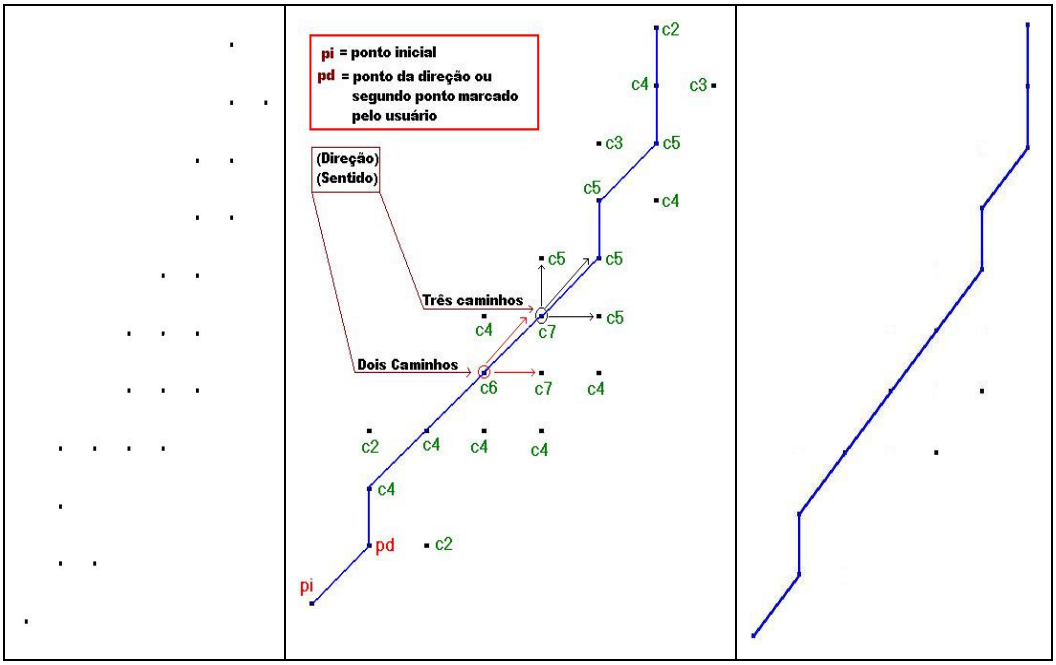


Figura 45 – Implementação do cálculo da trajetória da ferramenta

Na figura 45 item (a), o ponto central *pi* representa o ponto de início de cada segmento de reta do processamento. Esse ponto também representa o final de segmento de reta anteriormente identificada. A reta inicial para o processamento é a definida pelos dois pontos informados pelo usuário. A identificação da direção para o segmento de reta subseqüente depende do cálculo da conectividade de seus vizinhos. As possíveis direções são codificadas com números que variam de zero (0) a sete (7).

Além da conectividade, o ponto preferencial para estabelecer a trajetória no próximo passo também depende da direção de origem, ou seja, da direção precedente. Com o intuito de ilustrar esse procedimento, a figura 46 é adotada como exemplo da trajetória através das regras de conectividade e direção.



a b c

Figura 46 – a) Pontos carregados no CAD, b) cálculo da trajetória através da conectividade e direção, c) trajetória ou caminho da ferramenta

No item (a) da figura 46, estão representadas as entidades "pontos" geradas no ambiente CAD. No item (b) está identificado o ponto inicial ***pi*** da trajetória e o segundo ponto ou ponto de direção ***pd***, que são marcados pelo usuário. Os símbolos *c2, c3, c4, c5, c6* e *c7* indicam o valor da conectividade de cada ponto, calculada pelo algoritmo. No exemplo, observa-se que o ponto ***pd*** tem pontos vizinhos à direita com conectividade 2 (*c2*) e acima com conectividade 4 (*c4*). Pelo critério da conectividade, a reta é traçada para o ponto de maior conectividade (*c4*), como mostra a figura.

Quando o processo depara-se com pontos cujos vizinhos possuem mesmos valores de conectividade como mostra o ponto de valor *c6* (círculo vermelho), dois caminhos são possíveis: à direita na horizontal e à direita na diagonal, ambos com valor de conectividade 7 (*c7*). Neste caso, aplica-se a segunda regra, que refere-se ao sentido da trajetória. Como a trajetória precedente está no sentido diagonal, o ponto de preferência para passar a reta de ajuste será o da diagonal. Na seqüência, observa-se que no próximo passo, três posições de avanço são possíveis, pois seus vizinhos possuem valores de conectividade igual a cinco (5). Novamente, pelo critério da direção preferencial, a trajetória da ferramenta segue o sentido da diagonal. Todos os pontos que fazem parte da trajetória são eliminados do processamento e os pontos que não pertencem à trajetória, ou descartados pela mesma, também são eliminados, resultando somente o trajeto que deve ser percorrido pela ferramenta como mostra a figura 46

item (c). Isso evita que qualquer caminho já percorrido seja novamente reconhecido pelo processamento como seqüência da trajetória.

Nesse contexto, os seguintes passos devem ser seguidos para obter o caminho da ferramenta:

1 – Marcar o ponto inicial;

2 – Marcar o ponto que indica a direção da trajetória, o segundo ponto da reta;

3 – Procurar os pontos vizinhos observando a conectividade e o sentido da trajetória;

4 – Eliminar do processamento os pontos usados na trajetória;

5 – Eliminar os pontos descartados, que não fazem parte da trajetória;

Pelo procedimento aplicado, cria-se em ambiente CAD segmentos de retas que representam o ajuste de pontos de contorno da imagem de uma peça à trajetória da ferramenta para sua fabricação. Para gerar o código do programa CNC, recursos de integração CAD/CAM devem ser explorados. Para esse fim, foram aplicadas neste trabalho funções de geração automática de programas CNC, obtidas como resultado de trabalhos de pesquisa desenvolvidos pelo Grupo de Pesquisa em Automação e Processos de Fabricação – NAFA/UFSM, Silva (2000-2007). Funções de simulação de programas CNC, também desenvolvidas nessa linha de pesquisa, podem ser aplicadas para validação dos resultados.

A seguir, o capítulo 5 apresenta os resultados onde se pode avaliar a ferramenta como uma forma de aplicação do processamento de imagens e engenharia reversa para a geração do programa CNC.

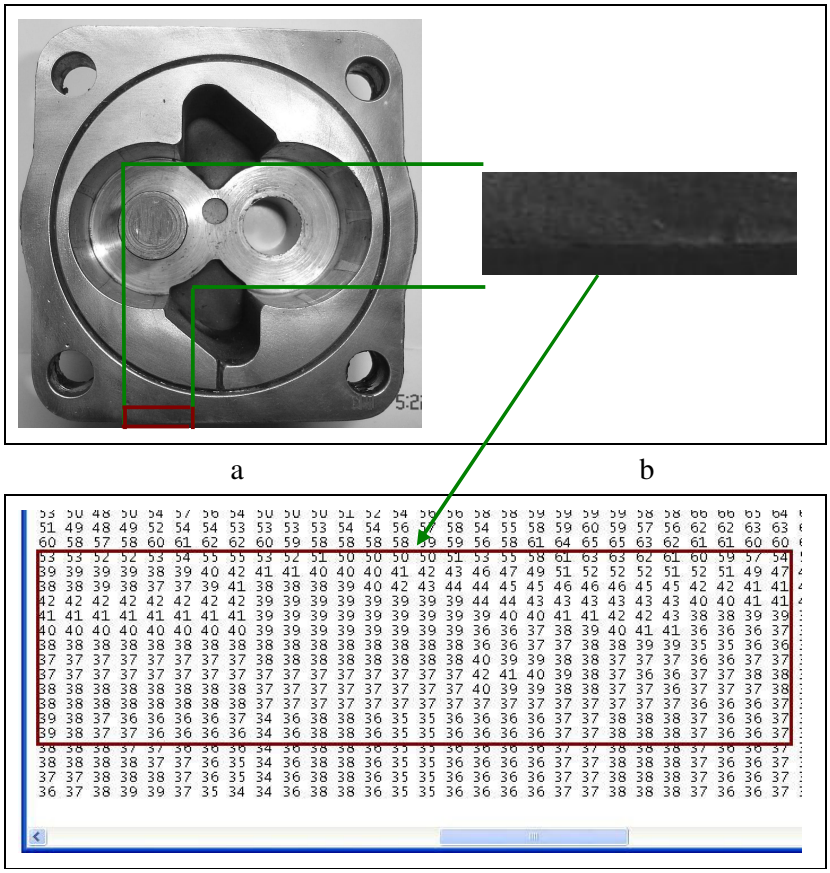
5 RESULTADOS E AVALIAÇÃO DA FERRAMENTA

A apresentação dos resultados é elaborada de duas maneiras. Primeiro, aborda-se os resultados com o uso de imagens de aquisição convencional, a partir de uma fotografia de uma peça. Segundo, é apresentado o método *multi-flash*, cuja aquisição das imagens foi elaborada através do uso de luzes nos quatro lados da imagem para simulação da câmara *multi-flash*.

Observa-se que a aplicação da ferramenta limita-se as imagens 2D, ou seja, foi desenvolvida para aplicar em imagens planas. Porém, para exemplificar a diferença da aplicação das duas técnicas, foram utilizadas imagens com perspectivas 3D demonstrando a diferença na detecção de bordas de profundidade.

5.1 Aplicação de Operadores Tradicionais

Para análise dos resultados, diferentes imagens foram adquiridas e processadas por diferentes operadores. Imagens selecionadas desse conjunto são apresentadas nas figuras e comentários a seguir, onde foi aplicado o *threshold* usando diferentes limiares e o filtro de mediana para melhor representar o contorno da peça. A figura 47 exibe a imagem de uma bomba de óleo do motor, já convertida para nível de cinza, utilizadas no processo de avaliação dos resultados. A figura 47 item (a) ilustra a imagem original adquirida pela câmara fotográfica digital. Para processamento dos dados, deve-se inicialmente analisar a qualidade da imagem. A figura 47 item (b) mostra uma parte da imagem com problemas de iluminação, cuja diferença em nível de cinza é muito pequena entre o fundo e a imagem original. A figura 47 item (c) apresenta o arquivo contendo os valores da intensidade dos *pixels*, cujas diferenças relativas não são significativas, como comentadas anteriormente.

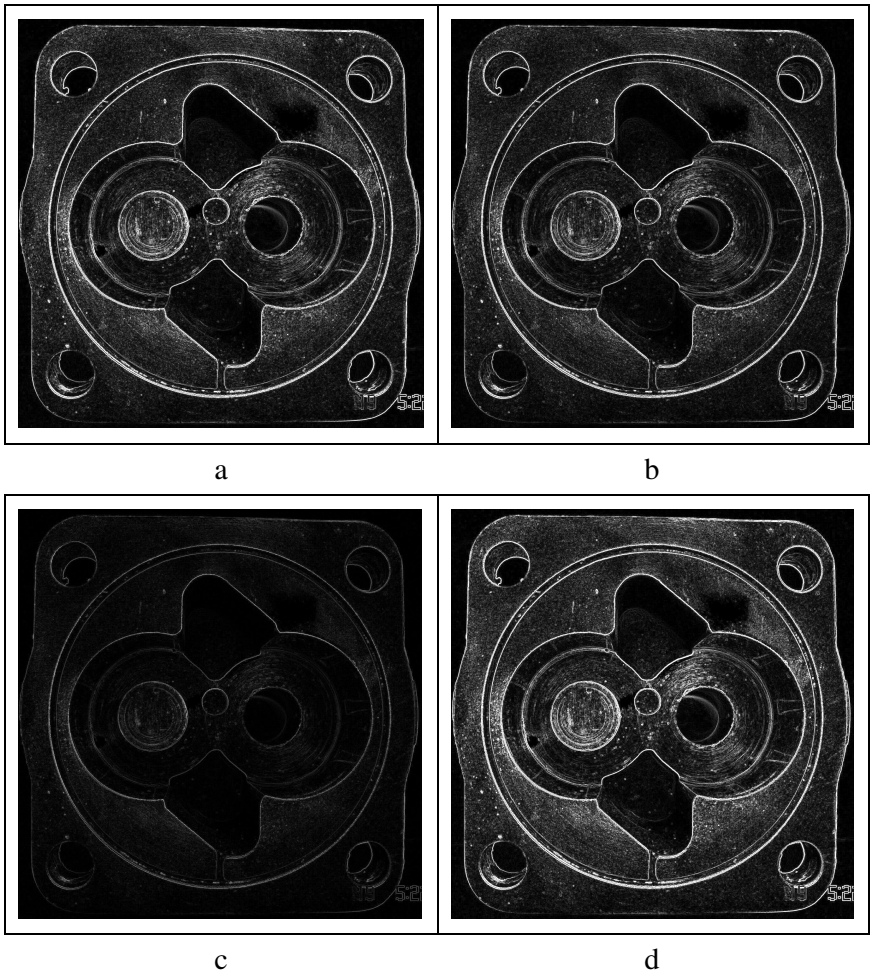


a b

c

Figura 47 – Peça em nível de cinza utilizada na avaliação dos resultados, a) imagem original, b) parte da imagem com problemas de iluminação, c) *pixels* da parte da imagem com valores próximos

A peça da figura 47 possui pequenos problemas de iluminação que podem ser vistas na parte superior e inferior da mesma. Estes problemas acarretam bordas não bem definidas, devido aos valores de intensidade dos *pixels* em nível de cinza da peça ser próximos aos valores dos *pixels* do fundo da imagem. Esse efeito é menos significativo na parte superior. Para executar a segmentação, vários operadores foram utilizados e estão implementados no *JImage Automation*. Entre eles estão: Frei-and-Chen, Gradiente, Kirsch, Prewitt, Robert, Sobel. Entretanto, são apresentados apenas quatro deles: Frei-and-Chen, Prewitt, Roberts e Sobel. A figura 48 ilustra a aplicação dos operadores de convolução com máscara 3 x 3 para a detecção das bordas.



a b

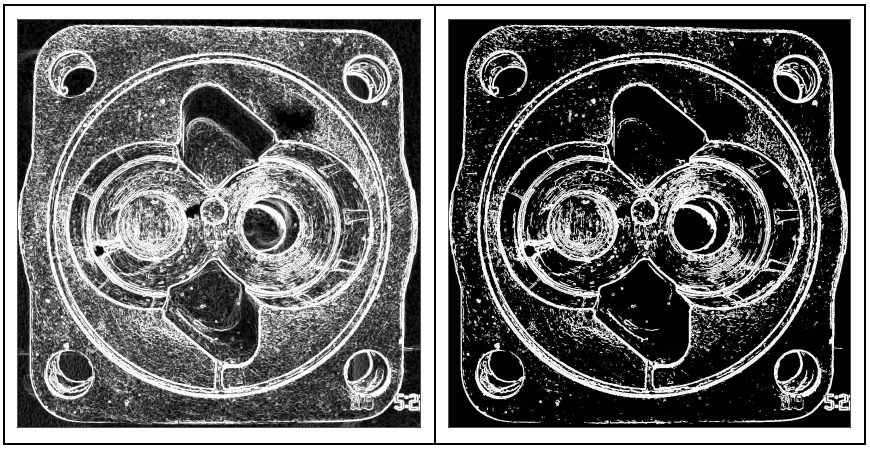
c d

Figura 48 – Resultado da aplicação dos operadores de convolução. a) Frei-and-Chen; b) Prewitt, c) Roberts e d) Sobel

Analisando as imagens da figuras 48, constata-se que existe diferença significativa entre os operadores Frai-and-Chen (a), Prewitt (b) e Sobel (d), em relação ao operador Roberts (c). Pouca diferença foi observada entre Frai-and-Chen (a) e Prewitt (b), sendo que o operador Sobel (d), que detectou as bordas com maior evidência, é considerado o mais sofisticado de todos.

Esta abordagem representa que os três operadores citados realçam mais as bordas, cuja convolução ocorre no sentido horizontal e vertical, diferentemente do operador Roberts (c), onde a convolução ocorre no sentido diagonal.

Para realçar mais as bordas, aumenta-se o tamanho da máscara. Neste caso, optou-se pelo operador Frei-and-Chen com máscara de tamanho 5 x 5, descrito na seção 4.3. A figura 49 ilustra a detecção de bordas com o referido operador.



a b

Figura 49 – a) Operador Frei-and-Chen com máscara 5 x 5, b) Aplicação do threshold 182

A figura 49 exemplifica o uso da máscara 5 x 5, ilustrando uma melhora na detecção de bordas ao realçar mais as mesmas. Porém, identificando muito mais as bordas que não fazem parte do contorno da peça. Por outro lado, consegue-se realçar mais as bordas na parte inferior da peça. Foi utilizado um *threshold* de 182, figura 49 item (b), para eliminar pontos que não pertencem à borda.

Após a aplicação dos operadores, utiliza-se um limiar (*thresholding)* global sob a imagem segmentada. Nesse caso, um procedimento indicado, pois os operadores não conseguem definir valores exclusivos para as bordas, nos seguintes aspectos:

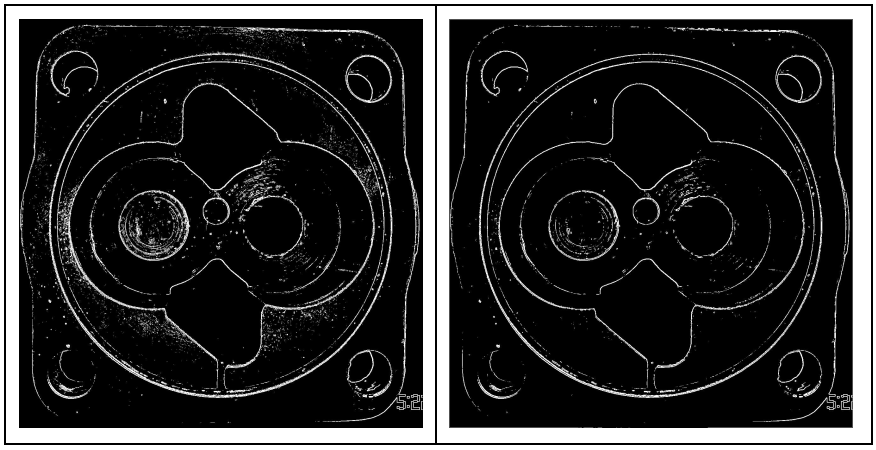
1) onde o operador diferencial apresenta respostas nulas em regiões homogêneas;

2) em regiões cuja derivada é nula;

3) em regiões onde a magnitude é representada por valores pequenos.

Nestes casos, os *pixels* que representam a borda da imagem ainda não são totalmente conhecidos. Como conseqüência, as bordas não ficam bem definidas.

Dessa forma, a limiarização (*thresholding*) transforma a imagem no formato binário, cujo plano de fundo é representado pelo valor zero (0) e as bordas pelo valor um (1), conseqüentemente, resultando na imagem segmentada, as bordas da mesma. A aplicação do *threshold* serve para realçar as bordas que não foram detectadas com os operadores ou eliminar as bordas falsas. A figura 50 item (a) exibe a aplicação da limiarização na imagem da figura 48 item (d), onde foi aplicado o operador Sobel. Na figura 50 item (b), foi aplicado o filtro de mediana antes de realizar o *threshold* cujo efeito gerado foi à suavização da imagem.



a b

Figura 50 – Threshold com limiar 148. a) após aplicação do operador Sobel, b) com filtro de mediana

A figura 50 item (a), ilustra a aplicação de threshold com limiar 148. Com este limiar, as bordas ficaram mais definidas. Porém, por um lado algumas bordas que pertencem ao contorno da peça foram eliminadas, principalmente as bordas externas da peça. Por outro, foram eliminadas as falsas bordas, aquelas que não fazem parte do contorno, isto pode ser observado comparando-se com a figura 48 item (b). Ao aplicar o filtro de mediana figura 50 item (b), a suavização melhorou na detecção eliminando ainda mais algumas bordas falsas, porém, também foram eliminadas ainda mais as bordas que pertencem ao contorno da peça.

Visualizando as imagens, observa-se que, principalmente na parte inferior, devido ao problema de iluminação citado anteriormente, o contorno da peça não foi totalmente detectado. Por outro lado, partes que não pertencem à borda foram detectadas como tal. Entretanto, a utilização de diferentes máscaras, o uso de diferentes operadores e o cuidado com a iluminação no momento de aquisição interferem no processo de segmentação da imagem. Assim, para segmentar a imagem, deve-se analisar caso a caso, buscando aplicar operadores e máscaras para obter o melhor resultado. Não existe um único método que solucione todos os problemas de segmentação.

Após a aplicação do *threshold*, é executado o afinamento da imagem, neste caso da borda, buscando reduzir sua espessura a um *pixel* de largura, eliminando pontos que não pertencem a mesma. Esse processo reduz consideravelmente o processamento de identificação da trajetória da ferramenta no ambiente CAD. A figura 51 ilustra o resultado do afinamento sobre a imagem apresentada na figura 50 item (b).

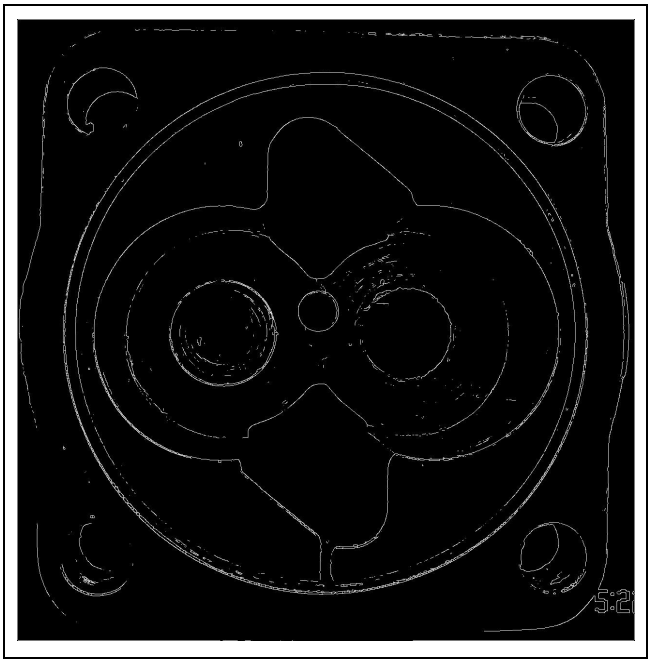
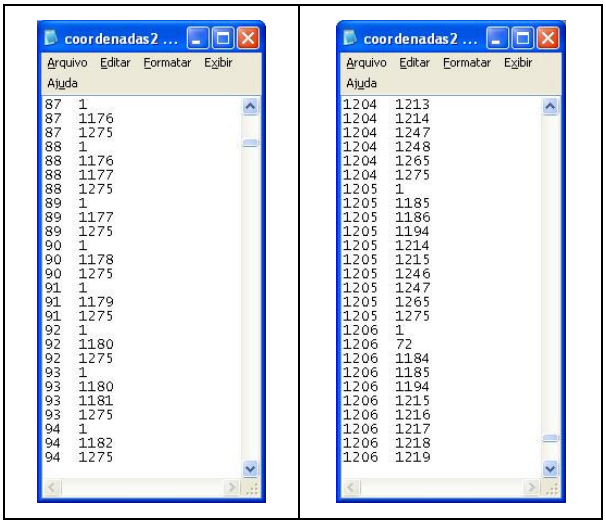
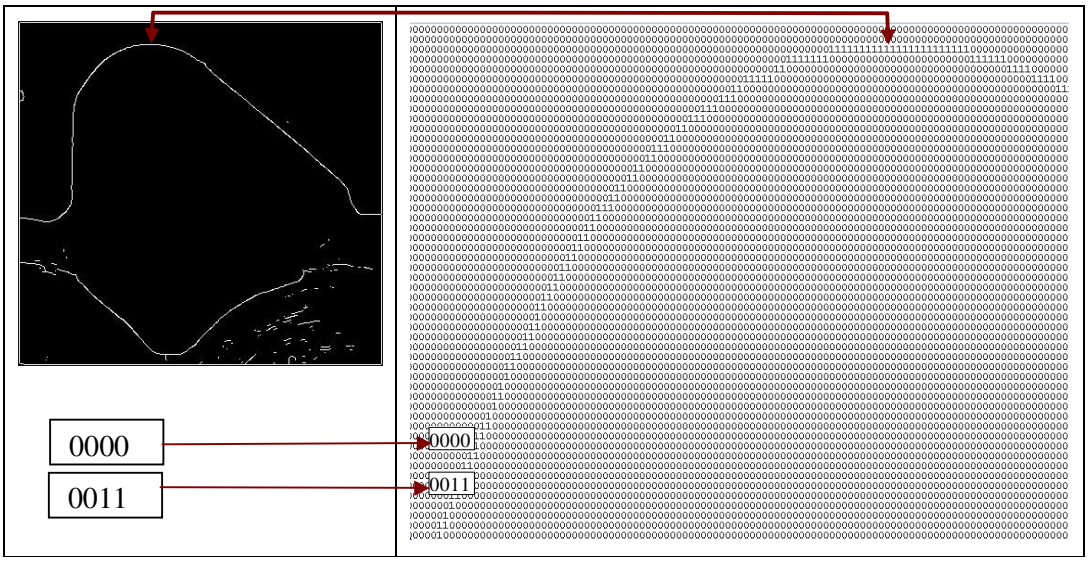


Figura 51 – Resultado do Afinamento

Finalizado o processo de segmentação da imagem, a ferramenta *JImage Automation* permite gerar dois arquivos textos contendo informações sobre as coordenadas e intensidade dos *pixels*, para importação no CAD. A figura 52 ilustra os arquivos textos gerados a partir da figura 51. Devido às dimensões da imagem, o arquivo de intensidade foi somente apresentado para a parte da imagem ilustrada pela figura 52 item (b).



a



b

Figura 52 – a) Arquivo de coordenadas da borda da imagem, b) arquivo de intensidade da imagem

A figura 52 item (a) apresenta as coordenadas que representam os pontos da borda da imagem, onde os valores para a geração da mesma são representados por um (1). Na figura 52 item (b), podem ser visualizados os valores dos *pixels* da imagem binarizada, ou seja, o valor zero (0) representa o preto, fundo da imagem, e o valor um (1) representa o branco, a borda da imagem.

A importação para o CAD é efetuada utilizando o arquivo de coordenadas, o qual irá reproduzir a imagem segmentada para o cálculo da trajetória da ferramenta através de entidades "ponto" no CAD. Para isso, foram desenvolvidos métodos que permitissem a importação do arquivo texto, calcular a trajetória da ferramenta e gerar o programa CNC,

conforme descritos na seção 4.7. O aplicativo CAD apresenta três possibilidades de direcionamento do processamento, como mostrado na figura 53. A opção “Buscar arquivo imagem” permite selecionar o arquivo de texto contendo coordenadas e intensidade de *pixels* obtidos no processamento da imagem. Essa função cria as entidades “ponto” correspondentes no CAD. O ajuste desses pontos a segmentos de reta para identificar a trajetória da ferramenta é executado pela função “Gerar trajetória”. O resultado do processo pode ser visualizado pela opção “Listar programa CNC”.

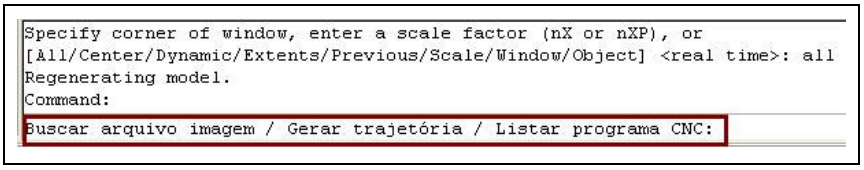


Figura 53 – Linha de Comando para Gerar o programa CNC no Autocad

A figura 54 ilustra os pontos desenhados em CAD, importados do arquivo texto de coordenadas da imagem da figura 51, processada pelo sistema *JImage Automation*. O detalhe observado em ampliação (*zoom*) mostra que a borda ficou com espessura de um *pixel*, devido ao afinamento executado.

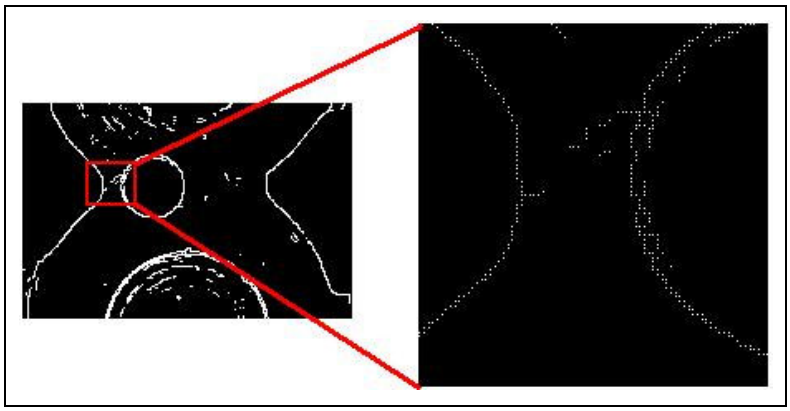


Figura 54 – Importação do arquivo texto no CAD

Para a imagem da figura 50 item (b), cuja imagem está segmentada sem o afinamento, os resultados para a mesma região são apresentados na figura 55. Através do recurso de ampliação de detalhes da tela de desenho do CAD (*zoom*), observa-se que o número de pontos que formam a borda é significativamente maior. Isso implica que, ao efetuar o cálculo da trajetória, o processo torna-se mais demorado, pois é necessário efetuar mais testes entre os *pixels* vizinhos para definir o caminho da ferramenta.

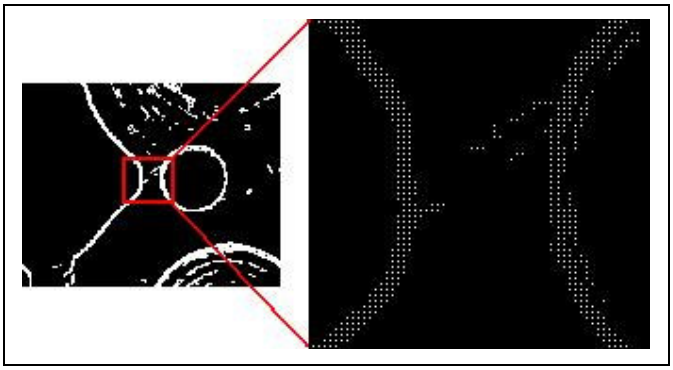
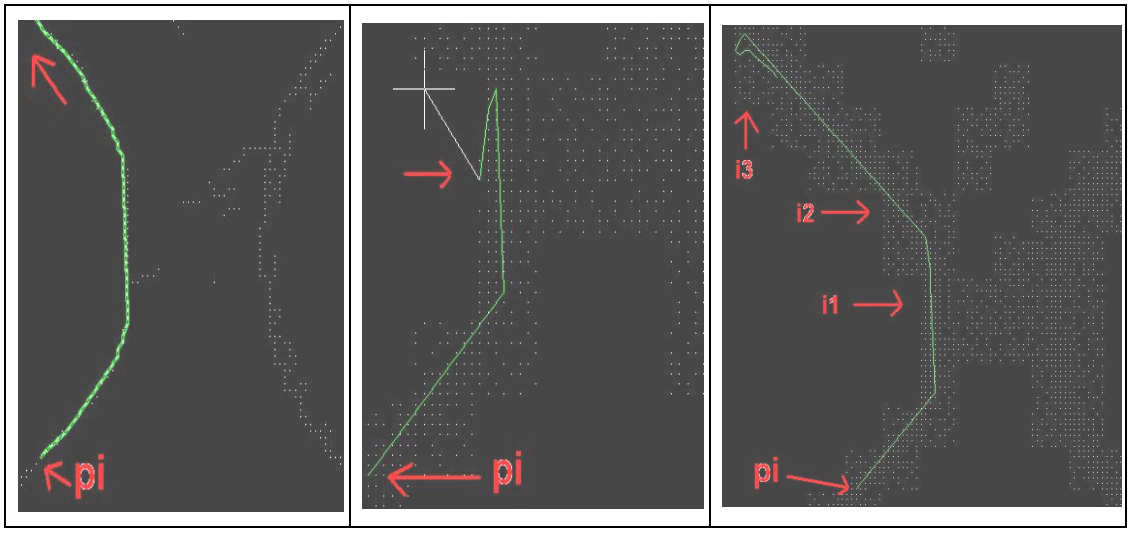


Figura 55 – Importação da imagem sem o afinamento

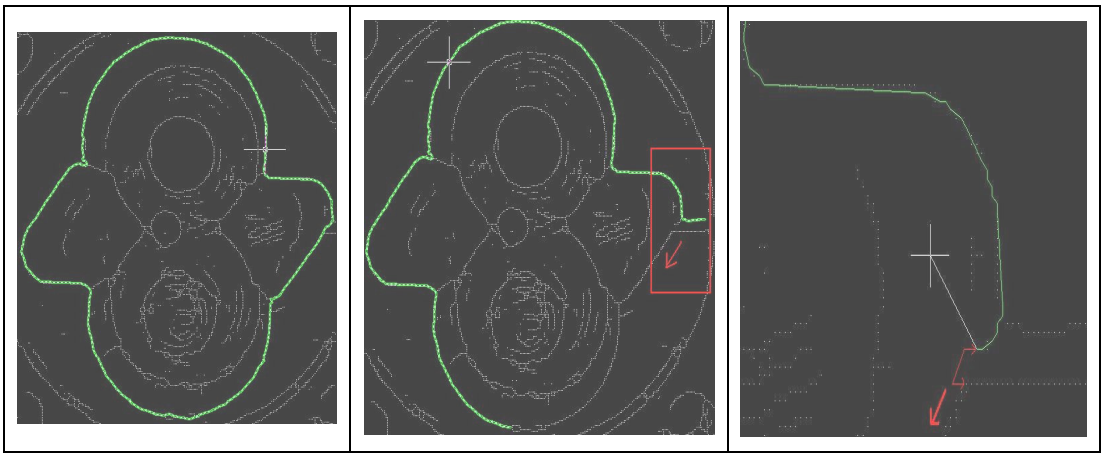
A diferença na aplicação do algoritmo proposto para os dois casos pode ser vista na figura 56, onde *pi* indica o ponto inicial e *i1*, *i2* e *i3,* são intervenções do usuário. Observa-se que a figura 56 item (a), imagem segmentada com o afinamento, a trajetória da ferramenta foi executada sem a intervenção do usuário, e seu processamento consumiu menos tempo devido à pequena quantidade de pontos da borda. Na figura 56, itens (b) e (c) as imagens fazem parte do mesmo arquivo de coordenadas, ou seja, da mesma peça e segmentada sem o afinamento. Nelas, foi necessário à intervenção do usuário por três (3) vezes, para corrigir o sentido da trajetória. Em alguns casos, figura 56 item (b) foi necessário retornar a trajetória e escolher novamente um ponto para dar o segmento da mesma. Também, o tempo de processamento foi maior, pois ao contrário da figura 56 item (a) esta possui vários pontos que compõem a borda, necessitando muitas comparações entre os vizinhos para realização do cálculo.



a b c

Figura 56 – Diferença cálculo da trajetória em imagens com afinamento e sem afinamento

Após iniciar a identificação da trajetória, através da especificação pelo usuário do ponto inicial e o segundo ponto da mesma, para indicar o sentido de deslocamento da ferramenta, o processo é interrompido nas seguintes situações: volta ao ponto de origem – contorno fechado, figura 57 item (a); pontos indesejados desviam a trajetória do seu caminho lógico, figura 57 item (b); ou a borda apresenta descontinuidade por problemas na qualidade da imagem, figura 57 item (c). No primeiro caso, o sistema encerra o processamento automaticamente. Para os outros, é solicitado ao usuário o procedimento a adotar através de três opções: encerrar o processamento; retornar a ferramenta pela trajetória gerada até o ponto de interrupção; ou especificar um novo ponto onde o algoritmo tenha condições de continuidade.



a b c

Figura 57 – Cálculo da trajetória da ferramenta a) contorno fechado; b) desvio da trajetória; c) descontinuidades nos pontos que compõem a borda

Finalmente, após a execução do cálculo da trajetória, é efetuada a geração do programa CNC. O código gerado representa a geometria da peça cujos comandos devem ser enviados à máquina de comando numérico para reprodução da peça. A figura 58 apresenta o código CNC inicial e final que representa a trajetória gerada na figura 57 item (a). Como o ajuste dos pontos da imagem ocorre pela formação de retas no CAD, o programa para a máquina de comando numérico tem que ser elaborado através de uma função de integração CAD/CAM. Nesse trabalho, essa função foi implementada através da exploração de recursos da linguagem de programação disponível no CAD para desenvolvimento de aplicativos. Esses comandos são destinados especificamente para o acesso ao banco de dados do CAD e através deles o sistema identifica os pontos iniciais e finais de cada segmento de reta do desenho, na seqüência de sua criação.

Na programação CNC, qualquer movimento da ferramenta em uma linha reta é denominado “interpolação linear” e tem como código os caracteres “G1” na linguagem padronizada. No programa codificado, cada bloco contendo uma interpolação linear no plano XY, o ponto especificado é o ponto meta. O ponto de origem não é informado porque é a posição atual da ferramenta.

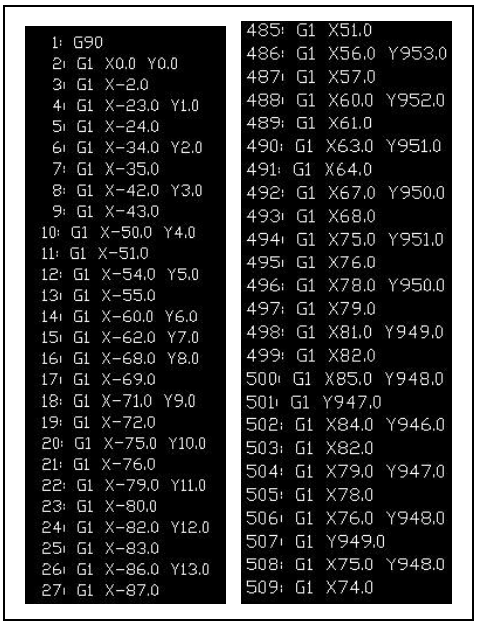


Figura 58 – Geração do código CNC

Com a geração do código CNC termina a etapa de processamento do CAD, com isso, é possível efetuar a simulação comprovando que a confecção da peça irá ser executada com sucesso, e após pode-se enviar o código CNC a máquina ferramenta para a reprodução da mesma, contemplando o processo de automação buscado neste trabalho.

5.2 Aplicação da Técnica *Multi-flash*

Para executar a técnica *multi-flash* é necessária a aquisição de imagens de mesmo tamanho, com aplicação dos flashes nos lados superior, direito, esquerdo e inferior, gerando sombras nas partes contrárias dos flashes. A partir das imagens, é observada a geometria

epipolar da sombra, efetuada aplicação de um dos operadores na detecção e aplicação do *threshold* automático para identificar as bordas da imagem.

No final são geradas três imagens, a primeira representa a intensidade máxima das três, na segunda é gerado o mapa de bordas da imagem ou mapa de profundidade, e a terceira é o resultado da segmentação *multi-flash*, as bordas da imagem. A figura 59 ilustra as quatro imagens adquiridas com o método de simulação *multi-flash*.

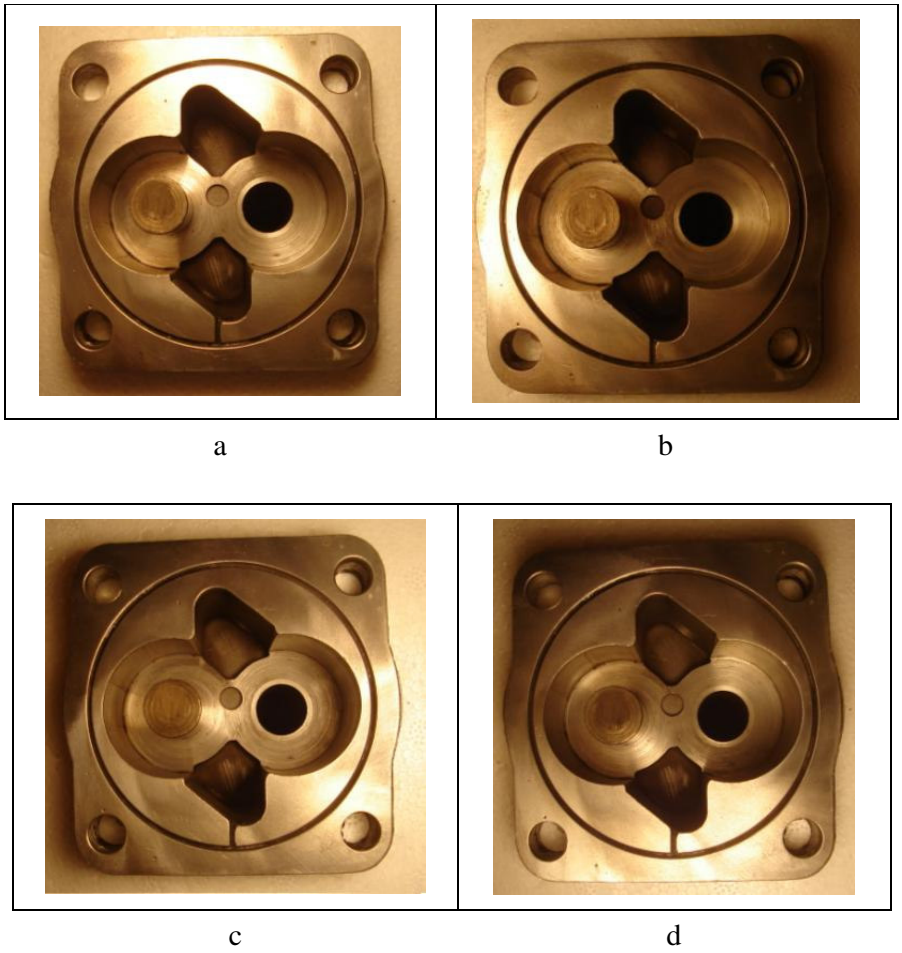
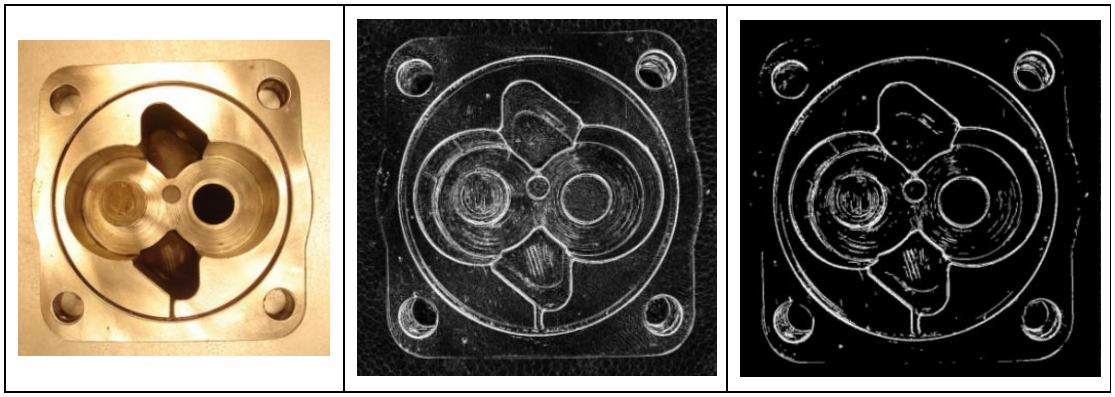


Figura 59 – Aquisição de imagens *multi-flash.* a) flash superior, b) flash esquerdo, c) flash direito, d) flash inferior

A aplicação de luzes nos lados da imagem simula a máquina *multi-flash* criado por Ramesh et. al (2004). Entretanto observam-se problemas com essa simulação. A geração das sombras não ocorre de maneira homogênea, ou seja, comparando a figura 59 itens (a) e (b), por exemplo, na imagem b, o lado que gerou a sombra, uma parte maior da peça ficou mais escura, cuja iluminação ficou quase fora do objeto (peça), diferentemente da imagem em (a). O problema de não homogeneidade acarreta uma deficiência na detecção final das bordas.

A seguir, a figura 60, apresenta o resultado da segmentação, onde foi gerada a imagem com intensidade máxima, a imagem com o mapa de profundidade e a imagem com as bordas detectadas.

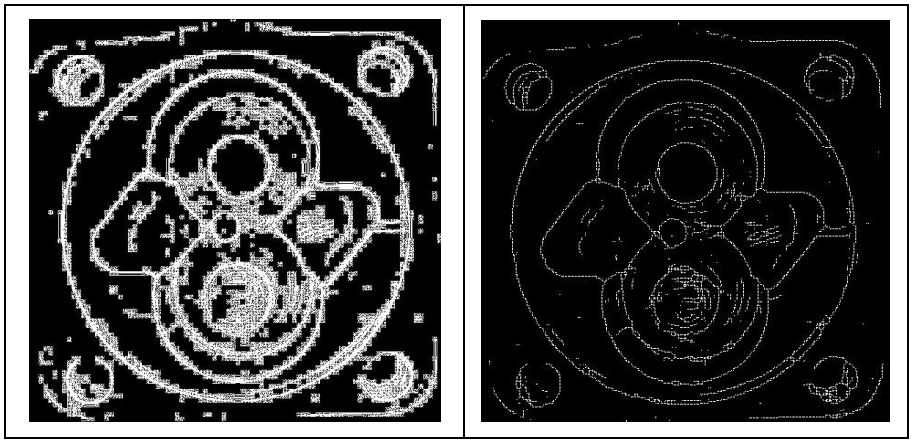


a b c

Figura 60 – Imagens resultado da técnica *multi-flash*. a) Intensidade máxima, b) Mapa de profundidade, c) bordas detectadas

Devido aos problemas na aquisição das imagens, os contornos externos da peça não foram totalmente detectados figura 60 item (c). Entretanto, sua aplicação gerou bons resultados, ao conseguir detectar os contornos internos da peça, levando em conta que a simulação foi efetuada sem equipamentos adequados. Para melhorar estes resultados, é necessária a utilização da câmara *multi-flash*.

A partir da figura 60 item (c), foram importadas as coordenadas para o CAD, da imagem que a técnica *multi-flash* detectou e outra após a aplicação do afinamento. A figura 61 mostra os resultados.

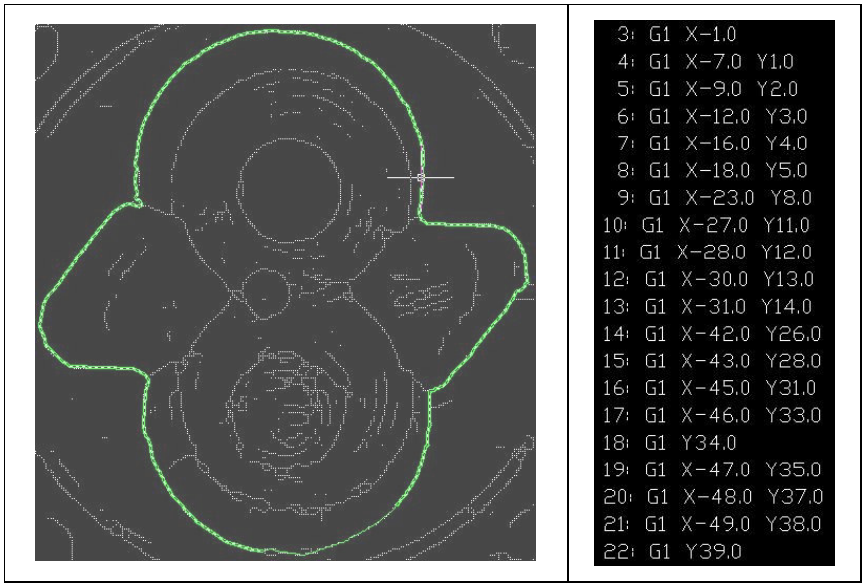


a b

Figura 61 – a) Peça importada da técnica *multi-flash*, b) peça importada da técnica *multi-flash* com afinamento

Na figura 61 observa-se a diferença após a aplicação do afinamento. Para o cálculo da trajetória da ferramenta é conveniente utilizar a imagem da figura 61 item (b), pois a anterior, 61 item (a), demandaria um maior tempo de processamento para definir o caminho da ferramenta.

A figura 62 itens (a) e (b), ilustram o cálculo da trajetória e parte do código CNC gerado pelo CAD.



a b

Figura 62 – a) Cálculo da Trajetória, b) Geração do Código CNC

A figura 62 exibe o resultado final do trabalho realizado. Entretanto muitas pesquisas e aprimoramentos podem ser efetuados a partir desse estudo, buscando melhorar cada vez mais a segmentação para melhorar o cálculo da trajetória e geração do código CNC.

Avaliando as duas abordagens de segmentação, constata-se que os métodos clássicos de detecção como os de Frei-and-Chen, Sobel, Prewitt, não apresentam bons resultados em imagens de diferentes tipos de bordas, ou seja, em imagens que podem conter tanto bordas de profundidade quanto de orientação, tais detectores não localizarão corretamente ambos os tipos de borda. Entretanto a utilização da técnica *multi-flash* permite detectar diferentes tipos de bordas a partir da combinação e fechamento destas bordas.

A geração do código CNC a partir do cálculo da trajetória da ferramenta contempla as etapas finais dos resultados, cuja implementação foi efetuada utilizando diferentes métodos e ferramentas contemplando um processo viável para automação industrial.

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

A segmentação é uma etapa fundamental no processamento de imagens, cuja técnica envolve diferentes métodos que permitem extrair diversas características importantes para interpretações das imagens, como a classificação e detecção de bordas.

As bordas delimitam o tamanho das superfícies dos objetos e fornecem informações importantes na extração das características da cena. Neste trabalho a detecção de bordas permitiu identificar a geometria dos objetos segmentados, neste caso, a peça, onde que a partir desses contornos criaram-se métodos para a geração do código CNC.

Uma das partes relevantes para obter bons resultados deve ser observada no momento da aquisição da imagem. Esta deve representar a peça o mais próximo possível da real, sem distorções e ruído, sem sombras ou iluminação em excesso.

As técnicas aplicadas, tanto com os operadores tradicionais como Sobel, Frei-and-Chen, Prewitt, quanto à *multi-flash*, apresentou resultados satisfatórios, porém alguns contornos não foram corretamente detectados, tanto com o uso dos operadores tradicionais quanto com o uso da técnica *multi*-flash. Ambos geraram partes que pertencem às bordas e partes que não pertencem às bordas, bem como não conseguiram detectar partes onde ocorreram problemas de iluminação e sombreamento.

Já a técnica *multi-flash* não conseguiu gerar corretamente os contornos externos da peça. Este fato ocorreu devido à má iluminação efetuada no momento da aquisição das imagens. Entretanto, a simulação efetuada com o uso de iluminação foi deficiente no processo de detecção de contornos. Porém a utilização de aparelhos adequados, como a câmara *multi-flash* permite resultados muito superior sendo a técnica que mais se aproxima na detecção de contornos real da peça.

A aplicação do afinamento na imagem permite que sejam gerados menos pontos no arquivo de coordenadas, o que resulta num menor tempo de processamento com menos intervenções do usuário para o cálculo da trajetória no Autocad.

As operações de cálculo da trajetória e geração do código CNC, efetuadas no Autocad, finalizam o processo de automação. Correspondem, mesmo que em estudos inicias, um grande avanço na busca pela automação de reprodução de peças sem a necessidade de programação do código CNC, ou seja, a utilização de técnicas de processamento de imagens e a geração automática de programas CNC, sem a intervenção do programador.

A possibilidade de importação das características de interesse a sistemas CAD, demonstra que o uso da engenharia reversa para obtenção do resultado final foi alcançado.

Sugestões para trabalhos futuros podem ser efetuadas na aplicação da técnica *multi-flash*, onde sugere-se a implementação em Java do código desenvolvido no Matlab®, eliminando a necessidade de conexão do Matlab® para executar a técnica *multi-flash*. Além disso, um novo protótipo pode ser desenvolvido com o objetivo de efetuar o cálculo da trajetória da ferramenta e geração do código CNC diretamente no *JImage Automation*, sem a necessidade de importação das características da geometria da peça no AutoCAD, transformando uma ferramenta completa para geração do código CNC.

Também, na aquisição das imagens, a utilização da máquina *multi-flash* elaborada por Ramesh, elimina a simulação efetuada por luzes que ocasiona problemas na detecção de contornos da peça e melhora consideravelmente os resultados.

Finalmente, muitos estudos podem ser elaborados na área de segmentação e melhoramento da ferramenta, além de criação de novas técnicas de cálculo da trajetória da ferramenta e geração do código CNC. Principalmente, estudos na detecção de contornos usando a técnica *multi-flash*, que apesar de ser utilizada através de uma simulação, conseguiu detectar contornos que muitas vezes não são detectados com a aplicação dos operadores tradicionais.

7 REFERÊNCIAS BIBLIOGRÁFICAS

ALBUQUERQUE**,** Márcio Portes de. **Visão por computador**. Centro Brasileiro de Pesquisas Físicas. Disponível em: http://www.cbpf.br/cat/pdsi/visao/index.html. Acessado em: outubro de 2007. 2000.

ÁLVARES, Alberto José; SILVA, Jones Yudi A. da; CORRÊA, Magno Batista; FERREIRA, João Carlos Espíndola. **WebCADbyFeatures: Desenvolvimento de um CAD baseado em features para projeto de peças cilíndricas via Internet**. 18th International Congress of Mechanical Engineering. Ouro Preto: COBEM 2005.

BARRETO, Marcos E.; SCHLEMER, Elgio. **Detecção de bordas com processador GAPP**. Disponível em: http://www.inf.ufrgs.br/procpar/disc/cmp135/trabs/barreto/t2/gapp.html. Acesso em: 22/05/06. 2004.

BASTOS, Vanessa Parodes. **Técnicas de segmentação de Imagens para recuperação de informações visuais**. Disponível em: http://atlas.ucpel.tche.br/~vbastos/pi.htm. Acessado em setembro de 2007. 2001.

BRILLINGER, Luiz Henrique. **Esqueletonização e Reconhecimento de Formas**. Técnicas de Esqueletonização. Disponível em: http://www.inf.ufsc.br/~visao/esqueleto.html. Acessado em: outubro de 2007. 1998.

CAGNIN, Maria Istela. **PERFAIT: Uma contribuição para a reengenharia de software baseada em linguagens de padrões e frameworks**. Tese de Doutorado apresentado ao Instituto de Ciências Matemáticas e de Computação – ICMC/SP. USP – Universidade de São Paulo - São Carlos. 2005.

CARVALHO, João Marques de; BRITO, Sérgio Ferreira de. **Sistemas de Processamento Digital de Imagens para fins Didático/Científico: Estudo, Seleção e Implementação de Algoritmos de Segmentação**. Relatório final de Iniciação Científica. LAPS – Laboratório de Automação e Processamento de Sinais. Universidade Federal de Paraíba. 1998.

CAVALCANTI, Sedecias. **Esqueletonização**. Técnicas de Esqueletonização. Disponível em: http://www.inf.ufsc.br/~visao/esqueleto.html. Acessado em: outubro de 2007. 1999.

CHENG, Cheok Yan. **Digital image processing application using Java Advanced Image**. Disponível em: http://www.geocities.com/yccheok81/zhangsuenthinning/index.html.

Acessado em: dezembro de 2007. 2004.

CHIKOFSKY, E. J.; CROSS, J.H. **Reverse engineering and design recovery: A taxonomy**. IEEE Software, 7(1): p.13-17. 1990.

COSTA, D. D. da; PEREIRA, A. G. **Desenvolvimento e avaliação de uma tecnologia de baixo custo para programação CNC em pequenas empresas**. Produção, v. 16, n. 1, p. 048-063, jan/Abc 2006. Universidade Federal do Paraná. Disponível em: http://www.scielo.br/pdf/prod/v16n1/a05v16n1.pdf>. Acessado em: 18 nov. 2006.

COSTA, R. M. **Um método de Engenharia Reversa para Auxiliar a Manutenção de Software.** Dissertação de Mestrado. ICMC – USP. São Carlos. 100p 1997.

DASCHBACH, Abella; McNichols. **Reverse Engineering: A Tool for process planning**. 17th Internacional Conference on Computers and Industrial Engineering. Elsevier Science. Vol. 29, No. 1-4, pp. 637-640. 1995.

DICKIN, Peter. **Reverse Engineering regains popularity**. IEEE Review**.** Vol. 42, Issue: 5, pp. 213 –S1-S4. 1996.

FACON, Jacques. **Processamento e Análise de Imagens**. Curso de Mestrado em Informática Aplicada. Pontifícia Universidade Católica do Paraná. Disponível em: http://www.ppgia.pucpr.br/~facon/IndexPrincipalBrCursos.htm. Acessado em: maio de 2007. 2002.

FACON, Jacques. **Algoritmo de Afinamento de Holt**. Programa de Pós-Graduação em Informática Aplicada – PPGIA. Pontifícia Universidade Católica do Paraná. Disponível em: http://www.ppgia.pucpr.br/~facon/IndexPrincipalBrAfinamento.htm. Acessado em: novembro de 2007. 2001.

FALCÃO, Alexandre Xavier. **Fundamentos de Processamento de Imagem Digital**. Tópicos em Processamento de Imagens IC – UNICAMP. Disponível em: http://www.dcc. unicamp.br/~cpg/material-didatico/mo815/9802/curso/curso.html. Acessado em: novembro de 2007. 2003.

FELGUEIRAS, Carlos Alberto. **Processamento Digital de Imagens – Filtragens Especiais**. Instituto Nacional de Pesquisas Espaciais – INPE / DPI. Disponível em: http://www.dpi.inpe.br/~carlos/Academicos/Cursos/Pdi/pdi_filtros.htm. Acessado em: junho de 2007. 2006.

FERNÁNDEZ, Manuel Eduardo Loaiza. **Calibração de câmera baseado em algoritmo de detecção de cantos e inserção de objetos virtuais dentro da imagem de vídeo capturada**. Trabalho final de Realidade Aumentada e Colaborativa. Disponível em: http://www.tecgraf.puc-rio.br/~mgattass/ra/trb03/ManuelLoaiza/. Acessado em: agosto de 2007. 2004.

FERNEDA, Amauri Bravo. **Integração Metrologia, CAD e CAM: uma contribuição ao estudo de Engenharia Reversa**. Dissertação de Mestrado, Escola de Engenharia da Universidade de São Paulo – USP. São Carlos. 1999.

FILHO, Ogê Marquês; NETO, Hugo Vieira. **Processamento Digital de Imagens**. Rio de Janeiro: Brasport. p. 25 – 182. 1999.

FONSECA, L. M. Garcia. **Processamento Digital de Imagens** (Sensoriamento Remoto), Leila Maria Garcia Fonseca, INPE – São José dos Campos, Junho de 2000.

FRANÇA, José Alexandre. **Thinning – A Linha Essencial**. Técnicas de Esqueletonização. Disponível em: http://www.inf.ufsc.br/~visao/esqueleto.html. Acessado em: outubro de 2007. 2001.

GATTASS, Marcelo. **Imagem digital: conceitos e algoritmos**. Disponível em http://www.tecgraf.puc-rio.br/~mgattass/cg/cg.html. Acessado em: setembro de 2007. 2007.

GONZALEZ, Rafael. C, WOODS, Richard E. ***Digital Image Processing***. Prentice Hall. 2nd Ed. 2002.

GRANDO, Neusa. **Segmentação de Imagens Tomográficas Visando a Construção de Modelos Médicos**. Dissertação de Mestrado do Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial. CEFET-PR – Centro Federal de Educação Tecnológica do Paraná, 2005.

GUJ. **Introdução à programação gráfica em Java com swing**. Disponível em: <http://www.guj.com.br/jva/tutorial.artigo.38.1.guj>. Acessado em: Agosto de 2007. 2006.

HERTZMANN, Aaron. **Introduction to 3D Non-Photorealistic Rendering: Silhouettes and Outlines**. Non-Photorealistic Rendering (SIGGRAPH ’99 course notes) Disponível em: http://mrl.nyu.edu/~hertzman/hertzmann-intro3d.pdf. Acessado em: janeiro de 2008. 1999.

IMService ©. **DeskCNC® Windows Based CAM software and a powerful CNC Machine Control**. Disponível em: http://www.imsrv.com/deskcnc/. Acessado em: Janeiro de 2008. 2003.

JANKOWSKI, Mariusz. **Java and the Digital Image Processing Application Package.** Disponível em: http://www.usm.maine.edu/~mjankowski/docs/mdc2001/jai.html. Department of Electrical Engineering University of Southern Maine. Acessado em: novembro de 2006. 2002.

JARDIM, Eric. **Non-Photorealistic Rendering using Lines**. Página de Projetos – IMPA - Instituto Nacional de Matemática Pura e Aplicada. Disponível em: http://w3.impa.br/~lvelho/i3d07/demos/. Acessado em: janeiro de 2008. 2007.

KOLLIOPOULOS, Alexander; WANG, Jack M.; HERTZMANN, Aaron. **Segmentation-Based 3D Artistic Rendering**. Eurographics Symposium on Rendering. Disponível em: http://www.dgp.toronto.edu/~alexk/segegsr.html. Acessado em: janeiro de 2008. 2006.

LAUREGA, Luciane. **MeSegHi: Um Método de Segmentação para o Processamento Linear e Não-Linar de imagens**. Dissertação de Mestrado – Programa de Pós-Graduação em Engenharia de Produção (PPGEP) – Universidade Federal de Santa Maria. 2006.

LIMA, Cristiane Brasil. **Engenharia reversa e prototipagem rápida estudo de caso.** Dissertação de Mestrado apresentada ao Programa de Pós Graduação em Engenharia Mecânica da Universidade Estadual de Campinas. São Paulo. 2003.

MARROQUIM, Ricardo Guerra. **Non-Photorealistic Rendering**. Disponível em: http://orion.lcg.ufrj.br/cg2/downloads/. Acessado em: dezembro de 2007. 2007.

MARTINEZ, Gabriela. **Depth Edges Detection Using Multiflash Imaging**. Report in Image Processing for Vision and Graphics. Disponível em: http://w3.impa.br/~lvelho/ip05/demos/p2/. Acessado em: Outubro de 2007. 2005.

MASCARENHAS N.D.A. e VELASCO F. R. D. **Processamento Digital de Imagens**, IV Escola de Computação, São Paulo, SP, Julho de 1994.

MATHWORKS. **Video and Image Processing Blockset**. The MathWorks. Disponível em: http://www.mathworks.com/access/helpdesk/help/toolbox/vipblks/ref/2dconvolution.html. Acessado em: janeiro de 2007. 1994.

MIRANDA Ricardo Florêncio. **Construção de uma linguagem baseada em Java**

denominada J4CV para visão computacional. Trabalho de Conclusão de Curso. Disponível em: http://www2.unoeste.br/~chico/FIPP/projetos/projeto2006/Monografia_J4CV_Ricardo_2006.pdf. Acessado dia 17 de 06 de 2007.

MÜLLER, Stefan. **JMatLink – Connect Matlab and Java**. Disponível em: <http://jmatlink.sourceforge.net/index.php>. Acessado em: maio de 2007. 2005.

OLIVEIRA, Flávio Z. de; RUBIO, Juan Carlos Campos. **Cenário de Manufatura Integrada para Produção de Moldes e Matrizes**. IFM – Instituto Fábrica do Milênio. Disponível em: http://www.ifm.org.br/fase/congresso/inscritos/teste2.php?id_trabalho=439. Acessado em: dezembro de 2007. 2005.

PEREIRA, **Desenvolvimento e Avaliação de um Editor para Programação CN em Centros de Usinagem**. Dissertação de Mestrado do Programa de Pós-Graduação em Engenharia Mecânica da Universidade Federal do Paraná. 2003.

PRESSMAN, R. S. **Engenharia de Software**. Makron Books. 2002.

QUEIROZ, José Eustáquio Rangel de, GOMES, Herman Martins. **Introdução ao Processamento Digital de Imagens**. Revista RITA. Volume XIII. Número 2. Disponível em: http://www.inf.ufrgs.br/~revista/. Acessado em: setembro 2007. 2006.

RAMESH, R., TAN, K.-H, FERIS, R., YU, J., AND TURK, M. 2004 **Non-photorealistic Camera: Depth Edge Detection and Stylized Rendering using *Multi-flash* Imaging**. ACM trans. Graph. 23, 3, 679-688. 2004.

RUGABER, S. **Program Comprehension for Reverse Engineering.** In: AAAI Workshop on AI and Automated Program Understanding. San Jose, California. P.106-110. Disponível em: http://www.cc.gatech.edu/reverse/papers.html. 1992.

RUSS. **The image processing handbook**. São Paulo: CRC Press LLC, 1999.

SANTOS, Gilmário Barbosa dos. **Processamento de Imagens – Segmentação**. Disponível em: http://www.joinville.udesc.br/processamentodeimagens/segmentacao.html. Acessado em: outubro de 2007. 2001.

SEARA, Daniela M. **Visão Geral de Detecção de Bordas**. Disponível em: http://www.inf.ufsc.br. Setembro. Acesso em: 06/09/06. 1998.

SILVA, Conceição M. **Tipos de Sistema de Informação: Automação e SPO.** Disponível em: http://www.infonet.com.br/caad/professores/conceicao/sig. Acessado em: agosto 2007. 2003.

SILVA, A. D.; CARPES, F. P.; BITTENCURT, W. S.; MOTA, C. B. **Image processing applied to CNC programming in drilling process**. 18th International Congress of Mechanical Engineering. Ouro Preto: COBEM 2005.

SILVA, Alexandre Garcia Costa da; ÁLVARES, Alberto José. **Aspectos Relevantes para a Implementação de um Ambiente Computacional CAD/CAPP/CAM Orientado a Processo de Furação**. Disponível em: HTTP. Acessado em Dezembro de 2007. 2000.

Softcover International Limited ©. **Scan2CAD® raster to vector conversion software**. Disponível em: http://www.softcover.com/. Acessado em: janeiro de 2008. 2008.

SUN MICROSYSTEMS, **Programming in Java Advance Imaging**. Disponível em: <http://java.sun.com/products/java-media/jai/forDevelopers/jai1_0_1guide-unc/JAITOC.fm .html>. Acesso em: novembro de 2006. 1999.

TAKEDA, Fábio Bento. **Sistema para Tomada de decisão baseado na geração de imagens estereoscópicas e reconhecimento de padrões circulares.** Dissertação de Mestrado do Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos – UFSC. 2000.

Trix Systems Inc ©. **TracTrix™ - the CAD user's raster-vector toolkit**. Disponível em: http://www.trixsystems.com/tractrix.html. Acessado em: fevereiro de 2008. 2008.

VextraSoft ©. **Vextractor, Raster to vector conversion tools**. Disponível em: http://www.vextrasoft.com/index.htm. Acessado em: fevereiro de 2008. 2007.

WANGENHEIM, Aldo Von. **Reconhecimento de Padrões - 6. Gerando Padrões: Análise de Sinais e Imagens**. Disponível em http://www.inf.ufsc.br/~patrec/imagens.html. Acessado em 22/11/2007. 2006.

WANGENHEIM, Aldo Von**. Reconhecimento de Padrões.** 6. Gerando Padrões: Análise de Sinais e Imagens. Disponível em: http://www.inf.ufsc.br/~patrec/imagens.html. Acessado em: novembro de 2007. 2006.

ZHANG, T. Y. and SUEN, C. Y. **A fast parallel algorithm for thinning digital patterns**. Communications of the ACM. Volume 27. Number 3. Disponível em:

http://www.geocities.com/yccheok81/zhangsuenthinning/p236-zhang.pdf. Acessado em: dezembro de 2007. 1984.

ZUBELLI P. Jorge; SILVA Moacyr; PASTORE H. Dayse. **Tutorial para MATLAB – Curso de Métodos Matemáticos em Finanças I.** IMPA – Instituto Nacional de Matemática Pura e Aplicada. Disponível em: http://w3.impa.br/~zubelli/tutorial. Acessado em: janeiro de 2008. 2000.