Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Informática
Pós-Graduação em Ciência da Computação

Determinando a Posição e a Orientação da Mão Através de Imagens de Vídeo

Eduardo Costa Lopes

Dissertação apresentada como requisito parcial à obtenção do grau de mestre em Ciência da Computação

Orientador: Márcio Serolli Pinho

Porto Alegre, Janeiro de 2005

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
Reconhecido pelo Parecer No. 930/98.C.N.E. Homologação Publicada no D.O.U. de 30/12/98.

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada **"Determinando a Posição e a Orientação da Mão Através de Imagens de Vídeo"** apresentada por **Eduardo Costa Lopes,** como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Área de Computação Científica, aprovada em 20/01/2005 pela Comissão Examinadora:

Prof. Dr. Márcio Serolli Pinho — PPGCC/PUCRS
Orientador

Profa. Dra. Isabel Harb Manssour – FACIN/PUCRS

Prof. Dr. João Batista Souza de Oliveira - PPGCC/PUCRS

Profa. Dra. Carla Maria Dal Sasso Freitas — UFRGS

Homologada em 25/09/06, conforme Ata Nº. 022 pela Comissão Coordenadora.

Prof. Dr. ***Avelino Francisco Zorzo***
Coordenador.

AV. IPIRANGA, 6681 - PRÉDIO 16/SALA 106
90619-900 - PORTO ALEGRE -RS -BRASIL
TELEFONE: 55 51 33203611 FAX 55 51 3320.3621
http: //www.inf.pucrs.br e-mail: ppgcc@inf.pucrs.br

Av. Ipiranga, 6681 - Prédio 16
CEP 90619-900 - Porto Alegre - RS - Brasil
http://www.pucrs.br/biblioteca

Fone: 0 (xx)51 3320-3544
Fax: 0 (xx)51 3320-3548
e-mail: bceadm@pucrs.br

Agradecimentos

Gostaria de agradecer o apoio e confiança de todos aqueles que me ajudaram e incentivaram para a conclusão deste trabalho. Em especial agradeço a minha família, à minha Mãe (Nelce M. C. Lopes), meu Pai (Adão B. Lopes), ao meu Tio (Edmundo Lopes), por sempre apoiarem as minhas decisões, mesmo que isso signifique permanecer longe deles por muito tempo. À minha amada, Marcela, por todo seu amor e carinho.

Como não poderia deixar de mencionar, também agradeço aos meus colegas, tanto da FURG como da PUCRS, por estarem sempre dispostos a ajudar, e pelo companheirismo deles nas horas difíceis.

Ao orientador Prof. Dr. Márcio Serolli Pinho que, durante o ano de 2004, me incentivou, apoiou e sempre esteve à disposição durante a pesquisa, bem como aos demais membros do GRV (Grupo de Realidade Virtual), que sempre apresentaram sugestões plausíveis durante o andamento deste projeto e pelas experiências de vida que estas pessoas proporcionaram.

Ao Prof. Antônio Scuri, pelo acompanhamento da pesquisa e por suas informações extremamente relevantes para a realização deste projeto.

À CAPES, pelo apoio financeiro, sem o qual este trabalho não seria realizado.

Resumo

Atualmente, a tecnologia de Realidade Virtual permite utilizar computadores com um grau de interação superior às interfaces tradicionais, baseadas apenas no teclado e no *mouse*, através de dispositivos que permitem inserir o usuário em um ambiente gerado em computadores. Neste ambiente virtual, o usuário pode visualizar em três dimensões os objetos que o compõe, com a possibilidade de interagir com os mesmos de maneira semelhante ao que ocorre no mundo real. O grau de interatividade em um ambiente virtual é influenciado pela capacidade do ambiente de rastrear determinadas partes do corpo, como a cabeça, a mão, ou até mesmo o corpo inteiro. Também é importante que este ambiente proporcione a execução de determinadas operações, que permitam o usuário interagir com um objeto virtual como se fosse um objeto real, promovendo a sensação de que o primeiro estava imerso em outra realidade. Infelizmente, o custo de tais equipamentos e a quantidade de fios necessários para conectá-los ao corpo do usuário, além de outras restrições, limitam a utilização da Realidade Virtual na vida diária. Este trabalho objetiva apresentar uma alternativa ao rastreador de posição e orientação utilizado para rastrear a mão. A sua contribuição é fazer uso das técnicas de Processamento de Imagens e Visão Computacional para implementar um rastreador de mão baseado em imagens de câmeras de vídeo. Para isto, o projeto foi divido em três fases distintas. A primeira fase detecta a mão em uma imagem através da segmentação de pele. Nesta fase, quatro algoritmos de segmentação de pele são implementados e vários testes são realizados, utilizando dois espaços de cores e dois modelos de cores. A segunda fase determina a posição da mão. Nesta fase dois algoritmos são implementados e testados. Na terceira a fase, a orientação da mão é determinada através de uma técnica conhecida na Visão Computacional denominada Momentos de Imagem. Em seguida, através da análise do contorno da mão, algumas características são detectadas, como as pontas dos dedos, os vales entre os dedos e o pulso, as quais podem ser utilizadas para calcular a posição e orientação da mão em 3D. Ao longo deste volume descreve-se em maiores detalhes cada fase de desenvolvimento do projeto juntamente com as técnicas utilizadas e seus respectivos resultados.

Abstract

Currently, the use of Virtual Reality technology has opened the possibility to use computers with an interaction level superior to the traditional interfaces based on keyboards and mouse. This is achieved trough devices that allow to insert the users vitually in a computer generated environment. In this virtual environment the user can visualize and manipulate, in three dimensions, virtual objects that are part of it, with the possibility of interacting with them in a similar way like the real world. The degree of interaction in virtual environments is influenced by the capability of track certain parts of body, such as the head, the hand or even the whole body. Also, it is important that this environment enable the execution of certain operations that allow a user interact with a virtual object in real world fashion, providing the feeling that he or she is immersed in another reality. Unfortunately, the cost of such equipments and the amount of wires that need to be attached to the user body, among others restrictions, limit the use of Virtual Reality in everyday life. This work aims to present an alternative to the position and orientation tracker hardware used to track the user hand. Its contribution is to use Image Processing and Computer Vision techniques to implement a video image based hand tracker. To do so, the project is divided in three distinct phases. The first phase detects the hand in an image through skin color segmentation. In this phase, four skin segmentation algorithms are implement and many tests are realized with two skin models and two color space. The second phase calculates the hand position and two algorithms are implemented and tested. In the third phase, the hand orientation is obtained through a Computer Vision technique called Image Moments. After that, through the contour hand analysis, some features are detected, such as fingertips, valleys between the fingers and the wrist. These features can be used to get the 3D hand position and orientation. Along this text each development phase, the techniques used, and its respective results are described in detail.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Símbolos e Abreviaturas

IHC	Interação Humano-computador	1
RV	Realidade Virtual	1
VRT	*Virtual Reality Therapy*	2
VRML	*Virtual Reality Modeling Language*	3
VC	Visão Computacional	5
WIMP	*Window-Icon-Menu-Pointer*	7
RGB	*Red Green Blue*	12
HSV	*Hue Saturation Value*	12
LUT	*Lookup Table*	14
DCT	*Distance Chamfer Transform*	33
MT	Matriz de Transformação	61

Capítulo 1

Introdução

Atualmente a forma de interação mais comum entre humanos e computadores é realizada por meio de dispositivos físicos, como teclado, *mouse*, caneta óptica e etc. Poucas são as aplicações com capacidade de interpretar dados no domínio do som ou vídeo [48]. Entretanto a complexidade de certas aplicações cresce cada vez mais, tornando insuficientes as formas atuais de interação como, por exemplo, a visualização de uma quantidade massiva de dados em indústrias como a automobilística e em setores como a medicina [8], ou ainda, a manipulação de objetos 3D com o *mouse*, visto que este dispositivo tem seus movimentos limitados a 2D [52].

Entretanto, algumas pesquisas estão sendo realizadas para prover interfaces intuitivas e inteligentes de forma a melhorar a comunicação entre humanos e máquinas [6, 9, 29, 40, 46], tornando a interação entre os mesmos uma atividade mais natural. Um dos objetivos dos estudos em IHC (Interação Humano-Computador) é transformar o corpo humano ou parte dele, em dispositivos de entrada para o computador. A pesquisa nesta área, atualmente concentra-se no reconhecimento de gestos, reconhecimento de linguagens de sinais, rastreamento de mão, entre outros [58].

Em contrapartida, os progressos na área de Realidade Virtual (RV) permitem atualmente simular ambientes do mundo real, a partir de computadores. A principal idéia que move a RV é a possibilidade de unir em um ambiente virtual usuários humanos e computadores, interagindo da mesma forma como no mundo real. Um exemplo de tal integração seria a possibilidade de projetar e manipular um objeto virtual podendo vê-lo, manipulá-lo e analisá-lo, como se o mesmo existisse fisicamente, inclusive podendo sentir o objeto através da emulação do tato [31, 32].

O surgimento da tecnologia de RV inaugurou um novo paradigma na simulação e interação através de computadores. Um ambiente virtual pode ser considerado como uma interface visual em terceira dimensão. No entanto, novos problemas surgem, pois embora se atue em um mundo 3D, muitas pessoas têm dificuldade de utilizar os sistemas que tentam imitá-lo de alguma forma, pois muitas condições e restrições para a atuação no ambiente real atualmente não podem ser representadas em um ambiente virtual [8].

Embora a RV possua algumas limitações, várias áreas do conhecimento sofreram fortes influências com a utilização desta tecnologia. Riva [49] argumenta que os ambientes virtuais possibilitam pela primeira vez na história um meio que permite um grande entendimento da dinâmica dos processos cerebrais, bem como podem ser utilizados para tarefas de treinamento de novos

médicos, nas quais as simulações de situações reais de emergência são utilizadas para auxiliar o novato em medicina a tomar decisões certas em situações reais a partir de várias fontes de informação. Outro exemplo que se pode citar é o trabalho de Benes e Bueno [5], no qual um Simulador de Hepatectomia permite que virtualmente um médico-aprendiz treine exaustivamente a ressecção de fígado, Figura 1.1.

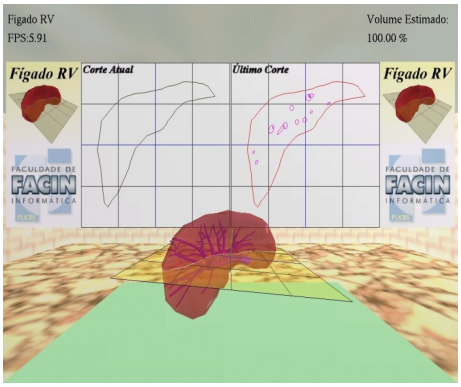


Figura 1.1: Simulador de Hepatectomia.

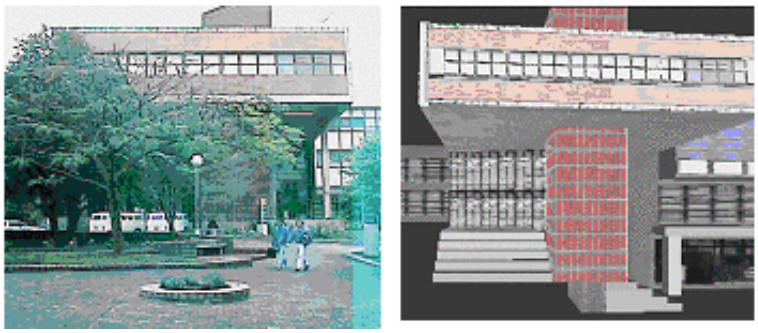
Em tratamentos psicológicos, atualmente a RV é geralmente utilizada para o tratamento de fobias [49], no qual o paciente é colocado em um ambiente virtual que simula situações que ajudam o indivíduo a superar seu problema. A VRT (*Virtual Reality Therapy*), tem sido utilizada com sucesso em terapias como medo de voar ou medo de alturas [42,43]. Maiores detalhes sobre o uso da VRT para tratamentos de distúrbios psicológicos podem ser encontrados em [51].

Em áreas como arquitetura e engenharia, algumas empresas e indústrias adotaram a RV como parte do processo de desenvolvimento de seus produtos, sob a justificativa de que esta tecnologia permite melhorar a qualidade da produção. Como exemplo pode-se citar a Embraer [17] que possui um centro de Realidade Virtual (CRV), que permite reduzir o tempo de desenvolvimento de novas aeronaves. O CRV está equipado com um avançado *hardware* gráfico, que garante aos engenheiros da empresa visualizar, em três dimensões, toda a estrutura de uma aeronave em fase de projeto, Figura 1.2. Além disso, o cliente pode visualizar o avião, na fase de produção, avaliar a configuração do mesmo e personalizar o produto conforme suas necessidades.

Figura 1.2: Usuário no CRV da Embraer.

No caso da arquitetura, a RV é utilizada para se ter uma idéia de como será uma determinada obra, principalmente como será o interior da mesma, sem a necessidade de construir uma maquete real. No Departamento de Expressão Gráfica da Faculdade de Arquitetura da UFRGS [18]

a realidade virtual foi utilizada para modelar os prédios de um campus universitário, com a possibilidade de navegação no interior dos mesmos, bem como permite acessar informações relativas às pessoas que freqüentam tais prédios e acervos bibliográficos, entre outros. Este ambiente virtual foi implementado através da linguagem VRML, e está disponível na Internet. Na Figura 1.3, tem-se um exemplo de modelagem de um prédio do campus da PUCRS (Figura 1.3(a)) através de uma maquete virtual (Figura 1.3(b)).



(a) Prédio. (b) Modelo virtual

Figura 1.3: Exemplo de modelagem de prédios [45].

Para que fosse viável tamanho grau de interação, novos dispositivos de entrada e saída foram desenvolvidos. Os principais dispositivos de entrada usados em RV são os rastreadores de movimento e as luvas de entrada de dados, vista na Figura 1.4.

Figura 1.4: Luva de entrada de dados.

A eficiência no rastreamento dos movimentos do usuário é um dos principais determinantes da maior ou menor sensação de imersão do usuário em um ambiente virtual. É a partir da captação dos movimentos do usuário que é possível interpretar comandos e executá-los de maneira adequada. Para este fim, utilizam-se os dispositivos de rastreamento, ou *tracking devices*, cuja principal função é fornecer a **posição** e/ou **orientação** de uma parte do corpo do usuário. Tais dispositivos baseiam-se em algum princípio da física e podem ser classificados nos seguintes tipos [45]:

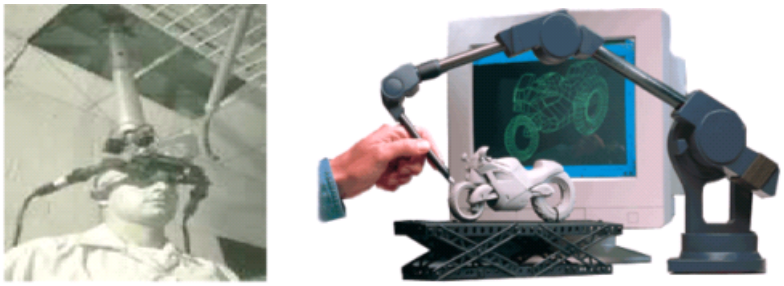
- **Rastreadores mecânicos:** são dispositivos articulados que rastreiam um determinado ponto do corpo do indivíduo, denominado ponto de referência, em relação a outro ponto fixo conhecido, tomado como base. Na Figura 1.5(a) tem-se um exemplo de um rastreador

mecânico, construído por Ivan Sutherland [55] que possibilita rastrear a cabeça do usuário, e na Figura 1.5(b) tem-se um rastreador da *MicroScribe* [38], utilizado para modelagem 3D de objetos;

- **Rastreadores Acústicos:** são rastreadores que localizam a posição de um determinado objeto por meio da reflexão do som. A posição é obtida através do cálculo de distância em função do tempo decorrido entre a emissão de um som e sua recepção, logo após ser refletido pelo alvo;

- **Rastreadores Magnéticos:** é o tipo de tecnologia de rastreador mais utilizada atualmente em ambientes virtuais. O princípio de funcionamento desses dispositivos é que quando um fio elétrico é submetido a um campo magnético, surge uma corrente elétrica induzida. Através de um receptor colocado no objeto a ser rastreado e da análise das correntes induzidas sobre ele é possível calcular a posição do objeto em relação a um emissor de campos magnéticos;

- **Rastreadores Ópticos:** são rastreadores que utilizam imagens para determinar a posição de um ponto. Existem dois tipos de rastreadores ópticos, aqueles que utilizam câmeras instaladas no ambiente que captam a imagem do usuário, obtendo a sua posição através da identificação de pontos marcados no corpo do mesmo e aqueles que utilizam câmeras acopladas no próprio corpo do usuário, identificando pontos marcados no ambiente.



(a) Rastreador de ponto fixo, [55].

(b) Rastreador para modelagem 3D, [38].

Figura 1.5: Exemplos de rastreadores mecânicos.

1.1 Motivação e Objetivos do Trabalho

A necessidade de equipamentos caros acoplados ao corpo limita muito a utilização da RV em tarefas do cotidiano. No caso da mão, que é o principal meio de manipulação de objetos no mundo real, a sua utilização em um ambiente virtual se dá com a utilização das luvas de entrada de dados, bem como de outros dispositivos utilizados para detectar sua posição e orientação.

Devido ao preço de tais equipamentos (conforme mostrado na Tabela 1.1), e à quantidade de fios que necessitam ser ligados a mão ou ao braço, causando um certo desconforto e limitando os

movimentos do usuário, é extremamente útil e economicamente interessante eliminar a necessidade de qualquer dispositivo acoplado às mãos, mas continuar a permitir ao usuário manipular qualquer objeto dentro do ambiente virtual de maneira semelhante ao que é feito no mundo real.

Tabela 1.1: Tabela de preços de alguns rastreadores fabricados pela *InterSense* [28], vendidos no Brasil pela empresa *Absolute Technologies* [1].

Modelo	**Preço US$**
InterTrax TM USB Precision Head/Hand Tracker	1.504,00
IS-900 Wired VWT Tracking System	22.950,00
IS-900 Wireless VWT Tracking System	44.700,00
IS-900 VET Virtual Environment Tracking System	38.300,00
IS-900 VET Virtual Environment Tracking System Wireless	60.100,00

Portanto, neste trabalho pretende-se substituir o rastreador de posição e orientação por uma ou duas câmeras que captam a imagem da mão e, a partir delas, obter a posição e orientação da mão. Para localizar a mão em uma imagem e separá-la do resto dos objetos podem ser utilizadas técnicas de Visão Computacional (VC). Em seguida, a posição e a orientação da mão podem ser calculadas por meio da localização de determinados pontos, denominados *pontos de controle*, localizados sobre a mão, ou ainda pode-se utilizar informações obtidas através da linha de contorno da mão.

Como o escopo deste trabalho não inclui o reconhecimento de poses da mão, qualquer configuração dos dedos e da própria mão é aceitável. Portanto, neste trabalho a determinação da posição e orientação da mão subdivide-se em três etapas:

- Segmentar a mão de uma imagem do mundo real;
- Determinar a posição 3D da mão;
- Determinar a orientação 3D da mão.

Embora alguns autores, como Rehg e Kanade [47], considerem o rastreamento visual (*visual tracking*), o ato de recuperar o estado da mão através de imagens, neste trabalho a palavra *rastrear* é utilizada para designar o ato de determinar a posição e a orientação da mão através de imagens, não importando o estado da mesma. O conceito de *Tempo Real*, neste trabalho, significa executar uma determinada função ou conjunto de funções tão rápido que o usuário não perceba que está havendo processamento nas imagens obtidas pela câmera. Além disso a palavra segmentar será utilizado no sentido de separar ou detectar, como por exemplo, segmentar a mão significa separá-la do fundo da imagem ou detectá-la em uma imagem.

1.2 Organização da Dissertação

Este documento é composto de 8 capítulos incluindo a Introdução e está organizado da seguinte forma:

- **No Capítulo 2** são mostrados alguns projetos relacionados na área de IHC e RV.
- **No Capítulo 3** são mostrados os algoritmos de segmentação de pele implementados para separar a mão do fundo da imagem. Em seguida, é fornecida uma análise em termos de tempo e qualidade da resposta de cada algoritmo.
- **No Capítulo 4** são mostrados os métodos utilizados para refinar a qualidade da segmentação de pele e localização de região da mão. Também são descritos dois algoritmos para obtenção da posição da mão, bem como a avaliação em termos de tempo e qualidade da resposta de cada algoritmo.
- **No Capítulo 5** é descrita a técnica denominada *Momentos de Imagens*, muito utilizada em VC para descrever propriedades geométricas de objetos presentes em imagens, que neste trabalho foi utilizada para calcular a orientação da mão.
- **No Capítulo 6** são descritas algumas técnicas para detectar características específicas na superfície da mão, como a pontas dos dedos, os vales entre os dedos e os pulsos.
- **No Capítulo 7** são descritas algumas técnicas que aproveitam os resultados obtidos na detecção de características da mão para resgatar informações em 3D e obter a posição da mão no espaço.
- **No Capítulo 8** são apresentadas as considerações finais e os possíveis trabalhos futuros.

Capítulo 2

A Mão como Dispositivo de IHC

A maior parte dos computadores pessoais possui algum tipo de interface gráfica com o usuário, todas elas baseadas no modelo WIMP(*Window-Icon-Mouse-Pointer*). Este modelo foi concebido através da metáfora da mesa de trabalho (*desktop*) [46], onde se tem uma mesa com objetos, documentos, planilhas, etc, que são manipulados através do *mouse*.

A diferença é que, na mesa de trabalho real, o usuário pode escrever seus textos utilizando uma caneta, ao mesmo tempo abrir um documento, guardar outros, usar uma calculadora ou qualquer outra atividade de uma rotina de trabalho normal. Tais atividades no computador requerem o uso do *mouse* para selecionar algum objeto e movê-lo. Caso existam vários documentos abertos em janelas, algumas deverão ser minimizadas, outras arrastadas até o usuário clicar e "pegar" o documento que deseja, enquanto que numa mesa real o usuário afasta os demais documentos, mesmo que este esteja empilhado com outros documentos e pega o que lhe interessa. Essa liberdade de manipulação, existente em uma mesa de trabalho real, o *mouse* infelizmente não fornece.

De acordo com o que foi dito na Introdução, a comunidade de RV, apesar de ter concebido equipamentos e métodos de interação que permitem construir interfaces que estão cada vez mais próximas do modelo real de área de trabalho, ainda enfrenta problemas com relação ao *hardware* utilizado, devido à quantidade de fios, peso dos equipamentos, limitação de movimentos, etc. Com o objetivo de diminuir ou até eliminar a necessidade de qualquer *hardware* acoplado à mão do usuário, muitas pesquisas, apoiando-se na Visão Computacional, estão sendo realizadas de forma a permitir a utilização da mão humana como dispositivo de computador através do rastreamento da mão por câmeras de vídeo. O arranjo básico de tais sistemas é ilustrado na Figura 2.1, onde, através de uma ou mais câmeras que captam os movimentos da mão, o computador executa uma determinada ação.

Na seção 2.1 serão apresentados alguns projetos relacionados a este trabalho que se beneficiam com o uso da mão humana como dispositivo.

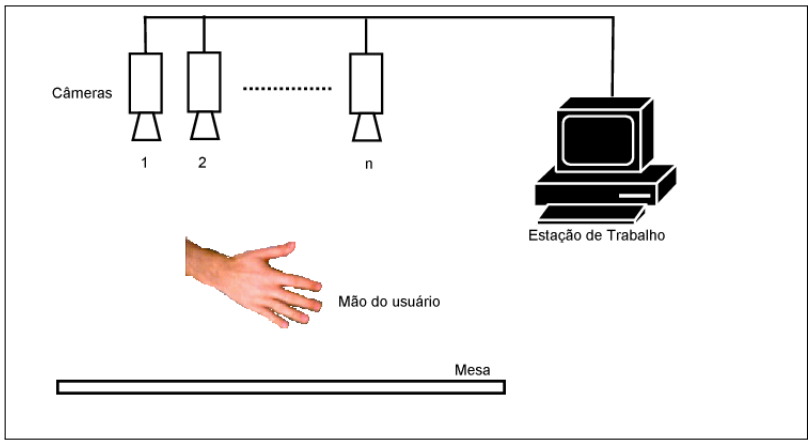


Figura 2.1: Arranjo básico de um sistema que utiliza a mão como dispositivo de computador.

2.1 Projetos Relacionados

Vários projetos científicos ou comerciais foram desenvolvidos com o objetivo de tornar as interfaces de computadores mais intuitivas e fáceis de aprender. No que diz respeito à utilização da mão, a maioria dos projetos subdivide-se em reconhecimento de gestos ou o uso da mão como *mouse* virtual. Nas seções 2.1.1 e 2.1.2, serão descritas algumas dessas aplicações.

2.1.1 Reconhecimento de Gestos e Poses

O uso de reconhecimentos de gestos se constitui num dos mais atrativos campos de pesquisa para o desenvolvimento de interfaces inteligentes [36] devido à grande quantidade de movimentos possíveis, e ao modo intuitivo de como se faz uso desses gestos [26]. Algumas aplicações em reconhecimentos de gestos incluem reconhecimento de linguagens de sinais [36, 41, 46] e reconhecimento de pose, que neste caso limitam-se a reconhecer estados da mão, como: mão fechada, aberta, dedo apontando para cima, para baixo, etc. Em áreas como a robótica o reconhecimento de gestos é utilizado para controlar robôs móveis à distância [15]. Através de uma interface de reconhecimento de gestos alguns eletrodomésticos também podem ser controlados remotamente, conforme afirma Freeman [20].

Em alguns ambientes de RV, gestos e poses são utilizados para implementar a navegação dentro de ambientes virtuais, como mostrado em Sato e Saito [52], onde é apresentado um sistema reconhecimento de poses e gestos, que permite o usuário navegar dentro de um ambiente virtual gerado através de um grande *display* imersivo, mostrado na Figura 2.2.

Nas apresentações multimídia, comuns em conferências e reuniões, geralmente o palestrante é obrigado a se deslocar para o computador e utilizar o *mouse* ou o teclado para poder navegar pelos *slides*. Para facilitar a navegação, sem a necessidade de utilizar o *mouse* ou o teclado, Berárd [6], construiu um sistema que por meio da identificação de gestos, pode-se executar os comandos *"next-slide"* e *"previous-slide"*. Outro sistema, com este mesmo objetivo, pode ser encontrado no trabalho de Licsár [34]. Wu [61] apresenta um quadro negro virtual 3D, destinado a ser usado em salas de aula e que permite construir objetos através de gestos.

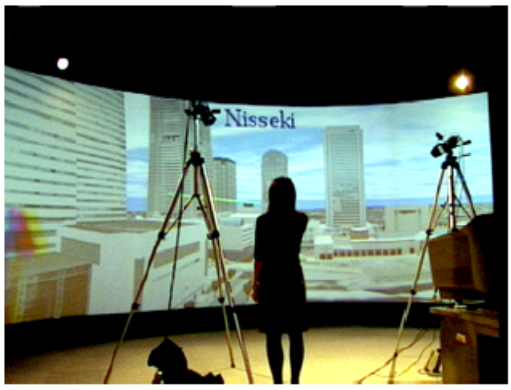


Figura 2.2: Sistema de navegação 3D em ambiente de RV [52].

Geralmente os sistemas de apresentações multimídia baseiam-se na arquitetura *câmera-projetor*, mostrada na Figura 2.3. Nesta arquitetura a câmera é utilizada para captar a mão e seus movimentos e por meio de reconhecimento de poses ou gestos executar uma determinada função do programa que está sendo exibido por meio do projetor.

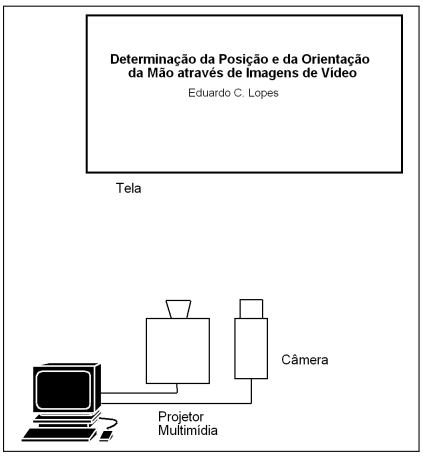


Figura 2.3: Arquitetura câmera-projetor.

2.1.2 Uso da mão como *mouse*

Algumas pesquisas realizam esforços para substituir o *mouse* por algo bem mais fácil de usar. Uma opção muito comum é utilizar um dedo como o cursor do *mouse*. Bérard [6] apresenta um sistema denominado *FingerMouse* que permite controlar o cursor do *mouse* através da mão nua. O usuário simplesmente movimenta a mão em frente a uma câmera para posicionar o cursor do *mouse* na tela e os *clicks* do *mouse* são gerados quando o usuário deixa um dedo esticado por mais de 1 segundo. Este sistema utiliza um projetor para exibir imagens do monitor em uma parede, e o *FingerMouse* permite controlar aplicativos *Windows* como o *Internet Explorer* e o *Paint Brush*, conforme mostra a Figura 2.4, onde o usuário controla uma navegador de *Internet* (Figura 2.4-a) ou faz desenhos utilizando um dedo no lugar do *mouse* (Figura 2.4-b).

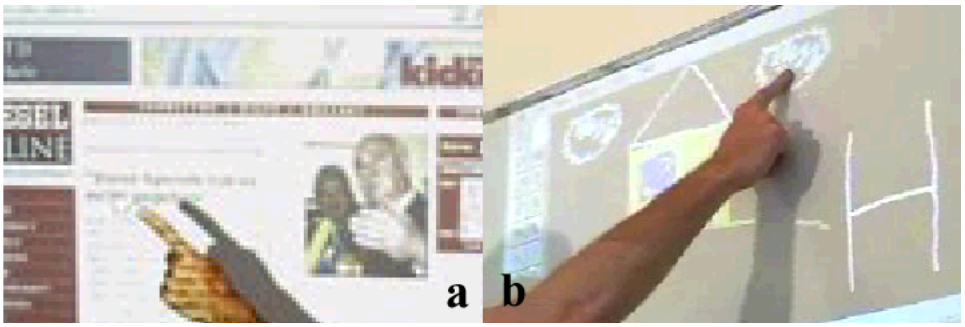


Figura 2.4: Exemplo do uso da mão como *mouse*.

Zelinsky [62] aproveita os resultados obtidos no rastreamento de dedos para implementar um *mouse* onde o usuário pode usar os dedos para manipular objetos. No trabalho de Rehg [47] é apresentado um *mouse* gráfico para um ambiente virtual 3D utilizando uma câmera para rastrear a mão. Outro exemplo da implementação de um *mouse* 3D, usando a mão, pode ser encontrado no trabalho de Burkhardt e Neumann [10]. Uma outra aplicação semelhante pode ser vista em [63] que, no lugar de usar um *mouse* virtual, implementa um menu virtual onde cada dedo representa um item do menu. As opções são acionadas quando o dedo correspondente é flexionado. Na Figura 2.5 temos um exemplo do *Finger Menu*.

Figura 2.5: Exemplo de um *Finger menu* [63].

Outra aplicação interessante do rastreamento de dedos no lugar do *mouse*, destinada a sessões de *brain storm*, comuns em universidades, é apresentada por Bérard [6]. Em seu sistema cada participante da sessão utiliza um teclado sem fio para colocar novas idéias em um *display* fixado numa parede e utiliza os dedos para selecionar ou arrastar cada item exibido. Ao final da sessão, o resultado pode ser salvo e distribuído pela *Internet*.

Além do *mouse*, dispositivos como *joysticks*, utilizados principalmente em jogos estão também sendo substituídos pela mão [19]. Outro exemplo, na área de jogos, é o trabalho de Parker [44], que substitui o *mouse* pela mão permitindo que o usuário utilize os mesmos movimentos que faria se estivesse jogando o jogo *Solitaire* com cartas reais.

Capítulo 3

Segmentação de Pele

3.1 Introdução

O primeiro passo para determinar a posição da mão do usuário, através de imagens, é segmentar a mão do resto do fundo. Em muitas atividades como detecção e rastreamento de faces, detecção de mãos em imagens digitais e reconhecimento de gestos [22, 59], detecção de imagens pornográficas [30], são utilizados algoritmos de detecção de pele.

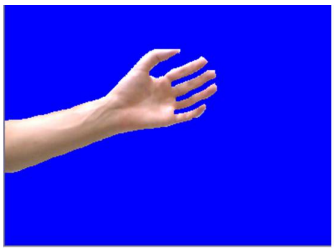
A possibilidade de detectar a pele em sistemas de rastreamento de faces e mãos é justificada principalmente pela característica invariante da mesma, ou seja, independente do ponto de vista e das deformações ocorridas devido a rotações e perspectiva, a pele não muda suas características como cor e textura.

A cor da pele basicamente é determinada por uma substância denominada melanina. Embora exista variação na concentração de tal substância nas diferentes raças humanas, o que permite existir peles com cores diferentes, a cromaticidade da mesma é uma característica que permite identificá-la com precisão [7]. Com relação à pele humana foi constatado que, independente de suas variações (branca, negra, amarela, etc) elas tendem a formar um *cluster* (aglomerado) no espaço de cores [14], denominado *cluster de pele*, como aquele mostrado na Figura 3.1(b), obtido a partir de uma imagem contendo somente pele como aquela mostrada na Figura 3.1(a).

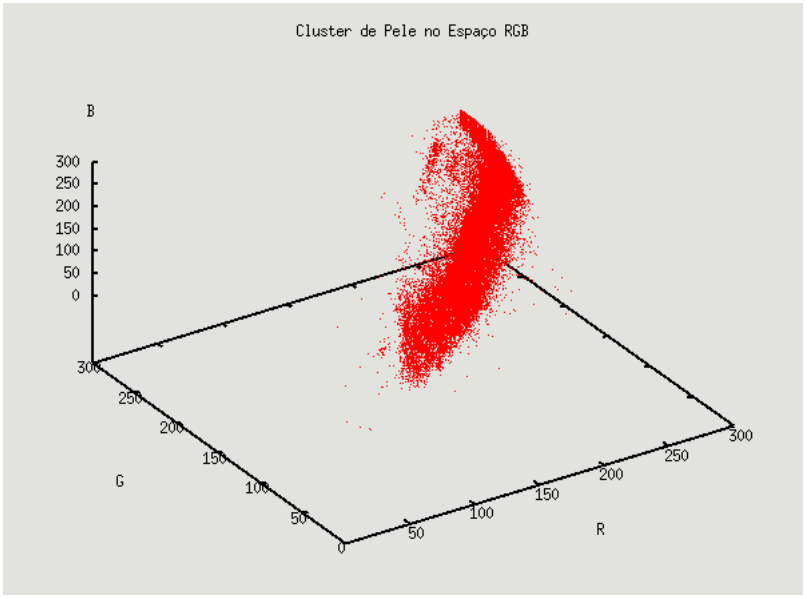
Em um sistema de detecção de pele, três problemas devem ser solucionados, nesta ordem:

- Escolher o espaço de cores;
- Modelar o que é a pele no espaço de cores escolhido;
- Definir algoritmo de classificação do que é pele;

A escolha do espaço de cores define a forma como uma cor é representada numericamente. A modelagem da pele consiste em construir um conjunto de regras ou uma estrutura de dados que define, dentre todas as cores possíveis de um espaço de cores, quais cores provavelmente são de pele e quais não são. Os algoritmos de classificação, por sua vez, de posse das informações contidas nos modelos de pele decidem se um determinado *pixel* em uma imagem de entrada é um *pixel* de pele ou um *pixel* que pertence ao fundo da imagem.



(a) Imagem contendo somente pele



(b) *cluster* de pele

Figura 3.1: Obtenção do *cluster* de pele.

Nas próximas seções deste trabalho serão apresentados os espaços de cores utilizados, bem como os modelos de pele e os algoritmos de segmentação de pele implementados.

3.2 Espaço de Cores

Os espaços de cores mais conhecidos e utilizados são o RGB, RGB normalizado e o HSV. Albiol *et al* [3] demonstra que o espaço de cores utilizado não tem grande influência sobre algoritmos de segmentação de pele, mas para um determinado algoritmo de segmentação, o seu desempenho pode ser maximizado quando se utiliza um espaço de cores específico. A fim de determinar qual o melhor espaço de cor, bem como determinar qual o melhor algoritmo de segmentação para este trabalho, alguns testes foram realizados com 4 algoritmos de segmentação utilizando os espaços de cores RGB, RGB-normalizado e o HSV, respectivamente.

O modelo RGB está presente na maioria dos dispositivos de captura de vídeo sendo o espaço de cores mais utilizado em imagens digitais. Nele a cor é representada pela combinação de três valores que correspondem ao nível de vermelho, verde e azul, respectivamente. Devido ao

fato de que o RGB mistura os dados de cor e luminosidade, é pouco recomendado utilizar este espaço de cor em algoritmos de detecção de pele [59]. Em contrapartida, o espaço de cores RGB normalizado consiste em retirar uma das componentes de cor do RGB e normalizar as demais conforme as Equações 3.1 e 3.2.

$$r = \frac{R}{R+G+B} \tag{3.1}$$

$$g = \frac{G}{R+G+B} \tag{3.2}$$

Nas Equações 3.1 e 3.2 a componente B (*blue*/azul) é retirada, por não possuir nenhuma informação relevante, além de diminuir a influência da iluminação [21] e permitir a redução da dimensionalidade dos dados de 3 para 2 dimensões [59].

Além do RGB, o HSV (*Hue, Saturation* e *Value*) é um espaço de cores muito utilizado em sistemas de detecção de pele. A componente H descreve a matiz da cor, S a saturação ou a quantidade de preto na cor e finalmente, V denota o brilho ou a quantidade de branco (luz) presente na cor. O principal motivo de se utilizar este espaço de cores é a possibilidade de amenizar os efeitos da iluminação simplesmente retirando a componente V, pois diferente do RGB este espaço de cores não mistura as cores com a luz.

Neste trabalho os três sistemas de cores foram testados e avaliados e, uma vez que se tenha definido o espaço de cores a ser utilizado o próximo passo é definir um mais modelos de pele, que serão vistos na 3.3.

3.3 Modelos de Pele

Como as cores da pele humana tendem a formar um *cluster* no espaço de cor a classificação um *pixel* se resume em determinar se a cor do mesmo pertence ou não ao *cluster* de pele.

Para que seja possível implementar algoritmos de classificação, deve-se, em um primeiro momento, modelar (definir, representar) o que é e o que não é pele. Neste trabalho foram utilizados dois modelos de pele: o Modelo Baseado em Regras e o Modelo Probabilístico, explicados em detalhes a seguir.

3.3.1 Modelo Baseado em Regras

A maneira mais simples de modelar os *pixels* de pele consiste em analisar as cores que a pele costuma assumir, em um espaço de cores específico, e representá-las explicitamente através de um conjunto de regras pré-definido que determina os intervalos ou conjunto de valores de cor que a pele pode assumir. Este conjunto de regras servirá para decidir se um determinado *pixel* deve ser classificado como *pixel* pele ou fundo [59].

Os classificadores baseados em regras geralmente são rápidos, embora apresentem como principal dificuldade a necessidade de utilizar um espaço de cores específico e um conjunto de

regras eficiente.

3.3.2 Modelo Probabilístico

O Modelo probabilístico, adotado neste trabalho, é representado por uma distribuição de probabilidades que permite calcular a probabilidade de um determinado *pixel* ser de pele. Nesta abordagem, surge o conceito de **dados de treinamento**, pois para construir o modelo de pele necessita-se previamente de imagens contendo apenas pele. A partir das imagens de treinamento, como aquela mostrada na Figura 3.2, constrói-se a distribuição de probabilidade da seguinte maneira:

1. Conta-se a ocorrência de cada *pixel* de pele e constrói-se uma tabela denominada histograma de cores ou **LUT** (*Lookup Table*). Para o espaço de cores RGB-normalizado, cada entrada da tabela é uma tripla (r,g,f), onde r,g é a cor do *pixel* e f é a freqüência de ocorrência deste *pixel* na imagem. De maneira semelhante para o espaço de cores HSV, as entradas da tabela consistem da tripla (H,S,f);

2. Após a construção da tabela, dividi-se todas as freqüências pela maior freqüência e normaliza-se estes valores entre 0 e 255.

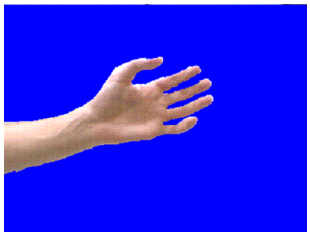


Figura 3.2: Imagem de treinamento utilizada para construir o modelo de pele.

O segundo modelo probabilístico utilizado neste trabalho baseia-se na construção de uma distribuição de intervalos de probabilidades, denominada **bin-histograma** [2]. A construção deste modelo consiste em particionar o espaço de cores em intervalos iguais, chamados de **bins**, conforme apresentado por Ahmad [2]. A geração do *bin*-histograma consiste em contar o número de ocorrências de pontos em um determinado *bin*. Ao ler um *pixel* da imagem de treinamento, suas coordenadas de cor são utilizadas para determinar a qual intervalo de *pixels* (*bin*) ele pertence e posteriormente a freqüência do seu respectivo *bin* é atualizada. Após determinar a freqüência de ocorrência de cada *bin*, divide-se as freqüências pela maior freqüência e normaliza-se os resultados dessas divisões entre 0 e 255.

A geração das tabelas LUT e do *bin*-histograma é denominada **fase de treinamento**. Uma vez que se tenha definido o modelo de pele, o próximo passo é construir um algoritmo que classifique os *pixels* de uma imagem qualquer, de acordo com as informações contidas no modelo adotado. A seguir serão mostrados os algoritmos de segmentação de pele avaliados neste trabalho.

3.4 Algoritmos de classificação

Os modelos de pele contém somente informações sobre o que provavelmente é pele ou não. A estratégia de classificação consiste de um algoritmo que analisa os *pixels* de uma imagem de entrada e de acordo com os dados contidos nos modelos de pele determina se o mesmo é de pele ou fundo. Os quatro algoritmos implementados foram projetados e testados para classificar *pixels* no espaço de cores RGB-normalizado e HSV. Alguns destes algoritmos utilizam também um modelo de fundo, sendo que este é construído da mesma maneira que o modelo de pele. A única diferença é que as imagens, agora, são imagens de fundo que não possuem nenhuma ocorrência de *pixels* de pele.

3.4.1 Algoritmo Baseado em Regras

Como visto anteriormente, a estratégia mais simples de segmentação de pele são os algoritmos baseados em regras. Um pseudocódigo usando este conceito é mostrado no Algoritmo 1. As regras consistem de comparações simples dos valores de cada componente de cor do espaço de cores RGB. Este algoritmo foi utilizado por Tomaz [57], para remover *pixels* de fundo, baseando-se na premissa de que a área ocupada pelos *pixels* de pele é bem menor do que a área ocupada pelos *pixels* de fundo.

Um teste com este algoritmo foi realizado utilizando um conjunto de imagens contendo somente pele e outro conjunto de imagens contendo somente imagens de fundos. Se algoritmo não apresenta nenhuma falha de detecção de pele, então para uma imagem contendo somente pele a imagem de saída será igual a imagem de entrada, ou seja, todos os *pixels* de pele serão detectados. O contrário ocorrá caso seja aplicado ao algoritmo uma imagem contendo somente fundo, pois nenhum *pixel* de pele será detectado. Entretanto, como qualquer algoritmo de segmentação de pele, este também apresenta falhas de detecção de pele e, pode-se calcular a taxa de detecção de pele dividindo-se o número de *pixels* de pele da imagem de saída pelo número de *pixels* de pele da imagem de entrada. Essa taxa quanto mais alta melhor é o algoritmo. De maneira semelhante, pode-se calcular a taxa de detecção de fundo, dividindo-se o número de *pixels* de fundo da imagem de saída pelo número *pixels* de fundo da imagem de entrada. Mas, ao contrário de taxa de detecção de pele, esta, quanto mais baixa melhor é o algoritmo, pois significa que o algoritmo está descartando os *pixels* de fundo presente na imagem de entrada.

Sendo assim, o Algoritmo 1 foi testado com 50 imagens de pele e fundo e, para cada uma delas foi calculada a taxa de detecção de pele e fundo respectivamente. Obteve-se 71% de detecção de pele e 32% de detecção de fundo, com tempo médio de processamento de 0.0043 segundos.

Na Figura 3.3 temos um exemplo de uma imagem processada pelo Algoritmo 1, que apresenta um desempenho razoável em termos de detecção de pele, mas deixa muito a desejar devido a alta taxa de falsos positivos, ou seja, *pixels* de fundo erroneamente classificados como pele.

Algoritmo 1 Algoritmo Regras(R,G,B)

```
se ((B > 160 and R < 180 and G < 180) or
(G > 160 and R < 180 and B < 180) or
(B < 100 and R < 100 and G < 100) or
(G > 200) or
(R+G > 400) or
(G > 150 and B < 90) or
(B/(R+G+B)> 0.40)or
(G/(R+G+B)> 0.40)or
(R < 102 and G > 100 and B > 110 and G < 140
and B < 160) )  então
  é fundo
senão
  é pele;
fim se
```

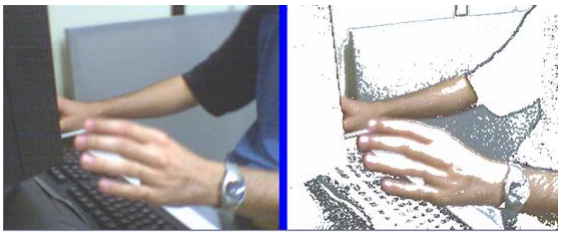


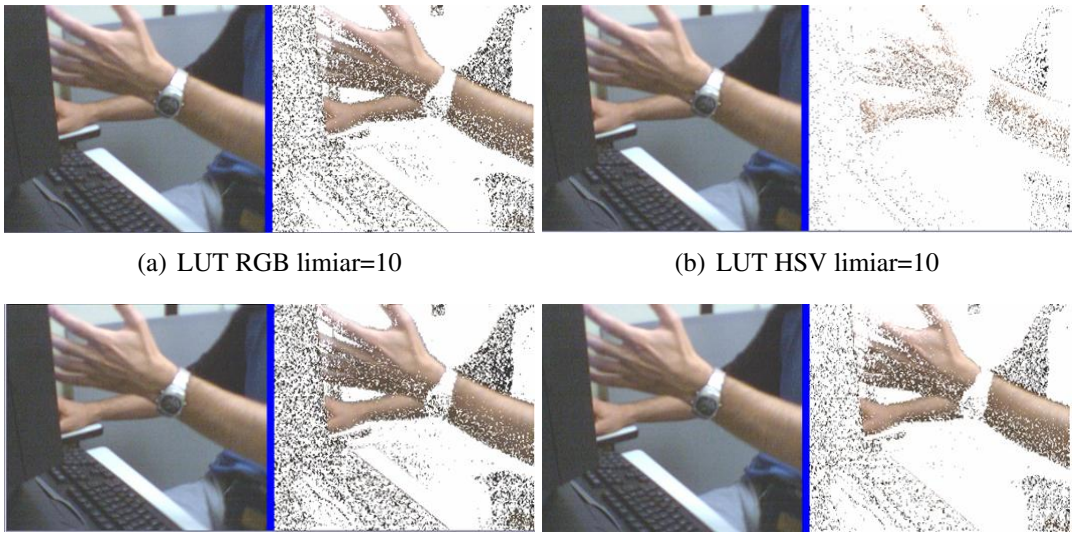
Figura 3.3: Esquerda: imagem de entrada, direita: imagem segmentada.

3.4.2 Algoritmo Limiar Simples

O algoritmo denominado *Limiar Simples* é um dos mais rápidos. Nele, um *pixel* de pele é classificado como sendo de pele se a sua freqüência no histograma de pele, ou no bin-histograma, for maior que um limiar (*threshold*). No Algoritmo 2 temos a listagem do algoritmo Limiar Simples e na Figura 3.4, alguns resultados obtidos com esta técnica.

Algoritmo 2 Algoritmo Limiar Simples(pixel, modelo de pele)

```
se frequencia do pixel > limiar   então
  é pele
senão
  é fundo
fim se
```



(a) LUT RGB limiar=10 (b) LUT HSV limiar=10

(c) Bin de tamanho 4 RGB limiar=10 (d) Bin de tamanho 4 HSV limiar=10

Figura 3.4: Imagens segmentadas pelo algoritmo Limiar Simples com limiar=10.

3.4.3 Algoritmo da Maior Freqüência

Semelhante ao algoritmo anterior, este utiliza a tabela de pele e a de fundo. O critério de seleção é simples: se a freqüência de um determinado *pixel* é maior no histograma de pele do que no histograma de fundo, então o *pixel* tem grande probabilidade de ser pele, portanto, classificado como tal, e vice-versa. No Algoritmo 3, temos a listagem do algoritmo da Maior Freqüência e, na Figura 3.5 exemplos de imagens processadas por ele.

Algoritmo 3 Algoritmo Maior Freqüência(pixel, modelo de pele, modelo de fundo)

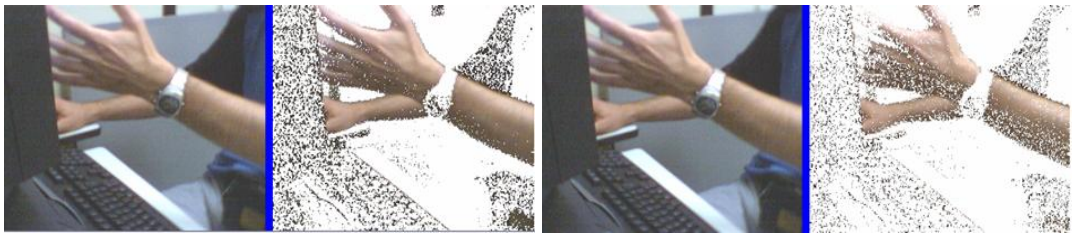
```
se pele.frequencia > fundo.frequencia  então
  é pele;
senão
  é fundo.
fim se
```

3.4.4 Algoritmo Baseado no Teorema de Bayes

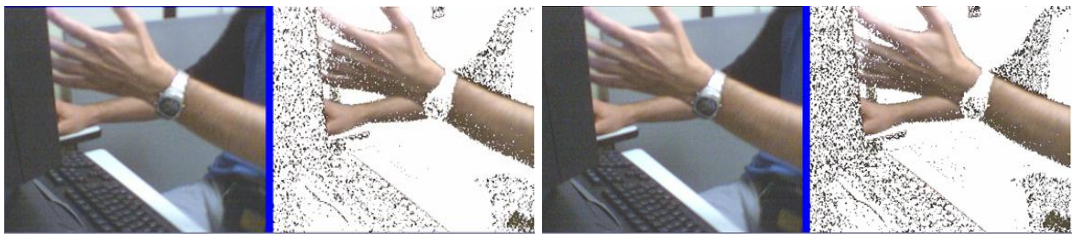
O quarto algoritmo construído também utiliza os modelos de pele e fundo, sendo que a probabilidade da cor do *pixel* ser de pele ou fundo é calculada pela Equação 3.3.

$$p(pele|c) = \frac{p(c|pele)p(pele)}{p(c|pele)p(pele) + p(c|\neg pele)p(\neg pele)} \tag{3.3}$$

Na Equação 3.3 $p(c|pele)$ representa a probabilidade de ocorrer a cor c, sabendo-se que a mesma é de pele. Os elementos $p(pele)$ e $p(\neg pele)$, representam a probabilidade da cor ser de pele e de fundo respectivamente, determinadas diretamente dos modelos de pele e fundo (LUT ou bin-histograma). Essas probabilidades são calculas pelas Equações 3.4 e 3.5.



(a) LUT RGB (b) LUT HSV



(c) BIN-4 RGB (d) BIN-4 HSV

Figura 3.5: Exemplo de imagens processadas pelo algoritmo de maior freqüência.

$$p(pele) = \frac{nPele}{T} \tag{3.4}$$

$$p(\neg pele) = \frac{nFundo}{T}, \tag{3.5}$$

onde $nPele$ representa o número total de *pixels* de pele e $nFundo$ o número total de *pixels* de fundo presentes no conjunto de treinamento, e T o número total de *pixels* presentes no conjunto de treinamento, independente de serem pele ou fundo. O valor de $p(c|\neg pele)$ é retirado do modelo de fundo e denota a probabilidade da cor não ser pele. Na Tabela 3.1, para facilitar o entendimento deste algoritmo cada elemento da Equação 3.3 é listado, seguido de uma breve descrição e a origem de seus valores. Na Figura 3.6, são mostrados alguns resultados do uso desta estratégia.

3.5 Testes e Resultados

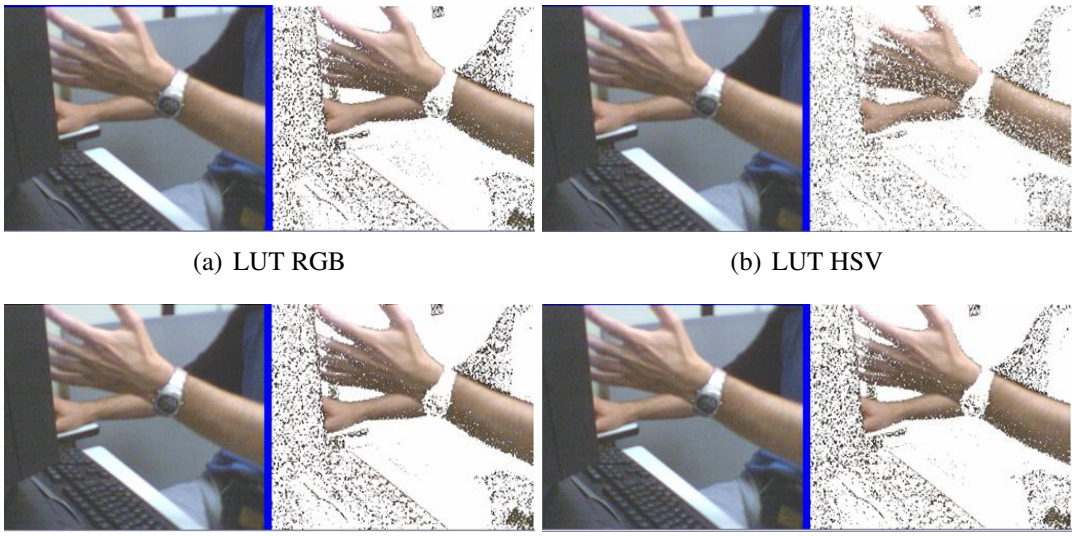
Os testes foram realizados com 50 imagens do conjunto de treinamento, utilizando os espaços de cores RGB-normalizado e HSV e os modelos de cores LUT e bin-histograma. O critério para calcular as taxas de detecção de pele e fundo é o mesmo que foi utilizado para testar o Algoritmo baseado em Regras (Algoritmo 1), descrito na seção 3.4.1.

As Figuras 3.7 a 3.9 apresentam gráficos com o desempenho dos algoritmos de detecção. A métrica de desempenho é a taxa de detecção correta dos *pixels* de pele e fundo. Nos gráficos, quanto mais próximas de 1 são as taxas de detecção de pele e fundo, mais eficiente é o algoritmo.

Como um dos objetivos deste trabalho é detectar a mão em uma imagem através da segmentação da pele, o algoritmo ideal seria aquele que detectasse apenas a pele humana, separando-a completamente do fundo da imagem.

Tabela 3.1: Significado e origem de cada termo da Equação 3.3.

Elemento	**Significado**	**Origem**
$p(pele\|c)$	probabilidade da cor c ser de pele	resultado da Equação 3.3
$p(c\|pele)$	probabilidade de ocorrer a cor c sob condição da mesma ser pele	LUT ou bin-histograma
$p(pele)$	probabilidade de uma cor ser pele	resultado da Equação 3.4
$p(c\|\neg pele)$	probabilidade de ocorrer a cor c sob a condição da mesma ser fundo	resultado da Equação 3.5
$p(\neg pele)$	probabilidade de uma cor ser fundo	LUT ou bin-histograma



(a) LUT RGB (b) LUT HSV

(c) BIN-4 RGB (d) BIN-4 HSV

Figura 3.6: Exemplos de resultados obtidos utilizando o Teorema de Bayes, com limiar=0.5.

Mas, como sempre existe um erro de classificação, a avaliação dos algoritmos testados leva em consideração a taxa de detecção de pele e fundo. Portanto, quanto mais altos e próximos os valores numéricos das duas taxas melhor é o algoritmo. Para o caso da taxa de detecção de fundo, quanto maior o valor da mesma menor o número de *pixels* de fundo classificados como pele, o que facilita na localização da mão, pois tem-se pequenas áreas de *pixels* de fundo em contraste com uma grande área formada por *pixels* de pele.

Nas próximas seções, além da análise dos algoritmos em termos de eficiência na detecção de pele e fundo, também uma análise de tempo de processamento de cada algoritmo é apresentada.

3.5.1 Algoritmo Limiar Simples

O algoritmo Limiar Simples tem seu desempenho mostrado na Figura 3.7, onde é possível notar que este, quando utiliza o limiar=5, atinge uma taxa alta de detecção de pele tanto para o modelo de pele LUT, como para bin-histograma, bem como no espaço de cores RGB e HSV.

Apesar disso, a taxa de detecção correta do fundo da imagem é muito baixa, fazendo com que a maior parte do fundo da imagem, seja classificada erroneamente como pele. Entretanto, na Figura 3.7(b) utilizando um limiar=50, pode-se notar que o modelo de pele LUT, no espaço de cores RGB, apresenta uma queda na taxa de detecção de pele, enquanto que a taxa de detecção de *pixels* de fundo aumenta sensivelmente. Para espaço de cores HSV a taxa de detecção de pele é extremamente baixa, não atingindo 3% de *pixels* de pele detectados corretamente. Para o modelo de pele *bin*-histograma, de tamanho 4, 16 ou 32, tanto no espaço de cores RGB como HSV, a taxa de detecção de pele caiu ou se manteve constante enquanto que a taxa de detecção de fundo aumentou ou se manteve, como ocorre quando se utiliza o modelo *bin*-histograma com um *bin* de tamanho 32, utilizando o espaço de cores RGB, Figura3.7(b). Na Figura 3.7(c), o algoritmo com limiar=100 apresenta piores resultados do que aqueles mostrados na Figura 3.7(b).

Observando os resultados dos testes, pode-se afirmar que à medida que se aumenta o limiar tem-se uma queda na taxa de detecção de pele e um aumento na taxa de detecção de fundo, ou seja, existe uma quantidade grande de *pixels* de pele sendo classificados erroneamente como *pixels* de fundo.

3.5.2 Algoritmo da Maior Freqüência

A Figura 3.8 mostra o desempenho do algoritmo Maior Freqüência, que utiliza os modelos de pele e fundo, classificando um determinado *pixel* como sendo pele ou fundo de acordo com a freqüência com que o mesmo ocorre nos dois modelos, ou seja, o *pixel* somente é considerado um *pixel* de pele se sua freqüência é maior no modelo de pele, e vice-versa. Observando a Figura 3.8 pode-se notar que a taxa de detecção de pele e fundo melhorou, com relação ao algoritmo Limiar Simples.

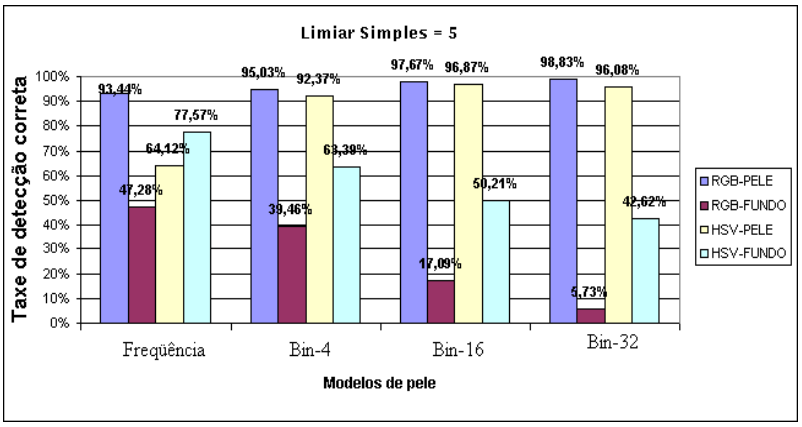
Além disso, é visível que este algoritmo é mais eficiente que o anterior, pois as taxas de detecção de pele e fundo são mais equilibradas do que no algoritmo Limiar Simples, pois os valores numéricos das duas taxas são próximos.

Este algoritmo apresenta melhor eficiência quando utiliza o espaço de cores HSV, apresentando as melhores taxas de detecção de pele e fundo, principalmente quando se utiliza o modelo de pele e fundo *bin*-histograma com *bin* de tamanho 16.

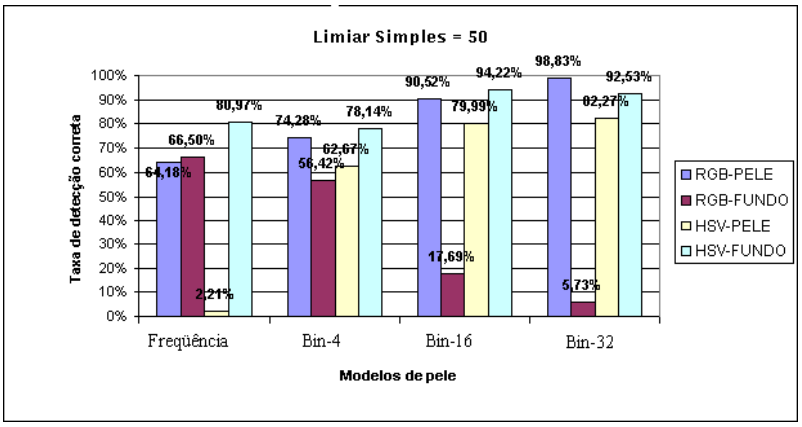
3.5.3 Teorema de Bayes

O algoritmo que utiliza o Teorema de Bayes também apresentou um desempenho semelhante ao algoritmo Maior Freqüência. Na Figura 3.9(a) pode-se notar que o algoritmo, com limiar=0.3[1], também apresenta melhores resultados quando utiliza o espaço de cores HSV. Para

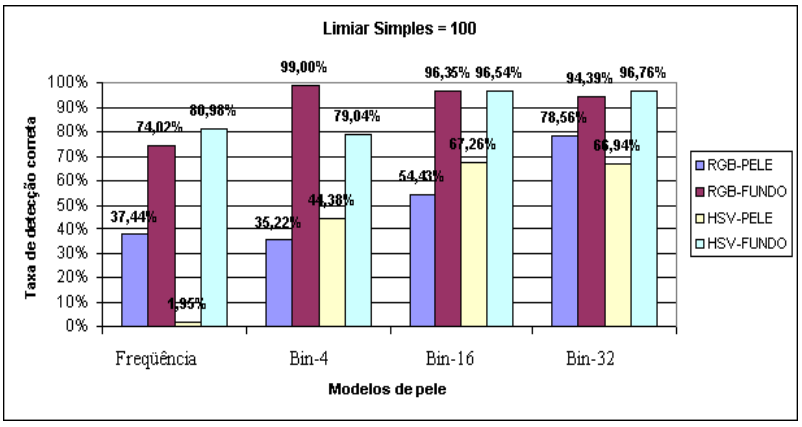
[1]Significa dizer que o algoritmo classificará como pele os *pixels* que apresentarem probabilidade igual ou superior a 30%.



(a) Taxa de detecção correta de pele e fundo com limiar=5



(b) Taxa de detecção correta de pele e fundo com limiar=50



(c) Taxa de detecção correta de pele e fundo com limiar=100

Figura 3.7: Gráficos de desempenho do algoritmo Limiar Simples para diferentes valores de limiar.

todos os casos, exceto para o modelo de cores LUT, o espaço de cores RGB apresentou resultados inferiores ao HSV.

Na Figura 3.9(b), utilizando um limiar=0.5, pode-se observar que os resultados utilizando o modelo de cores LUT no espaço RGB se tornam mais eficientes. Apesar do modelo LUT

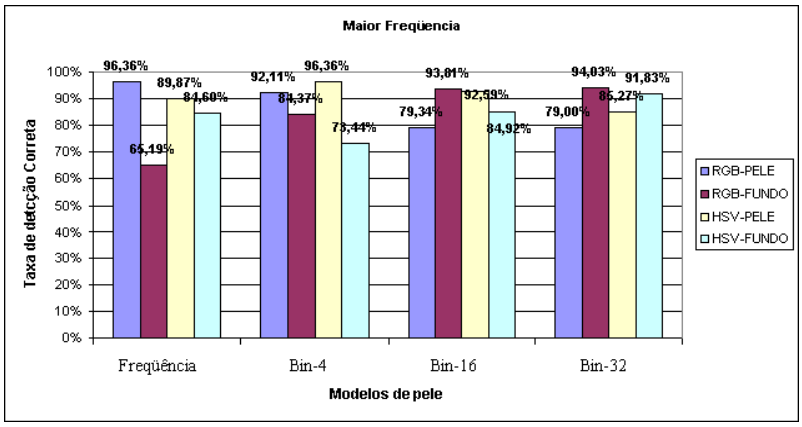


Figura 3.8: Gráfico do desempenho do algoritmo Maior Freqüência.

com RGB ter a taxa de segmentação de pele menor quando utiliza um limiar = 0.5, a taxa de detecção de fundo aumenta, e seu valor numérico se aproxima mais da taxa de segmentação de pele, fazendo com que a classificação de *pixels* entre pele e fundo seja mais equilibrada.

Novamente para o modelo de cores *bin*-histograma, o espaço de cores HSV mostra-se superior ao RGB, apresentando taxas de detecção de pele e fundo melhores, exceto quando se utiliza um *bin* de tamanho 32, conforme a Figura 3.9(a).

3.6 Tempo de Execução

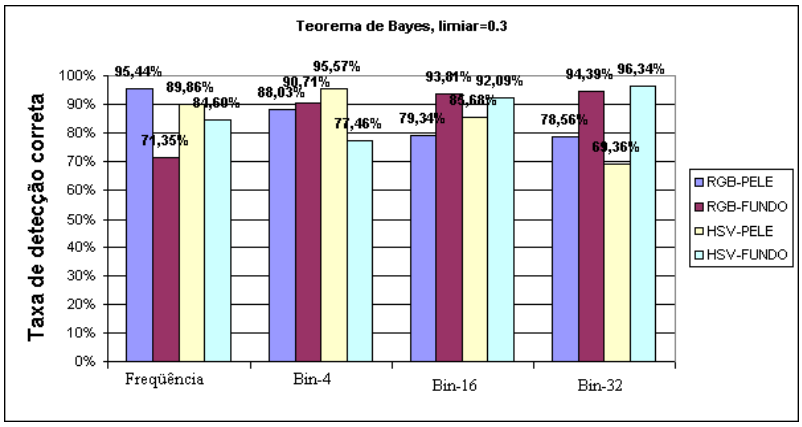
De acordo com análise feita na seção anterior, o espaço de cores HSV demonstrou bons resultados principalmente no algoritmo da Maior Freqüência e o algoritmo baseado no Teorema de Bayes.

Entretanto, antes de classificar o *pixel* com pele ou fundo, existe a necessidade de converter a cor do *pixel* de RGB para HSV. Como o presente trabalho exige o funcionamento destes algoritmos em tempo real, o tempo de conversão de um espaço de cores para outro deve ser considerado, bem como o tempo de processamento de cada algoritmo.

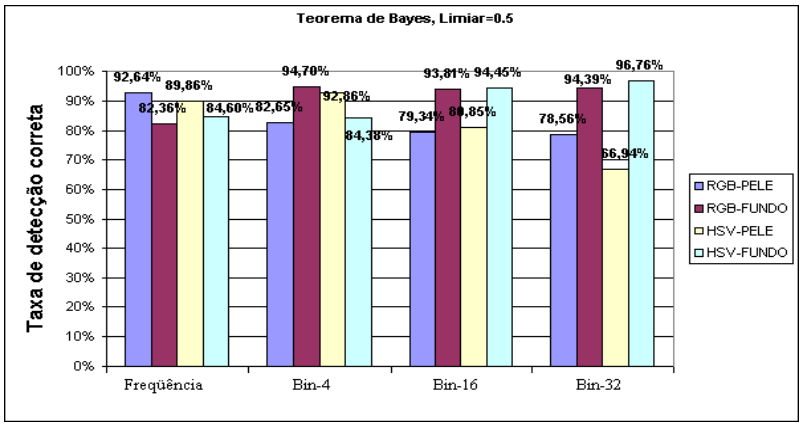
Para mensurar os tempos consumidos pelos algoritmos, aplicou-se 50 imagens em cada um deles, e o tempo gasto em cada imagem foi armazenado. Posteriormente a média de tempo de cada algoritmo foi calculada. Na Figura 3.10, tem-se um gráfico com a média de tempo de cada algoritmo, medida em segundos. Neste gráfico pode-se notar que trabalhar com RGB é mais rápido e que o pior algoritmo em termos de tempo é o algoritmo que utiliza o Teorema de Bayes no espaço de cores HSV, embora leve menos de 0.1 seg para processar uma imagem.

3.7 Testes Utilizando Imagens com Fundo Controlado

O conjunto de treinamento utilizado nos exemplos anteriores consistia de imagens contendo somente pele (Figura 3.2) e imagens que não possuíam nenhuma ocorrência de pele, conforme mostrado na Figura 3.11(a).



(a) Teorema de Bayes com limiar=0.3



(b) Teorema de Bayes com limiar=0.5

Figura 3.9: Gráficos do desempenho do algoritmo baseado no Teorema de Bayes.

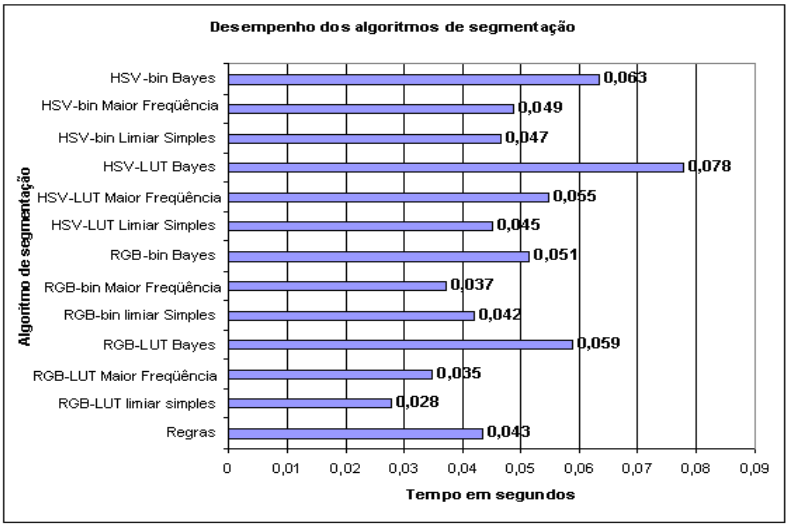


Figura 3.10: Desempenho em termos de tempo de cada algoritmo avaliado.

Como a utilização dos algoritmos de detecção de pele neste projeto não será feita em ambientes com fundo complexo como mostrado na Figura 3.11(a), um novo conjunto de treinamento

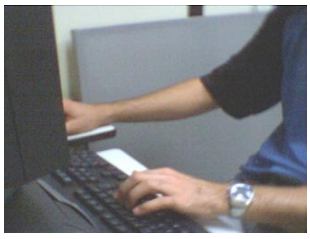
foi construído utilizando imagens como a imagem mostrada na Figura 3.11(b), onde o fundo não tem uma grande variação de cores.

Após a construção dos modelos de cores, com imagens de fundo controlado, foram testados os algoritmos Maior Freqüência e Teorema de Bayes, por terem apresentado os melhores resultados. Na Figura 3.12, é possível notar a melhora no desempenho dos algoritmos Maior Freqüência e Teorema de Bayes.

Além de aumentar as taxas de detecção de pele e fundo, os algoritmos apresentaram resultados mais eficientes, uma vez que essas taxas de detecção de pele e fundo apresentam valores numéricos muito próximos.

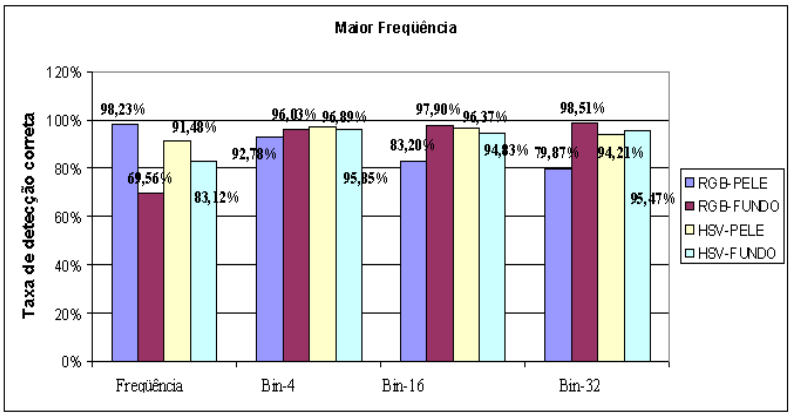
Pode-se notar que para o espaço de cores HSV, utilizando o modelo de cor *bin*-histograma com *bin* de tamanho 4, tem-se a melhor eficiência dos dois algoritmos. Entretanto o algoritmo Maior Freqüência é considerado superior, pois apresenta o mesmo desempenho que o algoritmo Teorema de Bayes, mas com tempo de processamento menor, conforme visto na Figura 3.10. Portanto neste projeto o algoritmo de segmentação de pele adotado é o Algoritmo da Maior Freqüência, utilizando o espaço de cores HSV e o modelo de cores bin-histograma com bin de tamanho 4 ou 16.

(a) Fundo complexo

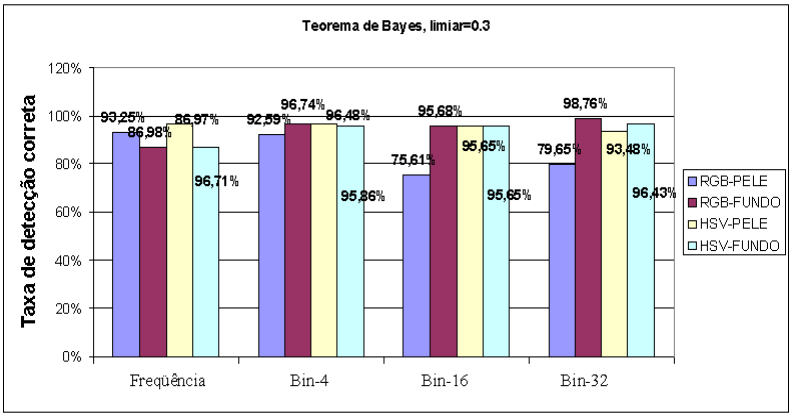


(b) Fundo controlado

Figura 3.11: Diferentes imagens utilizadas para construir os modelos de fundo complexo e controlado.



(a) Algoritmo: Maior Freqüência



(b) Algoritmo baseado no Teorema de Bayes com limiar=0.3

Figura 3.12: Desempenho com fundo controlado.

Capítulo 4

Determinando a Posição da Mão

4.1 Introdução

Neste capítulo são descritas as técnicas utilizadas para construir uma estratégia que forneça a localização da mão do usuário, dada uma imagem de entrada. Também serão mostrados alguns algoritmos utilizados para corrigir falhas resultantes da segmentação da pele, bem como serão mostrados os dois algoritmos construídos para determinar a posição da mão, seguidos de seus respectivos resultados.

4.2 Segmentação da Mão

Devido aos problemas da segmentação, não se obtém uma separação total entre os *pixels* de pele e fundo, como pode ser visto na Figura 4.1. Portanto é necessário refinar o resultado obtido da segmentação de pele obtendo o máximo de informação possível sobre a mão do usuário, para que se possa determinar sua posição com precisão.

Figura 4.1: Exemplo de segmentação de pele imperfeita.

Observando a Figura 4.1 pode-se notar as seguintes imperfeições da segmentação:

- *pixels* de fundo classificados erroneamente como pele;
- *pixels* de pele classificados como fundo, gerando buracos na superfície da mão.

Para contornar este problema e melhorar o resultado da segmentação da mão, foram construídos algoritmos auxiliares que corrigem tais falhas. Nas próximas seções serão discutidos cada algoritmos e sua função. Através de exemplos, serão mostrados os resultados obtidos com a aplicação dos mesmos.

4.2.1 Retirando *pixels* de fundo

A estratégia utilizada neste trabalho para localizar a mão e separá-la do fundo baseia-se no fato de que a imagem resultante da segmentação de pele é composta de pequenas áreas de fundo classificadas como pele em contraste com uma grande área de pele, como mostrado na Figura 4.1.

Seguindo este raciocínio, a maior área contígua de pele é selecionada como sendo a mão. Tal área é obtida através do algoritmo recursivo *floodfill*, no qual dado um ponto da imagem, ele retorna o número de *pixels* conectados a ele, ou seja, o número de *pixels* que compõe a área contínua a qual ele pertence. No Algoritmo 4 temos a listagem do pseudocódigo do algoritmo *floodfill* e, onde a variável global *contador* armazena o número de *pixels* de pele diretamente conectados ao *pixel* (x, y) passado como argumento para a função. A variável *contador* é sempre iniciada de zero para cada *pixel* analisado. Na Figura 4.2, tem-se um exemplo da aplicação do algoritmo *floodfill* sobre uma imagem de entrada.

Algoritmo 4 Algoritmo *floodfill*

```
se  x ou y maior que imagem  então
  retorna null
fim se
se  x ou y menor que zero  então
  retorna null
fim se
contador++
se cor(x,y) == pele  então
  se cor(x+1,y) == pele  então
    floodfill(x+1,y)
  fim se
  se cor(x-1,y) == pele  então
    floodFill(x-1,y)
  fim se
  se cor(x,y+1) == pele  então
    floodFill(x,y+1)
  fim se
  se cor(x,y-1) == pele  então
    floodFill(x,y-1)
  fim se
senão
  parar
fim se
```

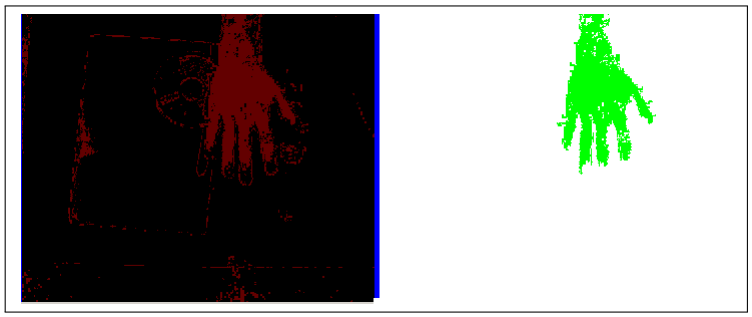


Figura 4.2: Exemplo de funcionamento do algoritmo *floodfill*.

4.2.2 Eliminando Buracos na Superfície da Mão

Conforme visto anteriormente, a segmentação produz alguns buracos na superfície da mão, como mostrado na Figura 4.1. Para corrigir este problema, usou-se novamente o algoritmo *floodfill*, para preencher a região externa à mão, pintando o fundo de uma determinada cor. Na Figura 4.3, é ilustrado um resultado da aplicação deste algoritmo, que recebe como entrada uma imagem com a região da mão delimitada por um retângulo (Figura 4.3(a)). Em seguida, o algoritmo preenche o fundo com uma determinada cor, enquanto os *pixels* que não pertencem ao fundo são pintados de outra cor (Figura 4.3(b)) e finalmente, retira os *pixels* de fundo, deixando apenas a mão sem falhas em sua superfície (Figura 4.3(c)). Como é possível notar, na Figura 4.3 todo este procedimento é realizado dentro de uma região, sem a necessidade de processar a imagem inteira.

(a) Imagem de entrada. (b) Preenchimento do fundo. (c) Seleção de *pixels* de pele.

Figura 4.3: Preenchimento de falhas na superfície da mão.

4.3 Obtendo a Posição da Mão

Após corrigir as falhas presentes na superfície da mão, já é possível utilizar algum método para calcular a posição da mesma. Considera-se como posição da mão a posição de um ponto que se encontra no centro da palma da mão. Descobrir este ponto pode não ser uma tarefa trivial, uma vez que a mão é um objeto articulado e muda sua forma de acordo como sua orientação e a configuração dos dedos. Para identificar a posição da mão duas estratégias foram estudadas e avaliadas, conforme será visto a seguir.

4.3.1 Centro de Massa

A primeira estratégia utilizada para calcular a posição da mão foi determinar o centro de massa, obtido pelas Equações 4.1 e 4.2, aplicadas sobre a superfície da mão. Nas esquações n corresponde ao número de *pixels* da área da mão.

$$x_c = \frac{\sum_{i=1}^{n} x_i}{n} \tag{4.1}$$

$$y_c = \frac{\sum_{i=1}^{n} y_i}{n} \tag{4.2}$$

Apesar de ser um algoritmo simples de implementar e rápido o suficiente para ser utilizado em tempo-real, este não apresentou o resultado esperadoem determinados casos. Dependendo da pose da mão e da presença do antebraço na cena, o resultado pode ser um ponto fora da região da palma da mão. Na Figura 4.4 temos alguns exemplos de resultados positivos obtidos.

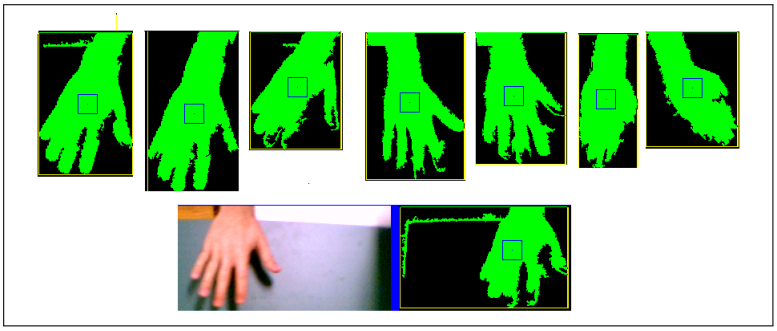


Figura 4.4: Posição da mão calculada através do centro de massa.

Entretanto na Figura 4.5 tem-se situações onde o centro de massa não corresponde ao centro da palma da mão e, por meio de testes foi possível notar que este fenômeno ocorre quando se tem uma presença marcante do antebraço na cena, fazendo com que o ponto calculado se desloque em sua direção. Os melhores resultados obtidos com esta técnica ocorre quando se tem a palma da mão paralela ao plano da câmera com os dedos esticados e com o antebraço aparecendo discretamente na cena, como visto na Figura 4.4.

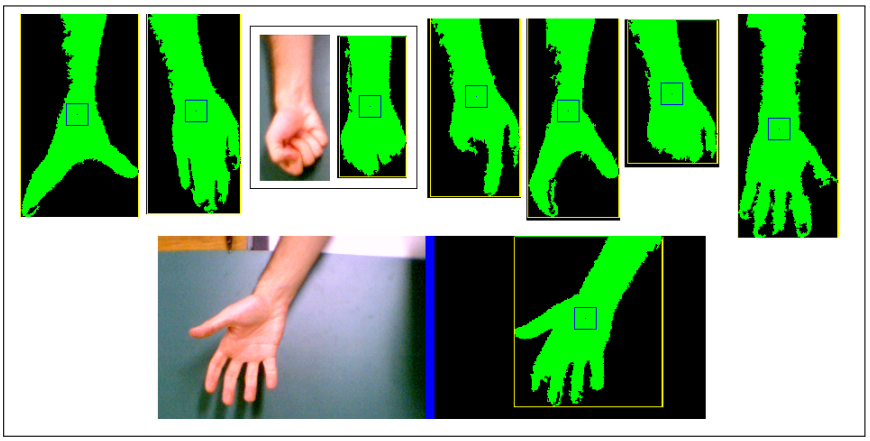
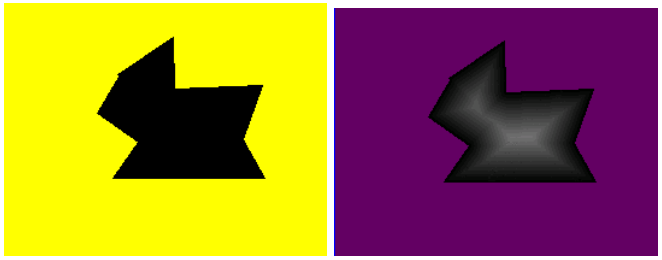


Figura 4.5: Exemplos de resultados negativos.

4.3.2 Transformada da Distância

A outra técnica avaliada baseia-se na Transformada da Distância, utilizada por Morris [39] e Deimel [16] para localizar o centro da palma da mão. A transformada da distância é normalmente aplicada sobre uma imagem binária, cujo resultado é outra imagem, denominada *imagem de distâncias*. O valor da intensidade dos *pixels* desta imagem é o valor da distância ao limite mais próximo do mesmo [39], sendo que o limite considerado pode ser o fundo da imagem ou o contorno de um objeto [37]. Um exemplo de transformada da distância aplicada à uma imagem binária (Figura 4.6(a)) pode ser visto na Figura 4.6, onde é possível perceber que quanto mais interno são os *pixels* dentro do objeto, maior é o valor de suas intensidades, ou seja, maior o valor da distância em relação ao fundo da imagem, como pode ser percebido ao se observar a Figura 4.6(b).



(a) Imagem binária de entrada. (b) Imagem de distâncias.

Figura 4.6: Exemplo de aplicação da transformada da distância.

Em uma primeira tentativa de determinar o centro da mão, foi implementado um método que toma o contorno da mão e calcula a transformada da distância dos pontos internos à mão com relação a este, sendo a Distância Euclidiana utilizada como cálculo de distância entre dois pontos, da seguinte maneira:

1. Para cada ponto P que pertence a região da mão calcula-se a sua distância em relação a cada ponto Q do contorno;

2. Atribui-se $d(P,Q)$ a menor distância calculada, ou seja a distância entre P e o ponto Q do contorno que está mais próximo dele.

3. Seleciona-se entre as distâncias $d(P,Q)$ calculadas no passo 2, a maior delas, ou seja, a maior distância entre as menores distâncias. O ponto correspondente a distância selecionada é considerado o centro da mão.

Alguns resultados podem ser vistos na Figura 4.7, mostrando que este método funciona bem com diferentes configurações da mão, sendo robusto com relação às falhas da detecção do contorno. Na Figura 4.7 o centro do círculo corresponde ao centro da mão.

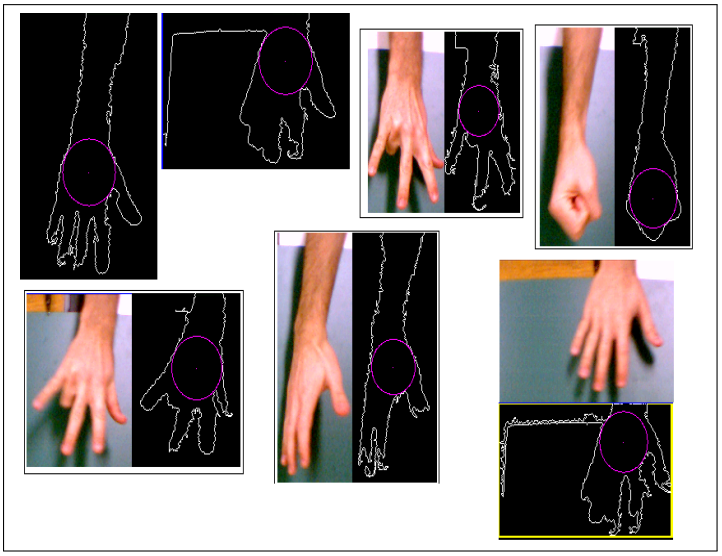


Figura 4.7: Centro da mão obtido pelo cálculo da transformada da distância.

Embora este método tenha apresentado resultados melhores que o centro de massa, existem casos como aqueles mostrados na Figura 4.8, que o cálculo do centro da mão degrada devido a má qualidade da segmentação da pele, que provoca erros na detecção do contorno. No caso da Figura 4.8 uma parte da superfície da mão não foi detectada, de tal forma que não é possível com este método localizar o centro da mão.

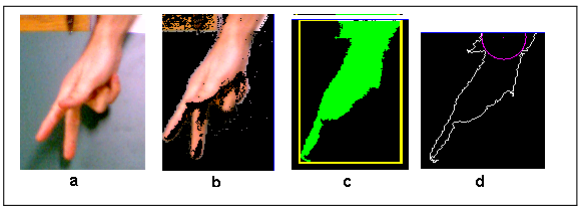


Figura 4.8: Erro no cálculo do centro da mão.

Embora exista a possibilidade de ocorrerem situações como visto na Figura 4.8, estas não se constituem em um problema grave, pois melhorando qualidade da segmentação e detecção do contorno da mão pode-se calcular o centro da mão através da transformada de distância com uma ótima precisão.

Entretanto o maior problema deste método reside no custo computacional do mesmo, devido a procura global da menor distância em um dado momento (passo 1 e 2) e da maior distância entre as menores (passo 3), ou seja para cada *pixel P* pertencente à mão calcula-se a distância $d(P,Q)$ em relação cada *pixel Q* pertencente ao contorno. Além disso, o cálculo de distância é obtido pela Distância Euclidiana, que utiliza operações complexas, como potências. Portanto, devido a estas desvantagens este método não serve para aplicações de tempo-real.

Felizmente, existe uma alternativa de calcular a transformada de distância sem utilizar a Distância Euclidiana e sem precisar utilizar o contorno da mão. De acordo com Butt [12] é possível utilizar uma aproximação da Distância Euclidiana de forma que o cálculo de distância entre pontos tenha baixo custo computacional e o mínimo de erro, tornando possível utilizar a transformada de distância em aplicações de tempo-real [16].

Segundo Gunilla [24], existem várias aproximações para a Distância Euclidiana, sendo as mais conhecidas as distâncias *City Block, Chamfer-3-4, Chamfer-5-7-11 e Chessboard*. Para o cálculo do centro da mão utilizou-se a distância *Chamfer-3-4* ou **DCT** (*Distance Chamfer Transform*) que, de acordo com Morris [39] e Deimel [16], apresenta os melhores resultados em termos de desempenho e precisão, devido ao uso de informações locais ao *pixel* analisado, como as distâncias dos *pixels* da vizinhança que já foram analisados [12].

O algoritmo para calcular a DCT consiste em deslizar uma máscara em uma imagem binária, sendo que o tamanho e os valores dessa máscara variam de acordo com a aproximação utilizada. Para a distância *Chamfer-3-4* a máscara utilizada é mostrada na Figura 4.9.

4	3	4
3	0	3
4	3	4

Figura 4.9: Máscara utilizada para calcular a distância *Chamfer-3-4*.

O algoritmo realiza dois passos sequenciais denominados *forward* e *backward* respectivamente. No passo *forward* a imagem é varrida da esquerda para direita e de cima para baixo, utilizando a máscara mostrada na Figura 4.10(a) que mostra em destaque quais os *pixels* utilizados para analisar o *pixel* central (quadrado pintado na Figura 4.10). Nesta fase a menor distância é calculada com relação aos *pixels* que estão a esquerda e aos *pixels* que estão acima do *pixel* analisado. No passo *backward* a imagem é varrida de baixo para cima, da direita para esquerda, utilizando-se a máscara mostrada na Figura 4.10(b) e, a menor distância é calculada com relação aos *pixels* abaixo e aos *pixels* do lado direito do *pixel* analisado. No Algoritmo 5 tem-se a listagem do pseudocódigo do algoritmo para calcular a DCT. Após obter a imagem de distâncias seleciona-se o *pixel* de maior intensidade como o centro da mão.

Alguns dos resultados dos testes realizados são mostrados na Figura 4.11, onde o centro da mão corresponde ao centro do quadrado presente na imagem. Nesta figura é possível notar que para diversas configurações da mão a DCT consegue detectar o centro da palma da mão com maior precisão do que o centro de massa. Além disso, o uso da distância *Chamfer-3-4* como aproximação da Distância Euclidiana melhorou o resultado do cálculo da transformada da distância, tornando esta mais robusta com relação as falhas de segmentação. Na Figura 4.12 pode-se

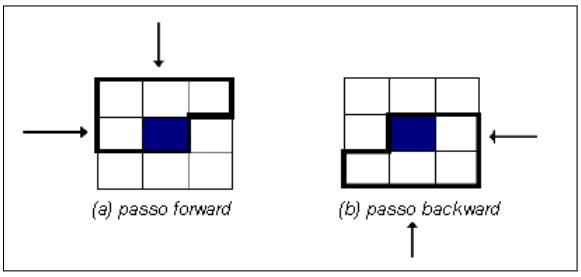


Figura 4.10: Aplicação da matriz de convolução.

Algoritmo 5 Algoritmo DCT(imagem binária imgIn)

```
A=3;
B=4; /* fase forward */
para i=1 até imgIn.nLinhas faça
  para j=1 até imgIn.nColunas faça
    se imgIn.intensidade(j,i) == 255  então
      mask[0] = imgIn.intensidade(j,i); /*pega intensidade do pixels, 0 ou
      255*/
      mask[1] = imgIn.intensidade(j-1,i)  +A;
      mask[2] = imgIn.intensidade(j-1,i-1)+B;
      mask[3] = imgIn.intensidade(j,i-1)  +A;
      mask[4] = imgIn.intensidade(j+1,i-1)+B;
      ptoMenor = minimo(mask); /*pega o valor mínimo*/
      ImgOut.setPixel(j,i,ptoMenor);/*muda a intensidade do pixel analisado*/
    fim se
  fim para
fim para
/*fase backward*/
para i=imgIn.nLinhas-1 até 1 faça
  para j=imgIn.nColunas-1 até 1 faça
    se imgIn.intensidade(j,i) == 255  então
      mask[0] = imgIn.intensidade(j,i);
      mask[1] = imgIn.intensidade(j+1,i)  +A;
      mask[2] = imgIn.intensidade(j,i+1)  +B;
      mask[3] = imgIn.intensidade(j+1,i+1)+A;
      mask[4] = imgIn.intensidade(j-1,i+1)+B;
      ptoMenor = minimo(mask);
      imgOut.setPixel(j,i,ptoMenor);
    fim se
  fim para
fim para
retorna imgOut; /*imagem de distâncias*/
```

observar a melhora na qualidade dos resultados da transformada da distância, onde para este caso específico o método anterior havia apresentado um resultado pior. Mas apesar dessas vantagens existem alguns poucos casos nos quais novamente a qualidade da segmentação prejudica a eficiência desta estratégia, como mostrado na Figura 4.13. Embora tenha ocorrido este erro, este é

justificável para o caso da Figura 4.11, pois a palma da mão não foi segmentada.

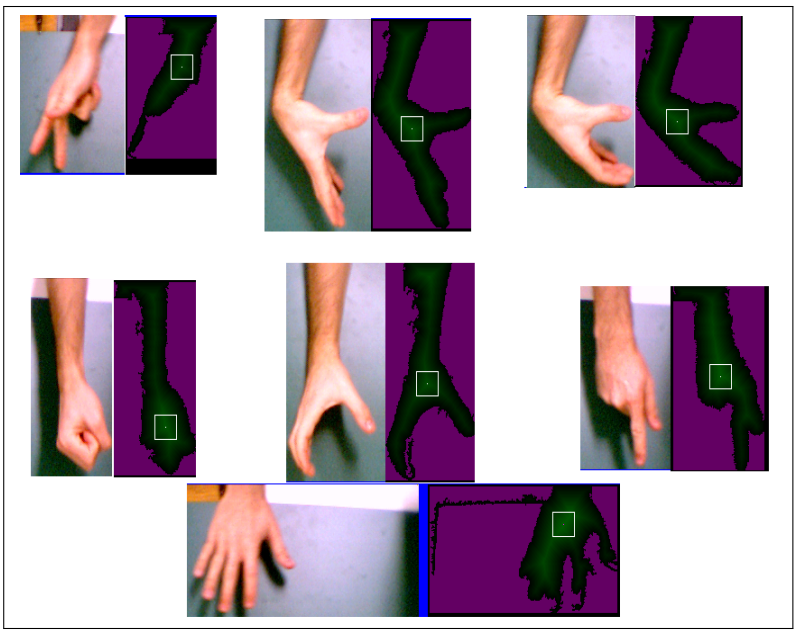


Figura 4.11: Centro da mão calculado através da distância *Chamfer-3-4*.

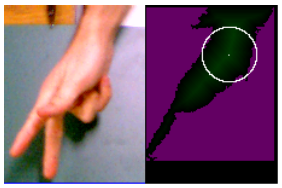


Figura 4.12: Centro da mão calculada sobre uma imagem mal segmentada.

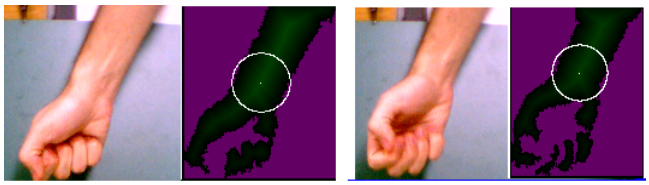


Figura 4.13: Casos em que a distância *Chamfer-3-4* falhou.

Apesar desses problemas, o cálculo da transformada da distância utilizando a DCT não sofre interferência alguma devido à presença do antebraço na cena, como acontece com o centro de massa, sendo que este método apresentou os melhores resultados. Portanto, para este projeto a transformada da distância, utilizando a DCT como aproximação da Distância Euclidiana se constitui na melhor opção para localizar a mão do usuário, podendo ser utilizada em uma aplicação de tempo-real além de ser a técnica mais robusta com relação as diversas poses da mão.

No entanto, vale lembrar que se a pose da mão for restrita a pose mostrada na Figura 4.4, o centro de massa se torna a melhor escolha para calcular a posição da mão devido ao seu desempenho, em termos de tempo, ser superior aos demais algoritmos.

Na Tabela 4.1 tem-se a listagem do tempo médio de cada algoritmo implementado, onde é possível observar que o centro de massa é o método mais rápido, mas devido aos problemas discutidos anteriormente, ele não se mostrou uma boa opção para determinar a posição da mão. E, apesar dos bons resultados, em termos de obtenção do centro da mão, a transformada da distância utilizando a Distância Euclidiana é o método mais lento, levando 0.5 segundos para calcular a posição da mão. Portanto o método que possui a melhor relação entre qualidade e tempo de resposta é a transformada da distância utilizando a DCT. Os tempos mostrados na Tabela 4.1, referem-se somente aos algoritmos de obtenção da posição da mão, sem considerar o tempo gasto em etapas anteriores como, a segmentação de pele, localização da mão e preenchimento de buracos na superfície da mão.

Além disso são mostrados, também, os tempos relativos dos algoritmos em relação ao centro de massa com o objetivo de fornecer uma noção da diferença de desempenho entre os mesmos, independente do *hardware* utilizado. Pode-se notar, através da Tabela 4.1 que a DCT é 20 vezes mais lerda que o centro de massa e que a transformada da distância com a Distância Euclidiana é 538 vezes mais lerda que o centro de massa. O tempo de processamento de algoritmo depende da quantidade de *pixels* analisados e, quanto maior a área de pele detectada, mais tempo cada algoritmo necessita para calcular a posição da mão.

Tabela 4.1: Desempenho dos métodos de obtenção da posição da mão.

Método	**Tempo médio**	**Tempo Relativo ao Centro de Massa**
Centro de massa	0.00102 seg	1
Transf. Dist. c/ Distância Euclidiana	0.54968 seg	538.90
Tansf. Dist. (*Chamfer-3-4*)	0.02075 seg	20.34

Capítulo 5

Determinando a Orientação da Mão

5.1 Momentos de Imagem

Os momentos de imagem permitem calcular determinadas propriedades geométricas de objetos presentes em uma imagem como posição, tamanho e orientação. Tais propriedades são interessantes na implementação de propostas de reconhecimento de gestos, rastreamento de objetos e extração de características para reconhecimento de padrões [13, 26, 35], onde há a necessidade de encontrar descritores de objetos que sejam invariantes com relação a translação, rotação e escala [23, 33].

Segundo Rocha [50] e Gonzales [23], para uma função bidimensional $f(x,y)$ os momentos de ordem $(p+q)$ são definidos pela Equação 5.1.

$$M_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x,y)dxdy \tag{5.1}$$

Como estamos aplicando a Equação 5.1 sobre um objeto binário numa imagem digital, a função $f(x,y)$ será igual a 1 para todos os *pixels* que pertencem ao objeto, portanto podemos simplificá-la para a Equação 5.2:

$$M_{pq} = \sum_{A} x^p y^q, \tag{5.2}$$

onde A é a área do objeto, ou simplesmente o número de *pixels* que o compõe, e corresponde ao momento de ordem zero M_{00}.

A partir da Equação 5.2, pode-se obter informações como o centróide do objeto e sua orientação. O centróide do objeto é obtido pelo cálculo dos momentos de primeira ordem definidos pelas Equações 5.3 e 5.4, que em última análise correspondem as equações do centro de massa apresentadas no Capítulo 4 (seção 4.3.1).

$$x_c = \frac{M_{10}}{M_{00}} \tag{5.3}$$

$$y_c = \frac{M_{01}}{M_{00}} \tag{5.4}$$

Na seção 5.2 será descrito como calcular a orientação da mão em uma imagem utilizando os momentos de imagens e os problemas ocorridos com o uso desta técnica.

5.2 Cálculo da Orientação da mão

Para facilitar a análise de objetos que possuem um formato complexo, pode-se utilizar uma elipse equivalente que melhor descreve sua forma, como ilustrado na Figura 5.1. A elipse equivalente permite determinar algumas das propriedades do objeto em análise, como sua posição, a qual pode ser calculada pelas Equações 5.3 e 5.4, que corresponde ao centróide do objeto e também ao centro da elipse. Além da posição, a orientação do objeto pode ser obtida através da orientação da elipse.

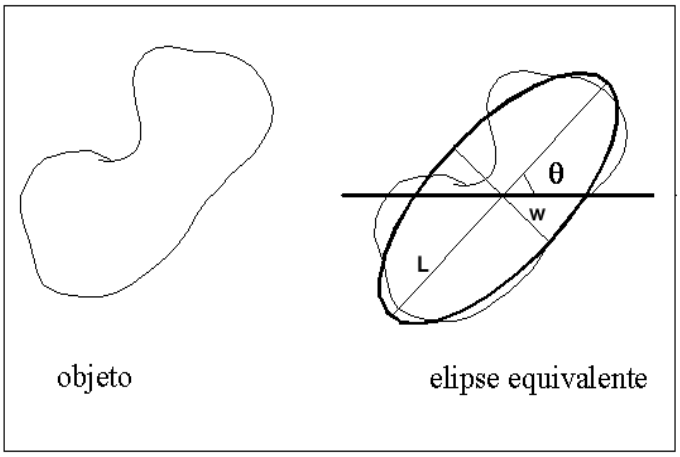


Figura 5.1: Elipse equivalente que descreve a posição e a orientação de um determinado objeto.

Neste trabalho seguiu-se a mesma de idéia de Rocha [50], Freeman [19] e Bradski [9], que consideram a orientação de um objeto a orientação do maior eixo principal da elipse equivalente. A orientação deste eixo pode ser calculado pela Equação 5.5

$$\theta = \frac{arctan\left(\frac{b}{a-c}\right)}{2}, \tag{5.5}$$

onde θ corresponde ao menor ângulo entre o maior eixo e a horizontal e, a, b e c são calculados pelas seguintes equações:

$$a = \frac{M_{20}}{M_{00}} - x_c^2 \tag{5.6}$$

$$b = 2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right) \tag{5.7}$$

$$c = \frac{M_{02}}{M_{00}} - y_c^2 \tag{5.8}$$

Os tamanhos dos eixos principais da elipse podem ser calculados através das seguintes 5.9 e 5.10:

$$W = 2\left(\sqrt{6(a+c-\sqrt{b^2+(a-c)^2})}\right) \tag{5.9}$$

$$L = 2\left(\sqrt{6(a+c+\sqrt{b^2+(a-c)^2})}\right) \tag{5.10}$$

onde W corresponde ao menor eixo e L corresponde ao maior eixo.

Na Figura 5.2 tem-se alguns resultados do cálculo da orientação da mão.

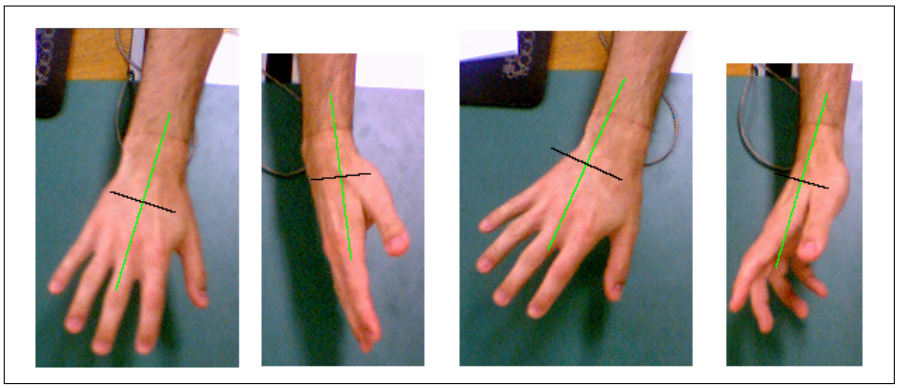


Figura 5.2: Orientação da mão.

Em alguns testes notou-se que a presença do antebraço na cena prejudica o cálculo da orientação da palma da mão, pois a orientação do eixo principal da elipse é influenciada pela orientação do antebraço, uma vez que a área dele também é utilizada no cálculo dos momentos de imagem. Na Figura 5.3 pode-se notar a influência do antebraço no cálculo da orientação, através da observação da orientação do maior eixo da elipse, que não corresponde a orientação da palma da mão.

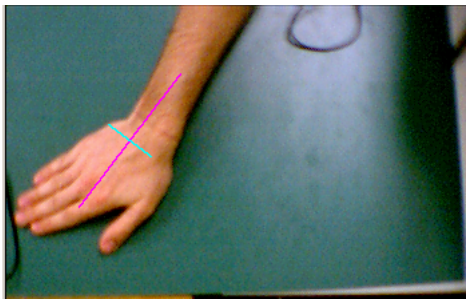


Figura 5.3: Antebraço prejudica o cálculo da orientação da mão.

Devido a este fenômeno, adotou-se a condição de que somente a mão aparece na cena, enquanto o antebraço permanece oculto pelo uso de roupas de manga comprida ou qualquer outro objeto que o impeça de ser segmentado junto com a mão. O uso de uma fita no pulso também é possível desde que área da mão seja a maior área de pele presente na imagem.

Embora haja a necessidade de restringir o antebraço, outros testes revelaram que em algumas poses da mão como as mostradas na Figura 5.4 o cálculo da orientação, utilizando somente a mão, freqüentemente retorna resultados errados. Para corrigir este problema adotou-se uma solução simples, ou seja, aproveitando-se do fato que o antebraço influencia na orientação, considera-se que somente pequena parte do antebraço (por volta de 1 polegada) deve aparecer na imagem de forma que seja possível obter um resultado mais correto para tais poses, como pode ser visto na Figura 5.5. Em seguida na Figura 5.6 tem-se vários exemplos da orientação da mão com algumas poses possíveis.

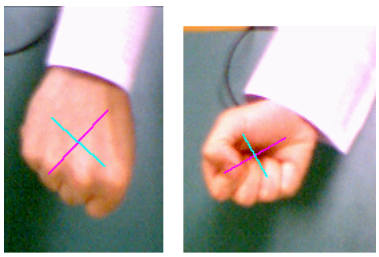


Figura 5.4: Problemas gerados devido a restrição imposta ao antebraço.

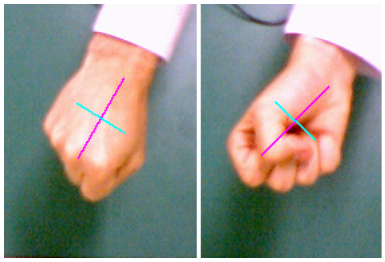


Figura 5.5: Correção dos problemas devido a restrição no antebraço.

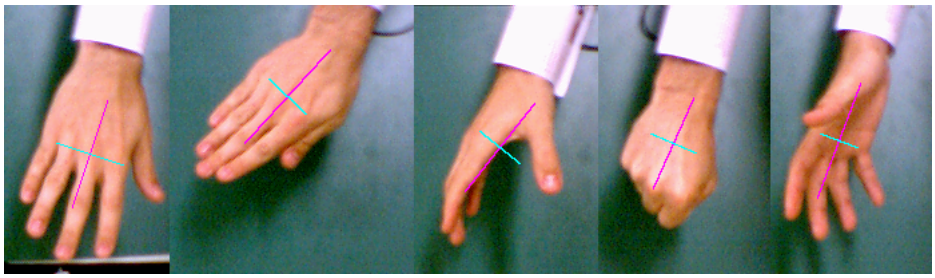


Figura 5.6: Exemplo de orientação da mão.

De acordo com o que foi mencionado em capítulos anteriores, uma das necessidades deste projeto é que o mesmo deve ser executado em tempo real. Como a orientação da mão é calculada aplicando-se a Equação 5.5 sobre a área da mão, um teste foi realizado aplicando-se esta mesma equação somente sobre o contorno da mão obtido a partir de uma modificação do algoritmo *floodfill*, cujo pseudocódigo é listado no Algoritmo 6. Nesta nova versão, o algoritmo *floodfill* quando utilizado para preencher a área da mão a fim de eliminar buracos (ver seção 4.2.2 do

Capítulo 4), ele também detecta o contorno quando atinge um *pixel* de fundo e armazena o último *pixel* de pele processado por ele em uma lista, obtendo dessa maneira o contorno da mão.

Algoritmo 6 Algoritmo *floodfill* versão 2

```
se cor(x,y) == pele  então
  se cor(x+1,y) == pele  então
    floodfill(x+1,y)
  senão
    borda.adiciona(x+1,y)
  fim se
  se cor(x-1,y) == pele  então
    floodFill(x-1,y)
  senão
    borda.adiciona(x-1,y)
  fim se
  se cor(x,y+1)== pele  então
    floodFill(x,y+1)
  senão
    adiciona.adiciona(x-1,y)
  fim se
  se cor(x,y-1) == pele  então
    floodFill(x,y-1)
  senão
    borda.adiciona(x-1,y)
  fim se
fim se
```

Alguns resultados do cálculo da orientação utilizando o contorno da mão podem ser vistos na Figura 5.7. Como é possível observar nesta figura a diferença entre o resultado obtido com a área e o resultado obtido com o contorno é pequena, sendo que a utilização do último diminui consideravelmente o número de pontos utilizados para o cálculo da orientação da mão, acelerando o cálculo da orientação da mão.

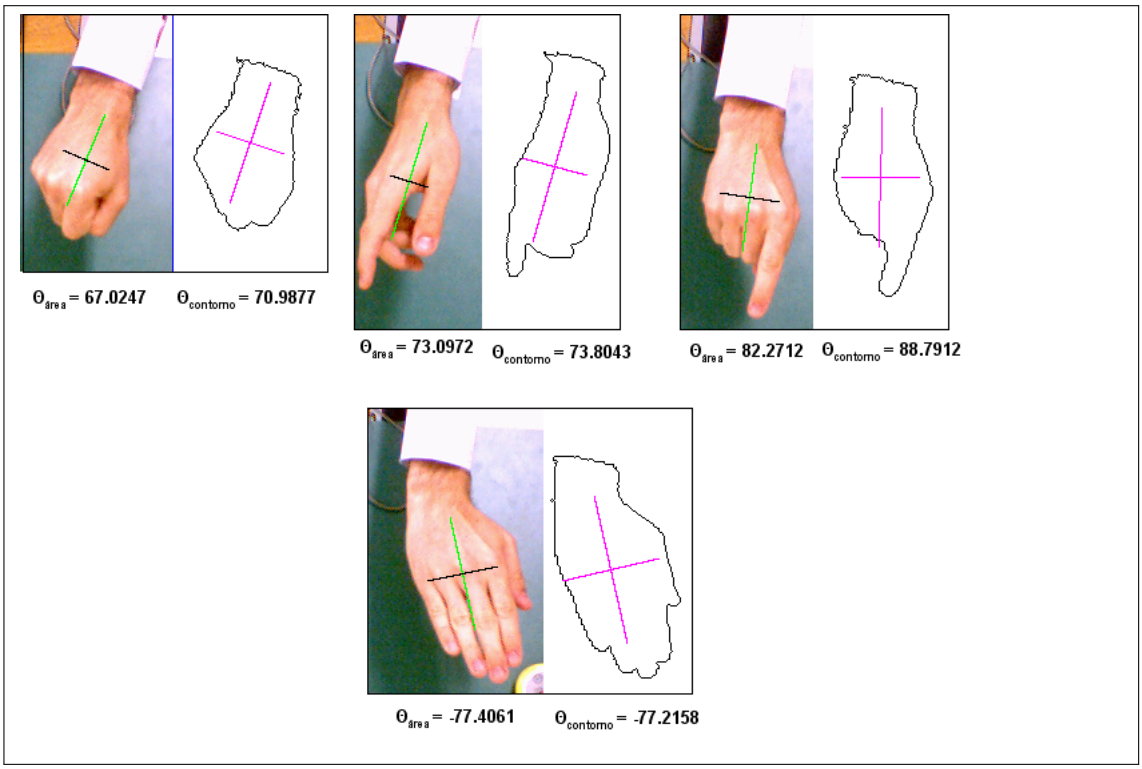


Figura 5.7: Orientação da mão obtida a partir do contorno.

Capítulo 6

Detecção de Características

6.1 Introdução

As técnicas implementadas para determinar a posição e a orientação da mão, mostradas nos Capítulos 4 e 5, retornam valores em duas dimensões. Para efetivamente comparar ou até mesmo substituir um rastreador magnético no processo de rastreamento, é necessário que se obtenha a posição e a orientação da mão em 3D. Isto pode ser realizado através de técnicas de Visão Estereoscópica que utilizam duas câmeras [11, 54]. O arranjo básico de um sistema de visão estereoscópica para duas câmeras sem nenhum alinhamento especial entre elas é mostrado na Figura 6.1. A posição do ponto $P_W = (X_w, Y_w, Z_w)$ no espaço pode ser obtida através do cálculo da intersecção das linhas O_1P_W e O_2P_W, pressupondo as seguintes condições [54]:

- Para cada câmera utilizada deve-se saber sua pose (posição e orientação) no espaço e suas características internas, como a distância focal e fator de distorção radial. Essas informações são representadas em uma matriz denominada *matriz da câmera* que determina como um ponto do espaço 3D é representado no plano da imagem da câmera em 2D, ou seja, essa matriz define o mapeamento (projeção) de um ponto 3D para um ponto 2D. A obtenção destes parâmetros é feita utilizando métodos de calibração de câmera [27,56] que, por meio de imagens conhecidas permitem calcular a matriz da câmera de cada câmera.

- Estabelecer a correspondência entre os pares de pontos P_1 e P_2 na Figura 6.1, ou seja garantir que estes pontos correspondem a um mesmo ponto do espaço P_w, para que se possa recuperar a informação 3D.

Por outro lado, no Capítulo 2 viu-se vários tipos de aplicações que utilizam a mão como dispositivo de entrada. Essas aplicações compreendem sistemas de reconhecimento de gestos e poses e a utilização da mão como *mouse*. Para que seja possível realizar o reconhecimento de gestos e poses, algumas características da mão devem ser detectadas como, por exemplo, as pontas dos dedos [26, 41, 47, 61], e no caso da utilização da mão como mouse virtual ou menu virtual, o mais comum é rastrear as pontas dos dedos e utilizá-los como ponteiros [6, 62] ou como itens de um determinado menu [63]. Portanto, para propostas de obtenção de posição e orientação da mão em 3D, bem como para implementação de interfaces 2D que operem com a

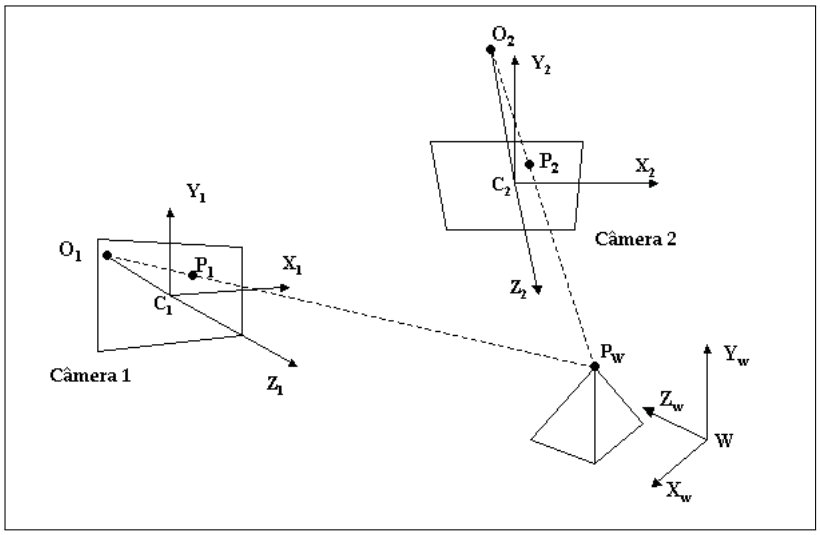


Figura 6.1: Geometria básica de um sistema de visão estéreo.

mão, é necessário obter determinadas características da mão, como o centro da mão, pontas dos dedos, vales entre os dedos, etc.

Um método de extração de características, muito comum, utilizado no *DigitEyes* [47] e no sistema de reconhecimento de gestos MIME [26] é a detecção das pontas dos dedos, das juntas entre os dedos e do pulso (Figura 6.2). Para localizar os dois primeiros elementos, pode-se analisar o contorno da mão em busca de porções do contorno onde há mudança brusca na curvatura (mínimos e máximos locais) do contorno. No caso do pulso, a estratégia mais comum é a busca por regiões de largura constante. Nas seções a seguir, primeiramente será apresentado um algoritmo para gerar a seqüência de pontos que define o contorno da mão. Este passo é necessário, pois originalmente o contorno é apenas um conjunto de pontos sobre uma imagem, sem nenhuma relação de ordem entre os pontos. Em seguida, serão apresentados alguns algoritmos para detectar os mínimos e os máximos locais do contorno e classificar estes elementos em ponta de dedo ou vale entre os dedos.

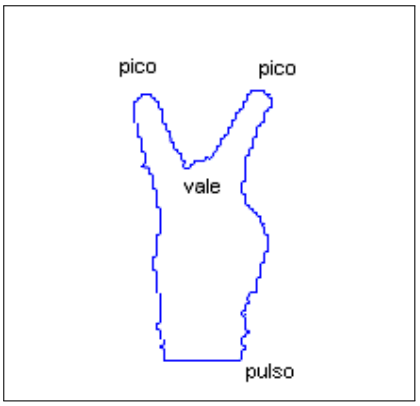


Figura 6.2: Contorno da mão.

6.1.1 Ordenação do Contorno

Como os métodos de detecção de características, implementados neste trabalho, aplicam-se sobre o contorno da mão, há a necessidade de que o mesmo esteja ordenado de forma que se possa percorrê-lo continuamente, ou seja, partindo-se de um ponto inicial qualquer, passa-se pelo seu vizinho, e assim, sucessivamente até encontrar novamente o ponto de partida.

Entretanto devido à natureza do algoritmo *floodfill*, utilizado para detectar o contorno, conforme explicado no Capítulo 5, os *pixels* não estão numa ordem seqüencial. Para resolver este problema foi implementado um algoritmo, cujo pseudocódigo é listado no Algoritmo 7 com o intuito de obter um contorno contínuo.

Ele recebe como entrada uma imagem binária contendo somente o contorno da mão, como um conjunto de pontos sobre uma imagem, que a partir de um ponto inicial, será percorrido no sentido anti-horário. A função *findNext()*, utilizada no Algoritmo 7, é responsável por determinar qual *pixel*, vizinho ao *pixel* analisado, é o próximo ponto do contorno, sendo que a ordem de procura por *pixels* não processados é mostrada na Figura 6.3. O algoritmo pára quando encontra pela segunda vez o ponto inicial.

Algoritmo 7 Algoritmo Ordenação do contorno - versão 1

```
Cria uma lista vazia L;
Determina o ponto P mais a esquerda e acima
Adiciona na lista L o ponto P associa um label S
enquanto P <> S faça
  P = findNext(P); /*acha o ponto subseqüente a P */
  Coloca P na lista;
fim enquanto
Retorne L;
```

6	7	8
5	*P*	1
4	3	2

Figura 6.3: Ordem de visita aos pixels vizinhos de um ponto *P*.

O Algoritmo 7 funciona sem problemas se o contorno não tiver "bifurcações", uma vez que o algoritmo *floodfill* retorna um contorno fechado e com 1 *pixel* de largura. Mas, devido às falhas na segmentação de pele, o contorno da mão não é uma linha com um caminho único. Na maioria dos casos o contorno apresenta bifurcações, como mostrado na Figura 6.4. Se o Algoritmo 7 chegar no ponto destacado na Figura 6.4, ele não terá para onde ir e, portanto, falhará.

Para contornar este problema foi desenvolvido um segundo algoritmo (Algoritmo 8) cujo princípio de funcionamento é o seguinte: se um *pixel* não possui vizinhos não processados, então retorna-se no contorno já percorrido até achar um *pixel* que tenha um vizinho não processado. Na Figura 6.5(a) são mostrados os pontos percorridos incorretamente, e na Figura 6.5(b) é mostrado o retorno que o algoritmo realiza até achar um *pixel* com um vizinho não processado. Outro detalhe importante, é que os *pixels* que compõe um caminho que leva a outro *pixel* que não possui nenhum vizinho a ser processado são descartados da lista de pontos.

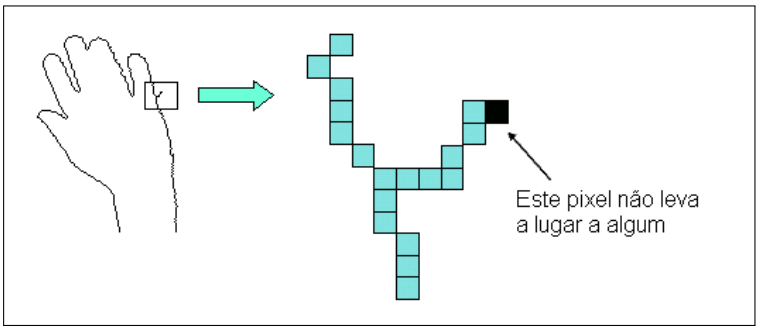


Figura 6.4: Falhas no contorno.

Algoritmo 8 Algoritmo Ordenação do contorno - versão 2

```
cria uma lista vazia de pontos L
Coloca-se como primeiro elemento da lista o pixel p mais a esquerda e acima
Marca-se o pixel p com a cor A
enquanto não processar todos os pixels e parar=falso faça
  pixel = L->last() /*pega o elemento do final da fila*/
  prox = findNext( pixel ) /*descobre o próximo pixel no contorno*/
  se cor(prox)==A  então
    sai do laço
  fim se
  se prox <> nenhum  então
    L->add(prox) /*adiciona mais um pixel na fila*/
  senão
    enquanto não achar um pixel válido e L cheia faça
      pixel = L->last()
      prox = findProx(pixel)
      se prox <> nenhum  então
        se  cor(prox) <> A   então
          L->add(prox)
        senão
          parar = verdadeiro
        fim se
      senão
        L->drop(prox)
      fim se
    fim enquanto
  fim se
fim enquanto
retorna L; /*contorno da mão ordenado*/
```

Uma vez que se tenha ordenado os pontos do contorno de forma que seja possível percorrê-lo como uma linha contínua, já é possível aplicar os métodos que analisam o mesmo e detectam os pontos característicos. Na próxima seção são apresentadas as estratégias testadas neste trabalho para determinar os pontos máximos e mínimos locais presentes no contorno, bem como os métodos empregados para a classificação dos mesmos em pontas de dedos ou vales entre os dedos.

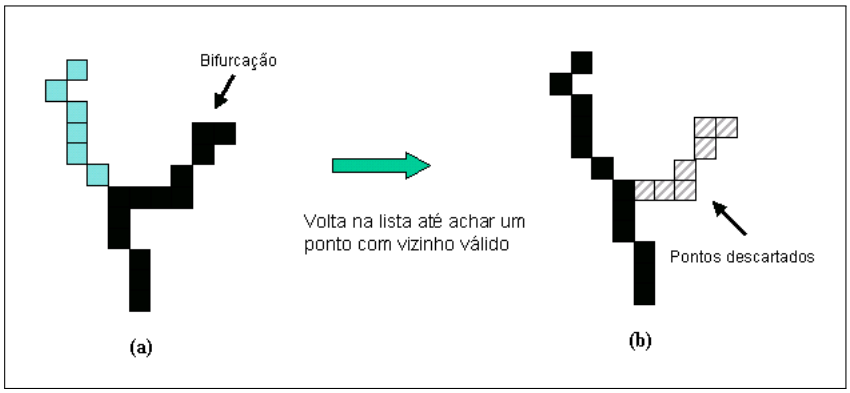


Figura 6.5: Descartando *pixels* fora do contorno.

6.2 Localizando Mínimos e Máximos Locais

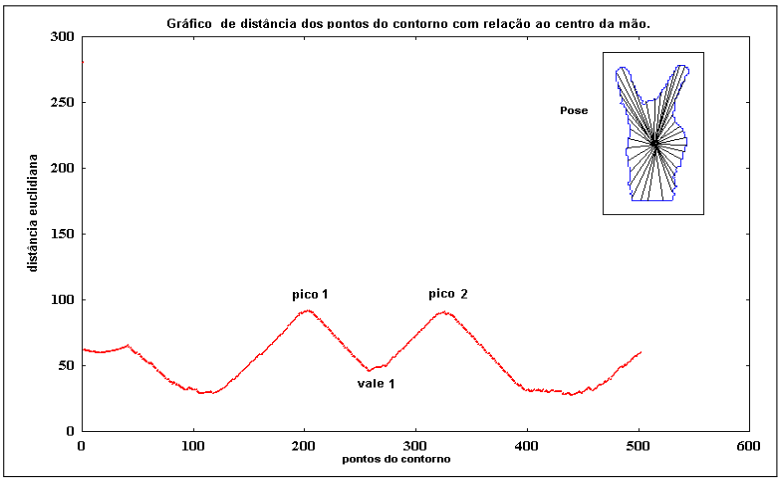
Nesta seção serão apresentadas duas técnicas de detecção de mínimos e máximos locais, que correspondem às pontas dos dedos e aos vales entre os dedos.

6.2.1 Curva de Distâncias

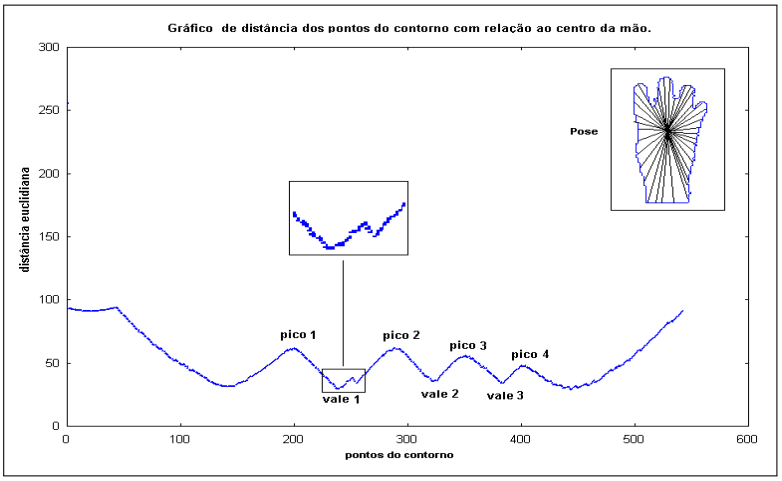
Uma técnica utilizada para detectar os mínimos os máximos locais de uma curva baseia-se em determinar a distância de cada ponto do contorno a um determinado ponto conhecido [46]. Para o caso da mão pode-se utilizar o centro da mão. Nas Figuras 6.6(a) e 6.6(b) tem-se dois exemplos da aplicação de tal técnica, tomando-se o centro da mão como ponto de referência e percorrendo o contorno no sentido anti-horário, a partir do ponto mais à direita e mais abaixo.

Na Figura 6.6(a), pode-se notar que existem dois máximos locais na curva que correspondem às pontas dos dedos e apenas um vale entre os picos, correspondendo ao vale entre os dedos. A mesma análise pode ser feita com relação ao gráfico da Figura 6.6(b), onde existem quatro máximos locais devido à presença de quatro dedos e três mínimos locais correspondendo aos vales entre os dedos. Também é possível observar, através da Figuras 6.6(a) e 6.6(b) que os mínimos locais, que correspondem aos vales da mão, aparecem entre dois máximos locais.

Um inconveniente desta técnica é a necessidade de um cálculo de distância para os pontos do contorno, antes de poder localizar os pontos característicos. Além disso deve-se tomar cuidado ao analisar a curva para não selecionar máximos e mínimos locais errados, como destacado na Figura 6.6(b).



(a) Curva de distâncias para a pose com dois dedos esticados.



(b) Curva de distâncias para uma pose com quatro dedos esticados.

Figura 6.6: Gráficos da Curva de Distâncias.

Na seção 6.2.2 será apresentada uma técnica mais sofisticada que permite encontrar os pontos de interesse da mão utilizando diretamente o contorno.

6.2.2 k-curvatura

A outra técnica usada para determinar os pontos característicos da mão é denominada k-curvatura ou k-curva, proposta por Segen e Kumar [53] *apud* [61]. Com esta técnica é possível localizar as pontas dos dedos e os vales entre os dedos sem a necessidade de calcular a distância de cada ponto do contorno com relação a um ponto conhecido, pois utiliza apenas o contorno. Exemplos de uso desta técnica podem ser encontrados no sistema MIME de reconhecimento de gestos [26] e no trabalho de Wu [61], que utilizou a k-curvatura para detectar os dedos. A definição de k-curva para um determinado ponto do contorno, é o ângulo formado entre os vetores $V_1 = [P(i-k), P(i)]$ e $V_2 = [P(i+k), P(i)]$, onde k é uma constante e $P(i)$ a lista de pontos do contorno da mão [53] *apud* [61].

Através da utilização de valores apropriados de k pode-se obter os mínimos e máximos locais analisando o ângulo entre os vetores V_1 e V_2, como mostrado na Figura 6.7(a-b). Nesta figura observa-se que ângulos pequenos correspondem a pontos onde existe uma mudança acentuada no gradiente da curvatura do contorno, ao passo que a Figura 6.7(c) ilustra uma situação contrária, onde não existe mudança significativa no gradiente da curva. A obtenção deste ângulo pode ser feita através do produto escalar entre os vetores V_1 e V_2.

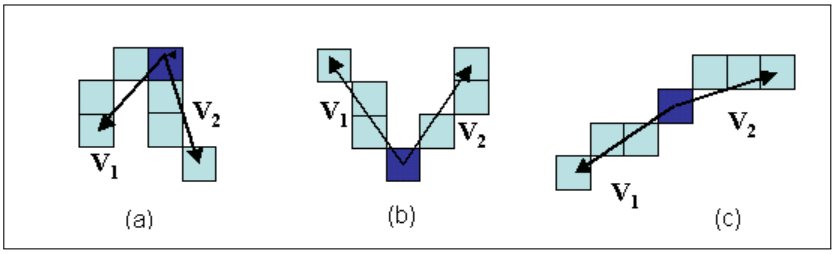


Figura 6.7: Ângulo entre os vetores.

Embora seja possível identificar os pontos de inflexão no contorno, ainda não se sabe a que característica cada um se refere. Como somente o valor do ângulo não é suficiente para realizar a classificação destes pontos, utilizam-se as direções dos vetores V_1 e V_2, para determinar se o ponto analisado é um vale da mão ou uma ponta do dedo. Considerando que neste projeto a mão estará sempre com os dedos abaixo do centro da mão, como mostrado na Figura 6.9(b), e que o contorno será processado no sentido anti-horário pode-se concluir que se os vetores V_1 e V_2 tem direção para baixo (Figura 6.8(a)) diz-se que o ponto analisado é um vale entre os dedos, por outro lado, se V_1 e V_2 tem direção para cima (Figura 6.8(b)) o ponto corresponde à ponta de um dedo.

Para calcular os ângulos entre os vetores o primeiro algoritmo implementado analisa o cosseno do ângulo, dado pela Equação 6.1 em um determinado ponto do contorno. Neste caso, quanto maior o valor do cosseno menor o ângulo. Portanto as características da mão corresponderão aos pontos com maiores cossenos.

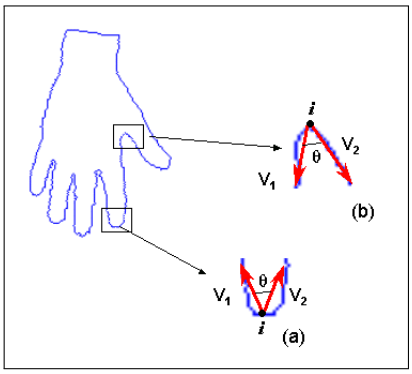
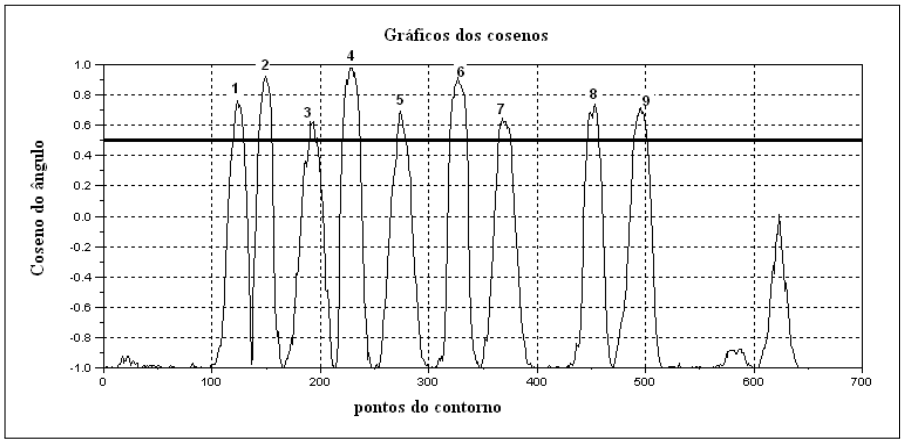


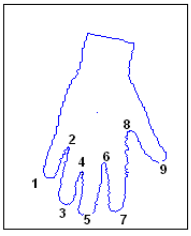
Figura 6.8: Classificação dos mínimos máximo locais.

$$cos(\theta) = \frac{V_1.V_2}{|V_1||V_2|} \tag{6.1}$$

Na Figura 6.9(a), tem-se um gráfico de uma k-curva construído a partir do valor dos cossenos em cada ponto do contorno da Figura 6.9(b). Como pode ser observado, os máximos locais da curva, correspondem aos pontos característicos da mão.



(a) K-curva obtida calculando-se o coseno entre os vetores.



(b) Pose utilizada.

Figura 6.9: Construção da k-curva através do produto escalar.

Uma alternativa para calcular o valor de cada ponto da k-curva, apresentada por Heckenberg [26], é baseada na diferença entre os ângulos dos vetores V_1 e V_2 com relação à horizontal, como ilustrado na Figura 6.10, calculada através da Equação 6.2, sendo que os ângulos de cada vetor com a horizontal são obtidos através das Equações 6.3 e 6.4.

$$\Theta(i) = \Theta_{+k}(i) - \Theta_{-k}(i) \tag{6.2}$$

$$\Theta_{+k}(i) = tan^{-1}\left[\frac{y(i+k) - y(i)}{x(i+k) - x(i)}\right] \tag{6.3}$$

$$\Theta_{-k}(i) = tan^{-1}\left[\frac{y(i) - y(i-k)}{x(i) - x(i-k)}\right] \tag{6.4}$$

Através da Equação 6.2 é possível identificar onde ocorrem os pontos de inflexão no contorno, pois esta equação mede a variação no gradiente da curvatura do contorno em um determinado ponto (Figura 6.10). Neste caso, valores grandes de $\Theta(i)$ corresponderão aos pontos de inflexão do contorno.

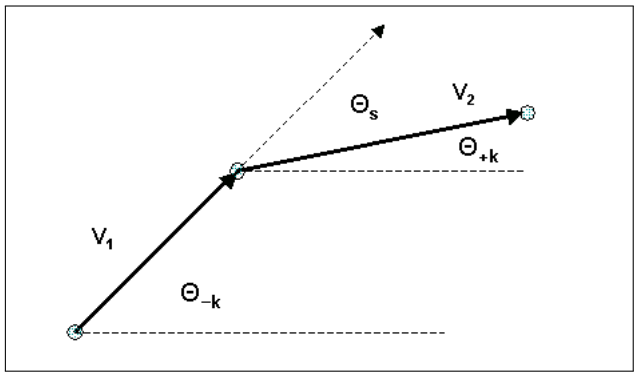


Figura 6.10: Diferença de ângulos.

Além disso, com esta técnica é fácil classificar os pontos entre ponta de dedo ou vale entre os dedos através do sinal do valor resultante da Equação 6.2, sendo que os valores positivos correspondem às pontas dos dedos e os valores negativos aos vales entre os dedos.

Para entender melhor como isto funciona é necessário analisar qual das duas situações, mostradas na Figura 6.11, ocorre quando um ponto característico é detectado. Para o caso da ponta do dedo os vetores V_1 e V_2 estão dispostos da maneira como é mostrada na Figura 6.11(a), ou seja, com V_1 apontando para baixo e V_2 apontando para cima. Levando em conta que o cálculo dos ângulos θ_{+k} e θ_{-k} com relação à horizontal, é obtido com o uso da tangente, Equações 6.3 e 6.4, este são positivos no primeiro e terceiro quadrantes e negativos no segundo e quarto quadrantes. Então, para o caso das pontas dos dedos o ângulo do vetor V_1 (θ_{-K}) é negativo enquanto que o ângulo do vetor V_2 (θ_{+k}) é positivo, como mostrado na Figura 6.12. Utilizando estes ângulos na Equação 6.2 obtêm-se um resultado positivo.

Da mesma maneira os vales entre os dedos podem ser detectados através do valores negativos da Equação 6.2. Observando a Figura 6.11(b) é possível perceber que o ângulo do vetor $V_1(\theta_{-k})$ é

positivo, pois o vetor pertence ao primeiro quadrante, e o ângulo do vetor V_2 (θ_{+k}) será negativo, como mostra a Figura 6.13. Portanto a Equação 6.2 resulta valores negativos quando o ponto analisado é um vale entre os dedos.

Na Figura 6.14(a), tem-se um gráfico dos valores da k-curva obtidos para cada ponto do contorno da pose mostrada na Figura 6.14(b), onde pode-se observar que os máximos locais da curva correspondem às pontas dos dedos e os mínimos locais correspondem aos vales entre os dedos.

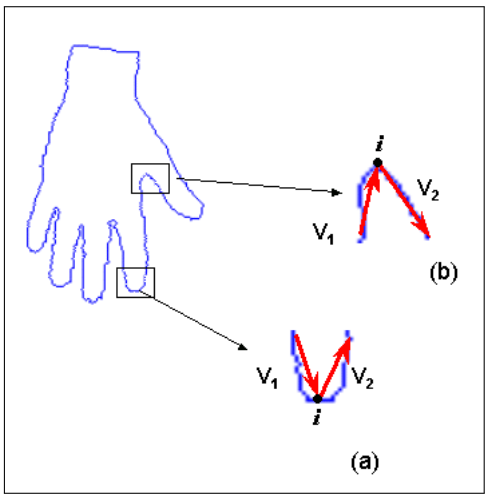


Figura 6.11: Identificação de vales e das pontas dos dedos.

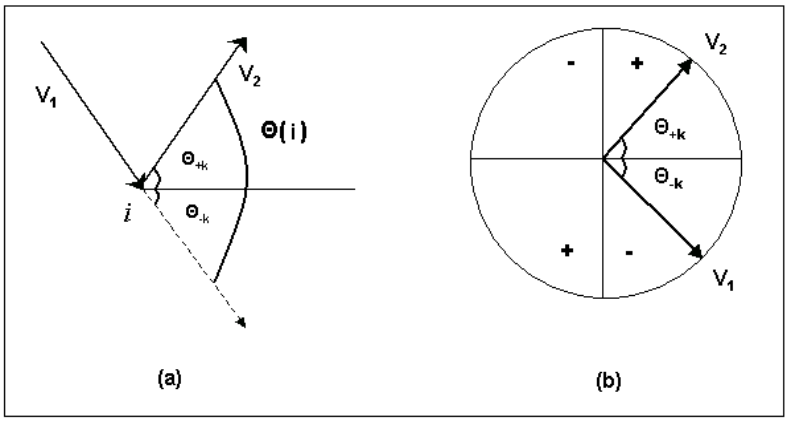


Figura 6.12: Detecção da ponta do dedo.

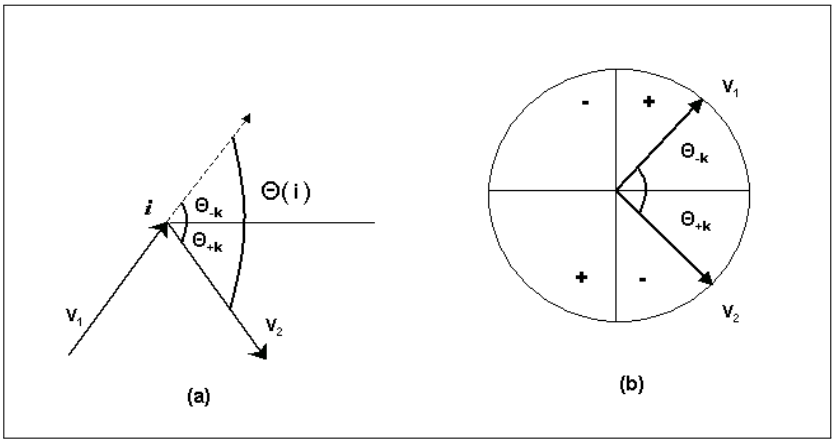
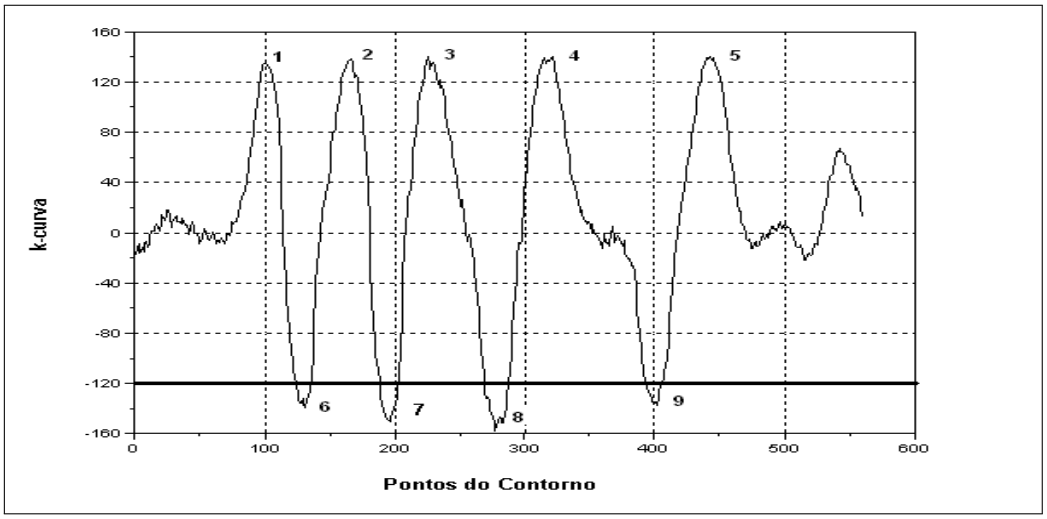
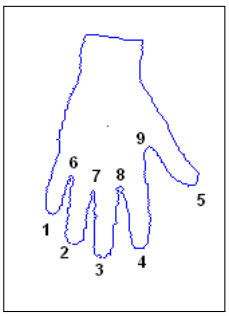


Figura 6.13: Detecção dos vales da mão.



(a) K-curva obtida através da diferenã de ângulos.



(b) Pose utilizada.

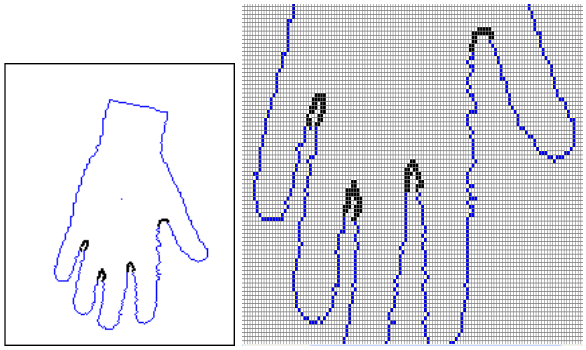
Figura 6.14: Traçado da k-curva para k=25.

6.2.3 Detecção dos Vales entre os Dedos

Neste projeto somente os vales entre os dedos são detectados. Para selecionar os pontos de interesse um algoritmo de limiar é utilizado para obter os pontos que possivelmente sejam os vales entre os dedos.

No caso do primeiro método, que utiliza o cosseno, seleciona-se os pontos cujo valor da função é maior do que um limiar l, de tal forma que somente alguns pontos candidatos, próximos aos máximos locais sejam selecionados, como visto na Figura 6.9(a). Em contrapartida, no segundo método, os pontos selecionados são aqueles cuja k-curva é menor que um limiar l, garantindo desta maneira que somente os pontos próximos aos mínimos locais sejam detectados, como mostrado na Figura 6.14(a).

Devido ao uso de um limiar vários pontos são detectados, pois possuem um valor semelhante de k-curva. Felizmente os pontos selecionados tendem a ocupar regiões bem definidas no contorno, formando um aglomerado de pontos em cada vale a ser detectado, como ilustrado na Figura 6.15. Para selecionar o ponto correto em cada vale entre os dedos, seleciona-se em cada aglomerado aquele que estiver mais próximo do centro da mão. Na Figura 6.16 tem-se um exemplo do resultado da detecção de vales entre os dedos.



(a) Pontos detectados.

(b) *Zoom* dos pontos detectados.

Figura 6.15: Pontos detectados como possíveis vales entre os dedos.

Embora a k-curva tenha-se mostrado uma técnica eficiente na detecção de pontos característicos, alguns cuidados devem ser tomados com relação à escolha do valor de k. Através de testes foi possível verificar que um valor de k entre 20 e 30 produz uma curva suave, como mostrado na Figura 6.14(a). Mas alguns problemas ocorrerão se a mão estiver muito afastada da câmera ou devido à escala utilizada, pois tais situações resultam em um contorno com poucos pontos e, um k com valor muito alto não permitirá detectar todos os vales da mão.

Além disso, como tem-se dois algoritmos para calcular a k-curva, um que utiliza o produto escalar e outro que utiliza a diferença de ângulos, um teste em termos de tempo de processamento de cada um foi realizado. Para que fosse possível comparar o desempenho dos algoritmos cada um deles foi alimentado com 10 imagens, sendo que cada imagem foi processada 5000 vezes e o tempo armazenado. Na Figura 6.17 tem-se um gráfico do tempo de processamento de cada imagem, onde é possível observar que o produto escalar é a melhor opção para ser usada em

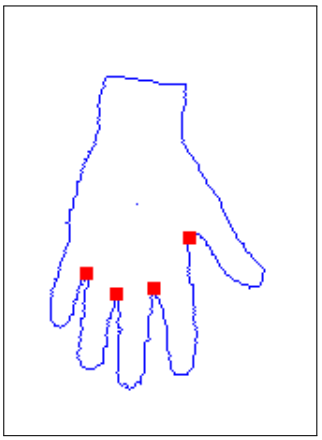


Figura 6.16: Juntas detectadas.

um sistema de tempo-real. Este algoritmo apresentou um tempo médio de processamento de 0.49 seg., ao passo que, o algoritmo que calcula a diferença de ângulos tem um tempo médio de processamento de 2.6 seg. Portanto neste projeto a detecção de características é realizada utilizando-se a k-curva obtida através do produto escalar.

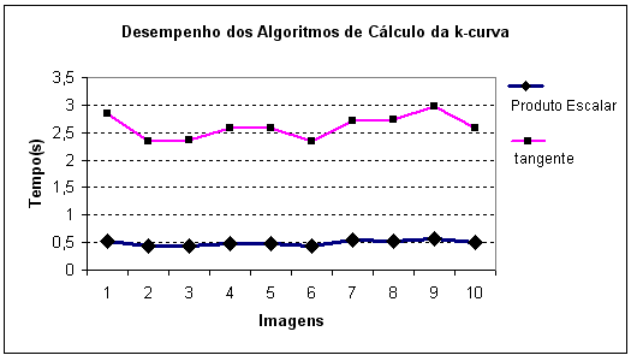


Figura 6.17: Desempenho dos algoritmos de k-curva.

6.3 Detectando o Pulso

Outros dois pontos que podem ser úteis para implementar um método de posição e orientação 3D da mão, devido ao fato de estarem presentes na maioria das poses, são os pontos extremos do pulso. Para detectá-los considera-se que a largura do antebraço é constante, e que a partir de um determinado momento, começa a aumentar quando se vai em direção ao centro da mão, como mostrado na Figura 6.18. O pulso, então, é o lugar onde ocorre essa mudança de largura. O algoritmo desenvolvido detecta os pontos P_1 e P_2 no antebraço e calcula a distância dos mesmos. Em seguida calcula-se as distâncias dos pontos subseqüentes a P_1 e P_2 de cada lado do contorno, (na Figura são chamados de P_3 e P_4) e calcula-se a diferença entre essas distâncias. Isso vai se repetindo para os demais pontos até que a diferença entre essas distâncias consecutivas esteja acima de um limiar d. Para facilitar a detecção destes pontos e o cálculo das distâncias a mão é

rotacionada de forma que fique alinhada com o eixo vertical. Na Figura 6.19 temos exemplos de alguns resultados da detecção de pulso. Vale lembrar que este método é o primeiro passo para a implementação de um procedimento automático de separação da mão do antebraço, semelhante ao método criado por Deimel [16].

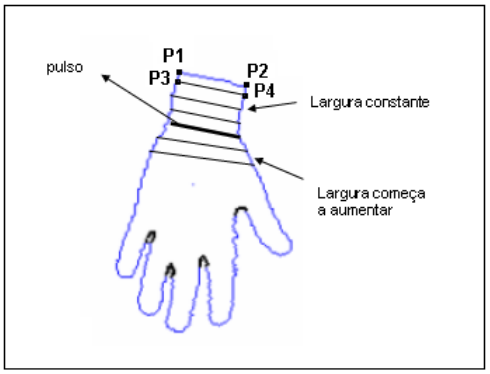


Figura 6.18: Localização do pulso.

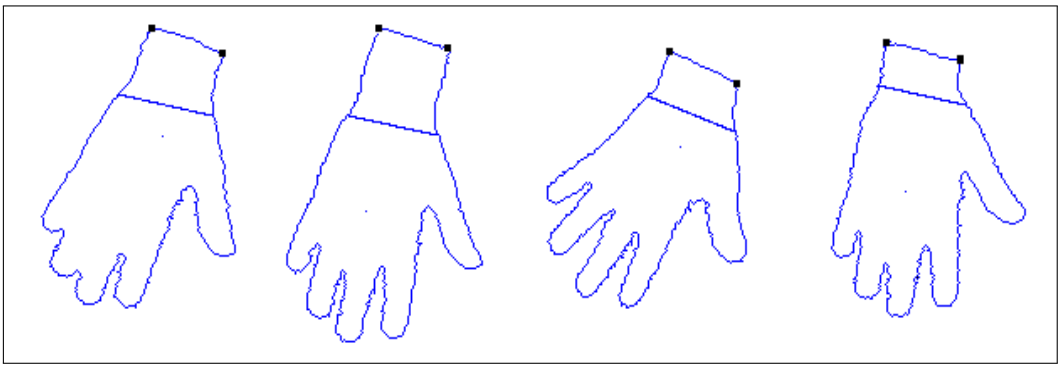


Figura 6.19: Exemplos de detecção do pulso.

Capítulo 7

Obtendo a Posição da Mão em 3D

A partir dos pontos característicos obtidos sobre a imagem da mão (centro da mão, pontas dos dedos, vales entre os dedos e o pulso), o próximo passo para o processo de rastreamento tridimensional é a obtenção da posição espacial de cada ponto identificado. Este processo comumente chamado de *reconstrução 3D* é realizado a partir de imagens captadas por uma ou duas câmeras.

Nas seções a seguir, são apresentados dois algoritmos baseados em duas câmeras e um algoritmo baseado na imagem de apenas uma câmera.

7.1 Posição 3D Utilizando Duas Câmeras

Para inferir a posição 3D de um ponto pode-se utilizar imagens de duas câmeras, ortogonais ou não, por meio de suas projeções nos planos das imagens. Para aplicar estes métodos é necessário conhecer o mesmo ponto nas duas imagens além de um ponto de referência que tenha uma coordenada previamente conhecida.

7.1.1 Uso de Câmeras Ortogonais

No caso de se utilizar câmeras ortogonais, na projeção gerada em cada câmera obtém-se um par de coordenadas. De acordo com o sistema de eixos apresentado na Figura 7.1, uma câmera superior pode gerar as coordenadas (x, y) do ponto e uma câmera frontal, as coordenadas (x, z).

Embora este método seja mais simples de implementar, ele apresenta alguns problemas para o caso da mão. O primeiro deles diz respeito diretamente à mão e o segundo, ao processo de geração da projeção da cena na imagem.

O primeiro problema acontece por que nem sempre é possível obter os mesmos pontos nas duas vistas. Visto que as câmeras são ortogonais pode ocorrer a situação mostrada na Figura 7.2, onde na vista superior foram detectados os vales entre os dedos, mas na vista frontal não se tem os pontos correspondentes.

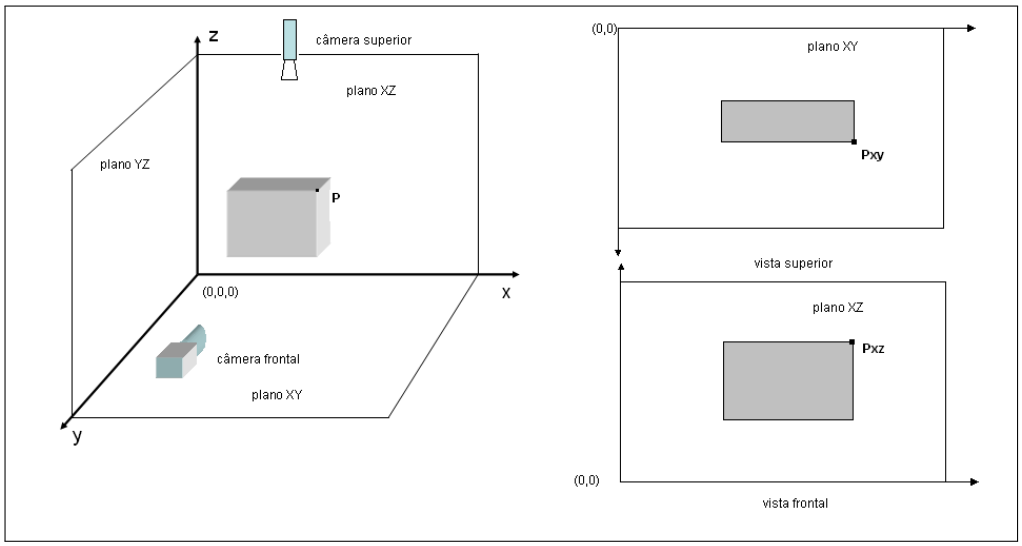


Figura 7.1: Câmeras ortogonais.

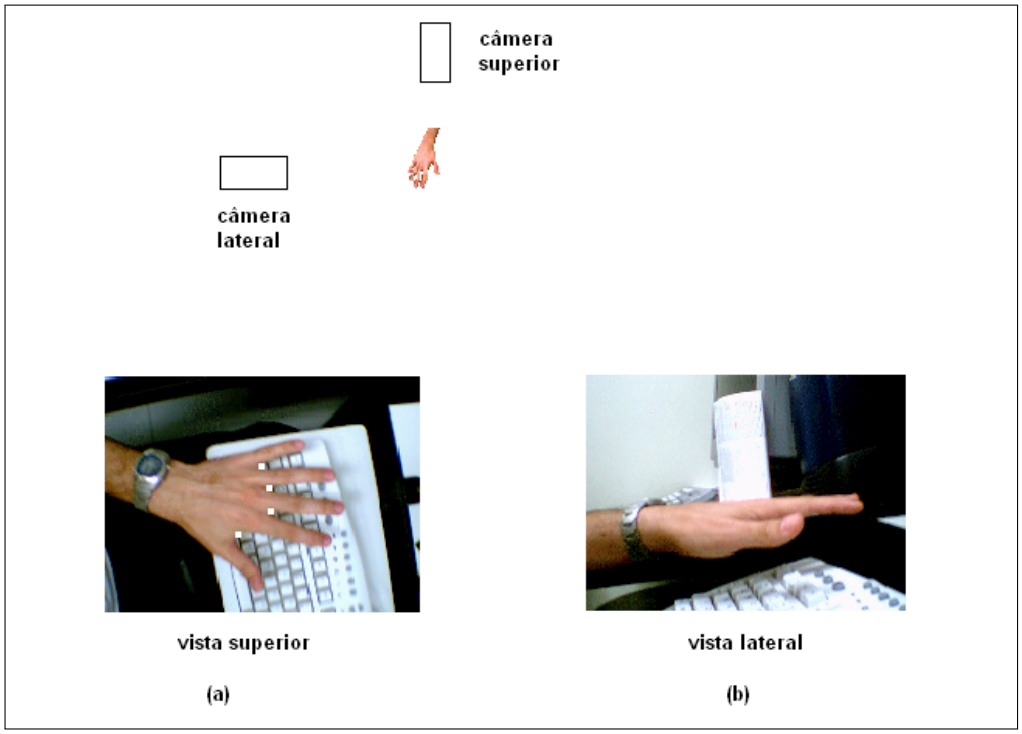


Figura 7.2: Pontos em uma vista não aparecem em outra vista.

O segundo problema com este método é causado pelas distorções criadas pelo processo de geração da projeção perspectiva realizado pela câmera. Por exemplo, na Figura 7.3 observa-se que dois pontos *A* e *B*, que possuem coordenadas *x* iguais no modelo do objeto, terão, na projeção frontal, coordenadas diferentes. Este problema é sempre minimizado se a distância dos pontos até a câmera é maior que a distância deles entre si e agravado se esta fica muito próxima dos mesmos.

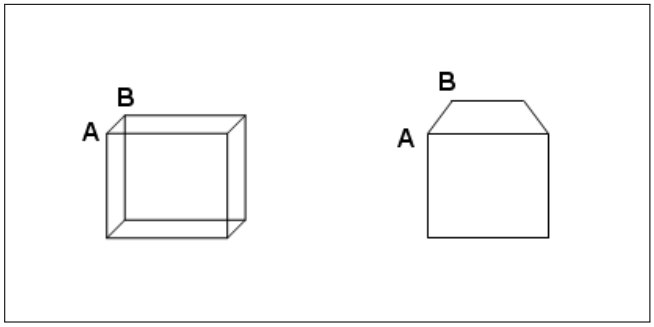


Figura 7.3: Problema causado pela perspectiva.

7.1.2 Uso de Câmeras Não-Ortogonais

Uma das idéias que ocorre quando se deseja obter medidas em 3D através de imagens é utilizar uma analogia do sistema visual estéreo do ser humano. Com o uso de duas câmeras, com configuração semelhante aos olhos humanos, ou seja, lado a lado e espaçadas entre si, como ilustrado na Figura 7.4, é possível calcular a profundidade relativa dos pontos. Para isso é necessário que as câmeras estejam calibradas (que se saiba a posição e a orientação delas do espaço) e que o mesmo ponto no espaço seja visto por ambas. Este último quesito é atingido mais facilmente do que com as câmeras ortogonais pois no caso em questão, como as câmeras estão próximas, elas tem quase a mesma visão do cenário.

Neste projeto, pontos específicos na mão são identificados nas duas imagens, como mostrado na Figura 7.5. A partir daí pode-se utilizar alguns destes pontos para localizar a posição da mão no espaço por meio de algoritmos de reconstrução 3D, como por exemplo o algoritmo de triangulação apresentado por Shapiro [54]. Esta estratégia não foi implementada devido a restrições de tempo, as quais este projeto foi submetido.

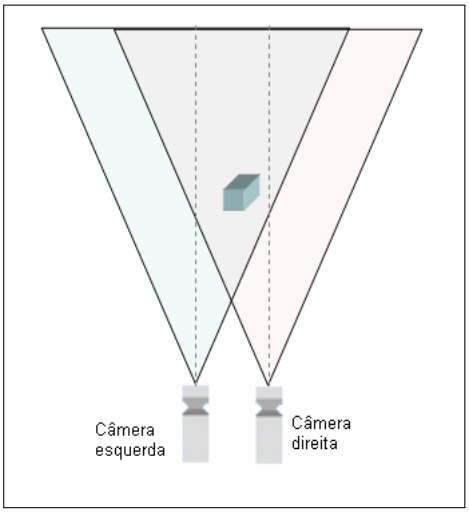


Figura 7.4: Sistema de visão estéreo.

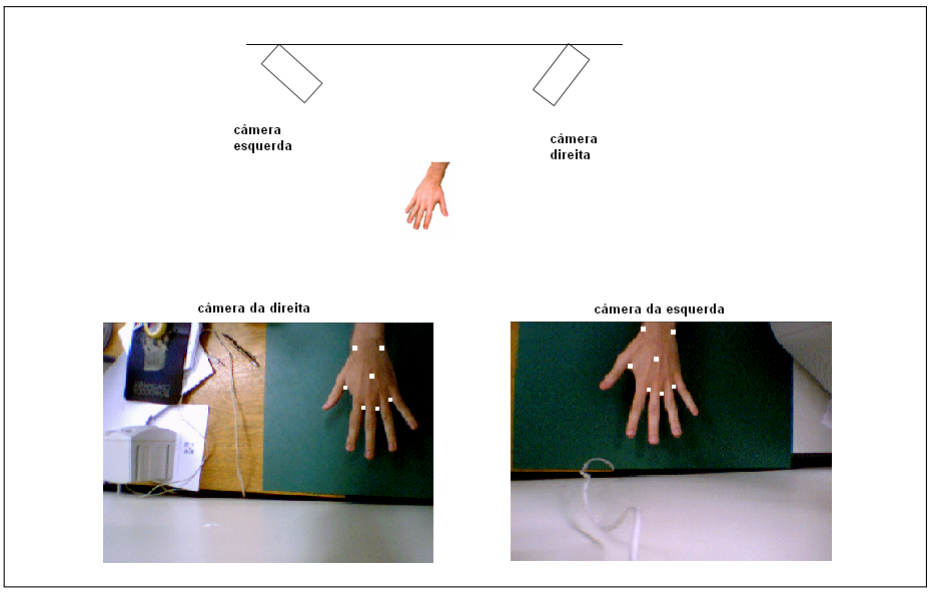


Figura 7.5: Par de imagens com os pontos de interesses detectados.

7.2 Posição 3D Utilizando Uma Câmera

Como foi mencionado na seção 7.1.2, o algoritmo de reconstrução pressupõe que o mesmo ponto pode ser obtido nas duas imagens. Se esta correspondência, entretanto, não for obtida com precisão, os dados de entrada do algoritmo estarão incorretos e o resultado acabará não satisfazendo os requisitos do rastreamento. Além disso, deve-se considerar que problemas de iluminação podem suprimir pontos em uma imagem, embora eles estejam presentes na outra.

Uma alternativa para calcular a posição 3D da mão é utilizar apenas uma câmera e detectar mais de um ponto sobre a imagem. A idéia baseia-se em um algoritmo chamado *Warp* [60]. Este algoritmo parte de um quadrado que está em uma posição conhecida e que ao ser "visto" por uma câmera, transforma-se em um quadrilátero, em face de sua nova posição em relação a esta câmera e também por causa da transformação perspectiva que é gerada pela projeção. Um exemplo destas transformações pode ser visto na Figura 7.6. Em (a) observa-se o quadrado em sua posição original, em (b) este quadrado sofre uma rotação e uma escala. Em (c) o quadrado sofre, adicionalmente, uma transformação perspectiva.

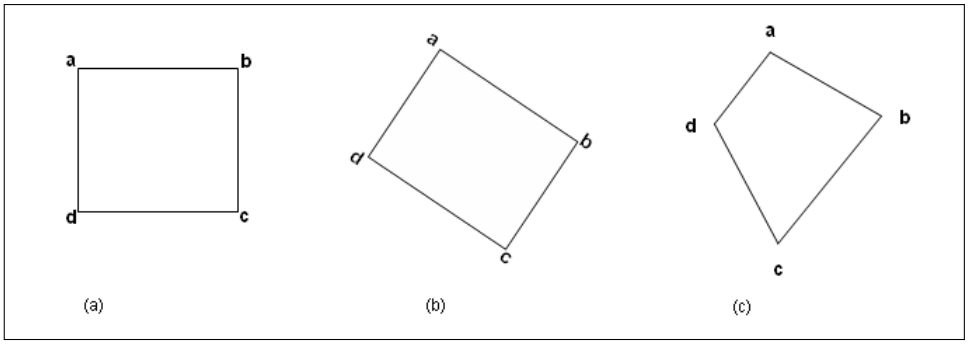


Figura 7.6: Exemplos de distorções ocorridas num objeto que se move na cena.

A partir desta nova imagem do quadrilátero, o algoritmo de *Warp* infere a *matriz de transformação geométrica e de perspectiva*, MT, capaz de levar os pontos do quadrado aos pontos do quadrilátero, através da Equação 7.1. Nesta equação o termo P_2 é um vetor-coluna e refere-se as coordenadas (x,y,z) dos pontos que compõe o quadrilátero e P_1 refere-se aos pontos do quadrado.

$$P_2 = P_1.MT \tag{7.1}$$

Ressalta-se aqui que o algoritmo não gera explicitamente as coordenadas dos pontos capturados pela câmera e nem a orientação destes no espaço. Entretanto, de posse da matriz de transformação é possível aplicá-la em um objeto que esteja sobre o quadrado original, de forma que este objeto sofra a mesma transformação ocorrida no quadrado. Este algoritmo é inclusive utilizado na ferramenta *ARToolkit* [4] que permite o rastreamento de objetos usando apenas uma câmera. Esta ferramenta, entretanto, exige a colocação de marcadores sobre o objeto a ser rastreado. A biblioteca segmenta a imagem e identifica os pontos extremos do marcador como pontos característicos a serem usados no algoritmo de *Warp*. Na Figura 7.7 pode-se observar

dois exemplos do uso do *ARToolkit* como rastreador. Note-se que na imagem da direita a mão é coberta por uma luva branca a fim de facilitar a segmentação.

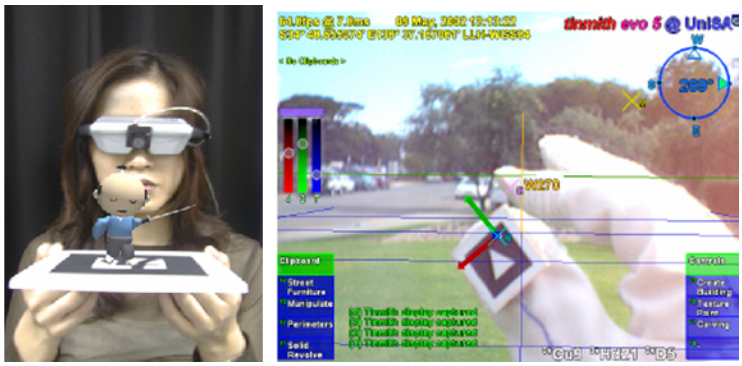


Figura 7.7: *ARToolkit* utilizado em ambientes de realidade aumentada (esquerda) e rastreamento de dedos (direita).

No presente trabalho por outro lado, os pontos característicos são obtidos diretamente da mão por meio da identificação dos vales existentes entre os dedos, sendo que a marca pode ser substituída por uma "marca virtual" como mostrado na Figura 7.8, utilizando também os limites do pulso para compor o quadrilátero, a partir do qual se infere a matriz de transformações. Vale lembrar que para usar este algoritmo é preciso que se tenha um quadrado como forma original e não um quadrilátero.

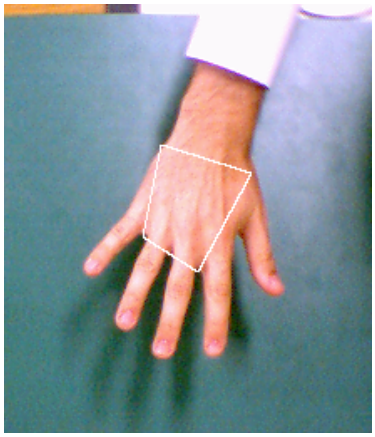


Figura 7.8: Quadrilátero inicial obtido através dos pontos característicos.

Entretanto, como no início do rastreamento não se consegue inferir um quadrado na mão do usuário, a partir dos pontos característicos e sim um quadrilátero, como visto na Figura 7.8 e assim durante todo o rastreamento. Diante deste cenário, optou-se por implementar um algoritmo que realiza um mapeamento de quadrilátero para quadrilátero, apresentado por Heckbert [25] *apud* [60]. Este algoritmo realiza tal mapeamento através da utilização de um quadrado intermediário, ou seja, um quadrilátero inicial é mapeado para um quadrado e, em seguida, o quadrado é mapeado para o quadrilátero final, conforme ilustrado na Figura 7.9. Devido a restrições de tempo do projeto implementou-se a solução mais simples, ou seja, em 2D, mas que permite verificar que a técnica funciona para os casos de translação, rotação e escala.

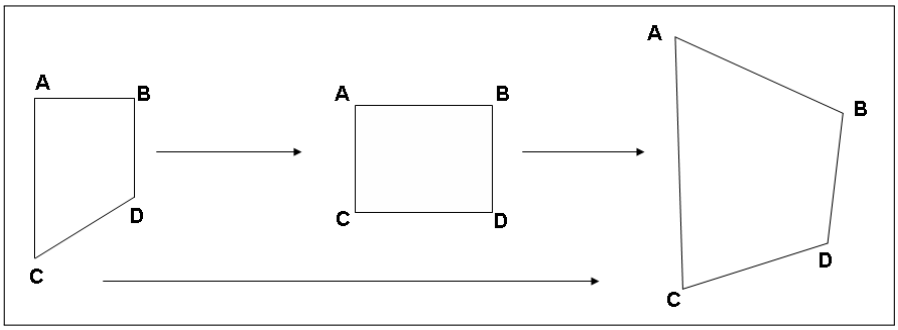


Figura 7.9: Mapeamento de quadrilátero para quadrilátero.

No caso do rastreamento da mão, o primeiro passo é obter a MT do quadrilátero para o quadrado padrão. Este procedimento é denominado *calibração* e requer que o usuário permaneça com a mão imóvel em uma determinada posição do espaço por um breve intervalo de tempo. Esta operação é realizada apenas uma vez, ou seja, a calibração consiste na determinação da MT que leva os pontos do quadrilátero inicial para o quadrado e em seguida mapeia-se o quadrado para os novos quadriláteros, obtidos nos quadros seguintes.

Com o objetivo de verificar o funcionamento do algoritmo de *warping*, tomou-se um losângulo, formado pelos pontos médios das arestas do quadrilátero inicial (Figura 7.10) que serve para analisar se o algoritmo consegue inferir, através da MT, as mudanças na translação, rotação e escala. O losângulo é obtido ainda na fase de calibração e seus pontos são mapeados para o quadrado padrão, de tamanho 100x100 *pixels* que, durante o rastreamento, estes novos pontos são mapeados para os novos quadriláteros obtidos. A medida que a mão se movimenta, rotaciona, aproxima ou se afasta da câmera o mesmo ocorre com o losângulo, lembrando que estas informações estão codificadas na MT.

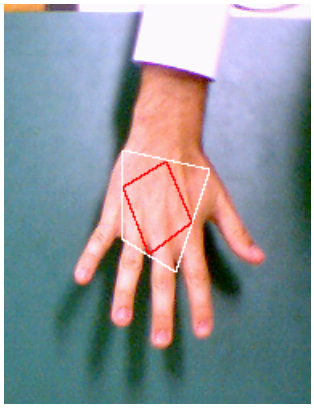


Figura 7.10: Posição inicial no momento da calibração.

Na Figura 7.11 tem-se um exemplo da translação, onde é possível notar que o losângulo é mapeamento corretamente em cada quadro. Em seguida na Figura 7.12, temos um exemplo de que a técnica funciona quando ocorre a rotação da mão. Na Figura 7.12(a) temos a posição inicial da mão e na Figura 7.12(b) temos a posição final da mão, após a rotação, onde o losâgulo é distorcido para refletir tal mudança. E, finalmente na Figura 7.13, temos um exemplo de mod-

ificação na escala devido a aproximação da mão em direção à câmera, pois é possível notar a diferença de tamanho do losângulo nas Figuras 7.13(a) e 7.13(b).

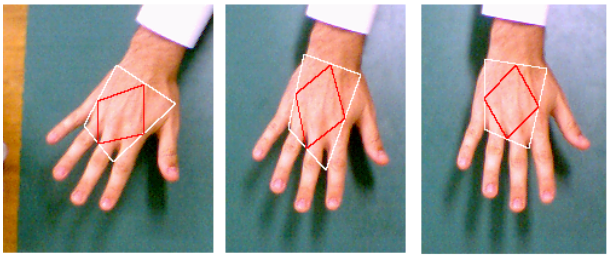
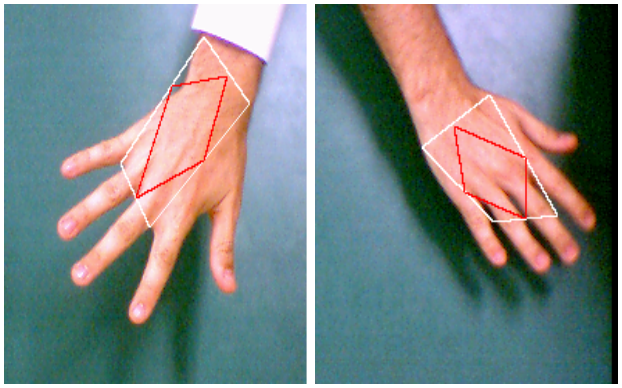
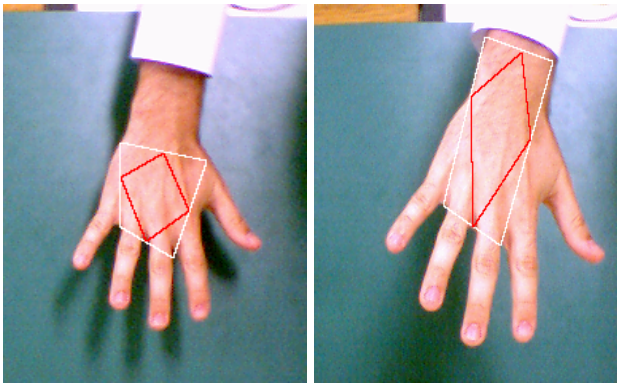


Figura 7.11: Translação.



(a) Posição inicial. (b) Posição após rotação.

Figura 7.12: Rotação.



(a) Posição inicial. (b) Posição final.

Figura 7.13: Exemplo de mudança de escala quando a mão se aproxima da câmera.

Embora esta técnica tenha apresentado bons resultados, alguns erros ocorrem devido a má qualidade na detecção de algumas características, principalmente o pulso, pois ao longo das imagens captadas pela câmera, ora o pulso está mais próximo da região da mão e ora está mais próximo da região do antebraço, como visto nas Figuras 7.13(a) e 7.13(b). Além disso, erros na

detecção do contorno fazem com que os vales entre os dedos sejam detectados fora dos limites da mão, como visto na Figura 7.11 e na Figura 7.12(b).

Capítulo 8

Considerações Finais

Neste trabalho foi apresentado um sistema de rastreamento de mão destinado a ser utilizado em ambientes virtuais, embora possa ser utilizado para implementação de interfaces intuitivas, como a arquitetura *câmera-projetor* ou um menu virtual, mencionados no Capítulo 2. A principal meta foi construir uma alternativa aos rastreadores magnéticos, muito utilizados em Realidade Virtual, para obter a posição e a orientação da mão do usuário. As vantagens do rastreador de mão baseado em imagens compreendem a sua facilidade de uso, sem a necessidade de uma carga de aprendizado grande para utilizá-lo e o conforto de não obrigar o usuário a usar algum tipo de *hardware* na mão.

O projeto foi dividido em três fases e, em cada uma delas, buscou-se soluções nas áreas de Processamento de Imagens e Visão Computacional, testando-se cada solução quanto à qualidade de resposta e tempo de processamento, uma vez que o rastreador de mão deve operar em tempo-real.

Na primeira fase deste projeto foram estudados os dois espaços de cores (RGB e HSV) mais utilizados para segmentação de pele, bem como foram implementados e testados 4 algoritmos de segmentação de pele e seus resultados apresentados. O uso da segmentaçãoimage de pele justifica-se pela necessidade de separar a mão do ambiente de trabalho, e pelas vantagens deste método que permite localizar a mão mesmo que ela mude de forma de uma cena para outra. Apesar do processo de segmentação de pele apresentar algumas falhas de classificação, estas foram superadas através da utilização de um algoritmo simples de preenchimento de áreas (*floodfill*).

Em seguida, na segunda fase, duas estratégias de obtenção da posição da mão foram apresentadas, o centro de massa e a Transformada da Distância utilizando a distância *Chamfer-3-4* (DCT). Neste projeto a DCT foi escolhida como estratégia de obtenção da posição da mão por fornecer respostas mais precisas mesmo com a presença do antebraço na cena e com diversas configurações da mão. Entretanto, recomenda-se o uso do centro de massa, caso se deseje trabalhar somente com a mão, descartando o antebraço e a mão em poses semelhantes aquelas apresentadas no Capítulo 3 (Figura 4.4). Além disso, este método é simples de implementar e rápido em termos de desempenho.

Quanto à técnica de momentos de imagens utilizada para calcular a orientação da mão, esta pode ser utilizada com várias câmeras, tornando possível o conhecimento da orientação da mão em vários planos como, por exemplo, para o caso da configuração de câmeras ortogonais,

mostrada na Figura 7.2, onde pode-se obter a orientação da mão nos planos *xy* e *xz*, sem nehuma modificação no algoritmo. Além disso, neste trabalho utilizou-se os momentos de imagens para calcular a orientação da mão a partir de seu contorno, ao contrário de muitos trabalhos que utilizam a área do objeto de interesse [13, 26, 50]. Desta forma o número de pontos utilizados pelo algoritmo cai drasticamente, culminando num aumento significativo de desempenho do mesmo.

Na Tabela 8.1 tem-se o tempo de processamento de cada fase do algoritmo de rastreamento. O algoritmo foi executado em um computador Pentium 4 1 GHz com 512MB de memória e uma placa de vídeo NVIDIA GeForce Ti 500 com 64MB de memória, uma webcam Creative WebCam 352x288, ligada a uma porta USB. Como o tempo de processamento mostrado na Tabela 8.1 varia de acordo a arquitetura sendo utilizada, a coluna *Tempo Relativo* serve para quantificar quão mais rápida é uma fase do algoritmo com relação a uma função hipotética que leve 1 segundo de processamento em qualquer arquitetura, ou seja, se na arquitetura X tem-se uma tarefa T que tome 1 segundo de processamento então a detecção de pele será executada 19,37 vezes mais rápida do que T.

Tabela 8.1: Tempo de execução de cada fase do algoritmo de rastreamento.

Fase do algoritmo	**Tempo de Processamento (seg.)**	**Tempo Relativo**
Segmentação	0,0516	19,37
Seleção da Maior área	0,0136	73,50
Preenchimento de Buracos	0,0059	170,28
Detecção da Região de Interesse	0,0119	84,37
Detecção da Borda da Mão	0,0067	149,65
Distância Chamfer	0,0166	60,34
Ordenação do Contorno	0,0015	679,47
Orientação	0,0004	2580,67
Detecção do Pulso	0,0048	208,62
Detecção dos Vales	0,0002	4235,30

Com os resultados obtidos, conclui-se que a idéia de implementar um rastreador baseado em imagens é viável, além de ser uma alternativa mais econômica, pois o *hardware* utilizado consiste de um computador com uma ou mais câmeras captando o movimento da mão do usuário.

Embora não se tenha concretizado o objetivo inicial de rastrear a mão em 3D, devido as restrições de tempo, as quais este projeto se submeteu, algumas aplicações podem ser desenvolvidas utilizando-se alguns dos algoritmos implementados. Exemplos de tais aplicações são mostradas a seguir:

- **Menu virtual**: um menu virtual semelhante ao *wearable menu* [63] pode ser implementado utilizando-se um algoritmo de segmentação de pele para localizar a mão em uma imagem do mundo real. Considerando que esteja sempre com o polegar virado para cima, pode-se detectar as pontas dos dedos e associar à cada uma delas um item do menu. O acionamento de um determinado item do menu pode ser feito detectando-se a flexão do dedo correspondente através da análise presença ou ausência de uma determinda ponta do

dedo, permitindo desta maneira, que um item do menu seja utilizado por meio de flexão dos dedos;

- ***Mouse* virtual**: considerando que apenas um dedo corresponderá ao ponteiro do *mouse* e o dedo indicador corresponde ao mesmo: usando algum algoritmo de segmentação de pele localiza-se a mão do usuário. Em seguida, por meio da k-curva detecta-se a ponta do dedo. O *click* do mouse pode ser implementado por meio da detecção da flexão do dedo, como feito por Zhu [63], ou pela permanência do mesmo por um determinado tempo em cima do objeto ou menu que se deseja utilizar. Esta aplicação poderia ser utilizada em ambientes de realidade aumentada ou sessões de *brain-storm* [6];

- ***Joystick* virtual**: este dispositivo pode ser implementado de maneira semelhante ao *mouse* virtual, mas com outro objetivo, por exemplo, pode-se utilizar um dedo esticado para apontar a direção, dentro de um ambiente virtual, para onde o usuário deseja se deslocar. Através de duas câmeras é possível descobrir a orientação da mão nos planos *xy* e *xz*, e permitir que o usuário se desloque em várias, que não sejam somente para cima e para baixo. Pode-se utilizar o *joystick* virtual em jogos de carta, onde todas as operações realizadas com as cartas consistem em pegá-las e arrastá-las com a mão. O ato de pegar uma carta pode ser implementado fazendo com que o usuário deixe por um determinado tempo a mão sobre a carta desejada, ou esticar um dedo, que será detectado através da k-curva. E, soltá-las pode ser feito da mesma maneira, mas em um local diferente de onde a carta estava inicialmente. Um exemplo de *joystick* virtual pode ser encontrado nos trabalhos de Freeman [19], que desenvolveu um sistema que permite controlar um carro de um simulador de fórmula 1 com a mão.

Como algumas fases do projeto contam com mais de um algoritmo para resolver o mesmo problema, na Tabela 8.2 listam-se quais as vantagens de desvantagens de determinados algoritmos implementados neste trabalho, com algumas observações com relação ao uso dos mesmos.

8.1 Trabalhos Futuros

No desenvolvimento desta pesquisa, várias idéias surgiram com intuito de refinar ainda mais a qualidade do sistema implementado. A seguir, algumas idéias para estender este trabalho:

- Melhorar a segmentação de pele levando em contas os vários tipos de pele, e melhorar o algoritmo de segmentação de forma que este se torne mais robusto às variações de iluminação. Pesquisas neste sentido podem ser encontradas no trabalho de Veznhevets [VEZ03];

- Além de melhorar o algoritmo de segmentação de pele com relação aos problemas de iluminação seria interessante que este utilizasse alguma estratégia de aprendizado sobre a pele do usuário, em tempo de execução, melhorando a qualidade da segmentação e não obrigando-o a utilizar somente informações estáticas provenientes de tabelas e arquivos;

- Melhorar a detecção do contorno da mão, uma vez que a má qualidade deste é a principal fonte de erros gerados pelos algoritmos de detecção de características;
- Desenvolver uma ou várias aplicações que utilizem a mão como dispositivo, com objetivo de testar os algoritmos deste projeto;
- Inserir mais uma câmera no sistema para que seja possível utilizar visão estéreo ou coletar informações sobre a mão com mais de uma vista;
- Permitir o uso das duas mãos, o que torna necessário o desenvolvimento de um modelo de mão, que diferencie a mão direita da esquerda. Atualmente este projeto não possui nenhum conhecimento sobre o que é mão, pois considera-se que, logo após a segmentação da pele, a maior área que permanece na imagem é a mão;
- Reimplementar o algoritmo *warp* para trabalhar com corrdenadas 3D permitindo obter os mesmos resultados que o *ARToolkit*, mas sem a necessidade de marcadores sobre a mão do usuário.

Tabela 8.2: Algoritmos implementados: vantagens e desvantagens.

Algoritmo	**Vantagens**	**Desvantagens**	**Observações**
Segmentação de Pele	• permite separar a mão em uma imagem independente de sua pose		• todos algoritmos implementados neste trabalho sofrem com mudanças de iluminação • para um ambiente controlado, onde se tem poucas cores de fundo com quase nenhuma mudança de iluminação, pode-se utilizar o algoritmo Limiar Simples, caso contrário, a melhor opção seria utilizar o algoritmo da Maior Freqüência com o espaço de cores HSV e o modelo de cores bin-histograma com bin de tamanho 4 ou 16
Centro de Massa	• rápido • fácil de implementar	• exige restrição do antebraço • não funciona para a maiorias das poses da mão	
Distância chamfer(DCT)	• rápido • não exige restrição do antebraço • funciona para a maioria das poses	• mais lerdo que o centro de massa • algoritmo seqüêncial, sem possibilidades de utilizar paralelismo	
Momentos de Imagens	• pode ser utilizado com várias câmeras sem alteração no algoritmo, de tal forma que é possível obter a orientação da mão em diferentes planos	• no caso da mão deve-se restringir o antebraço com o uso de uma camisa de manga comprida ou por meio de um algoritmo que separe a mão do antebraço	• neste trabalho utilizou-se os *pixels* que compõem o contorno na mão ao invéz de utilizar toda a superfície da mão, melhorando muito o desempenho do algoritmo de obteção da orientação da mão
k-curva	• detecta ponta dos dedos e vales entre os dedos diretamente do contorno da mão	• depende da qualidade da detecção do contorno	• a boa detecção dos pontos de interesse depende da escolha do valor de k. Este valor depende da resolução da imagem e da distância da mão com relação à câmera. Através de testes pode-se determinar o melhor valor de k

Referências Bibliográficas

[1] Absolute Technologies. Lista de Preços Para o Consumidor Final. Capturado em: `http://www.abs-tech/lista_precos_cf.pdf`. Último acesso: novembro 2005.

[2] AHMAD, S. A Usable Real-Time 3D Hand Tracker. In *Proceedings of the 28th IEEE Asilomar Conference on Signals, Systems and Computers*, pages 1257–1261, Palo Alto, USA, October 1994.

[3] ALBIOL, A.; TORRES, L.; DELP, E.J. Optimum Color Spaces for Skin Detection. In *Proceedings of the IEEE International Conference on Image Processing*, pages 122–124, Thessaloniki, Greece, October 2001.

[4] ARToolkit. ARToolkit Tutorials. Capturado em: `http://www.hitl.washington.edu/artoolkit/tutorials.htm`. Último acesso: dezembro 2004.

[5] BENES, J.A; BUENO, R.P. Hepatectomy Surgery Simulation. Capturado em: `http://grv.inf.pucrs.br/Pagina/Projetos/Hepatectomy/Hepato.pdf`. Último acesso: julho 2004.

[6] BERARD, F.; HANDENBERG, V.C. Bare-hand Human-Computer Interaction. In *Proceedings of the ACM Workshop on Perspective User Interfaces - PUI '01*, pages 15–16, Florida, USA, November 2001.

[7] BITTENCOURT, J.R; OSÓRIO F.D. O Uso de Redes Neurais Artificiais Na Detecção de Pele em Imagens Digitais Visando o Reconhecimento de Gestos. In *Anais do XI Seminário de Computação, SEMINCO*, pages 189–202, Blumenau, Brazil, August 2002. (In Portuguese).

[8] BOWMAN, D.A.; KRUIJIFF, E.; LAVIOLA, J.J.; POUPYEV, I. An Introduction to 3-D User Interface Design. *Presence*, 10(1):96–108, 2001.

[9] BRADSKI, G.R. Computer Vision Face Tracking for Use in Perceptual Interface. Microcomputer Research Lab, Intel Corporation. Capturado em: `http://developer.intel.com/technology/itj/q21998/articles/art_2.htm`. Último acesso: julho 2004.

[10] BRETZNER, L; LINDEBERG, T. Use Your Hand as a 3-D Mouse. In *Proceedings of the 5th European Conference on Computer Vision*, pages 141–157, Freiburg, Germany, June 1998.

[11] BROWN, C.M.; BALLARD, D.H. *Computer Vision*. Prentice-Hall Inc, New Jersey, 2002.

[12] BUTT, M.A.; MARAGOS, P. Optimum Design of Chamfer Distances Transforms. *IEEE Transactions on Image Processing*, 7(10):1477–1483, October 1998.

[13] CHAUMETTE, F. Image Moments: A General and Useful Set of Features for Visual Servoing. *IEEE Transactions on Robotics*, 20(4):713–723, August 2004.

[14] CHELLAPA, R.; WILSON, C.L.; SIROHEY, S. Human and Machine Recognition of Faces: A Survey. *Proceedings of the IEEE*, 83(5):705–741, May 1995.

[15] CORRADINI, A.; GROSS, H.M. Camera-Based Gesture Recognition for Robot Control. In *IEEE International Joint Conference on Neural Networks*, pages 133–138, Como, Italy, July 2000.

[16] DEIMEL, B. Development of a Stable Method to Remove the Forearm from Hand in Video Images. Master's thesis, Universität Dortmund, Dortmund, Germany, 1998.

[17] Embraer. Centro de Realidade Virtual. Capturado em: `http://www.embraer.com.br/portugues/content/empresa/tecnologia/default.asp`. Último acesso em: julho 2004. (In Portuguese).

[18] FENSTERSEIFER, M.; GARBACHI, T.; AYMONE, J.L.F. O Prédio da Faculdade de Arquitetura em Realidade Virtual. In *Anais do XIV Salão de Iniciação Científica da UFRGS*, pages 695–695, Porto Alegre, Brazil, March 2003. (In Portuguese).

[19] FREEMAN, W. T.; TANAKA, K.; OHTA, J.; KYUMA, K. Computer Vision For Computer Games. Technical Report TR1996-035, MERL - Mitsubishi Eletric Research Laboratories, 1996.

[20] FREEMAN, W.T.; WEISSMAN, C.D. Television Control by Hand Gestures. Technical Report TR1994-024, MERL - Mitsubishi Eletric Research Laboratories, 1995.

[21] FÉRIS, R.S. Rastreamento Eficiente de Faces em um Subespaço Wavelets. Master's thesis, Instituto de Matemática e Estatística, USP, São Paulo, Brazil, 2001. (In Portuguese).

[22] GÓMEZ, G.; MORALLES, E. Automatic Feature Construction and a Simple Rule Induction Algorithm for Skin Detection. In *Proceedings of the ICML Workshop on Machine Learning in Computer Vision*, pages 31–32, Sydney, Australia, July 2002.

[23] GONZALEZ, R.C.; WINTZ, P. *Digital Image Processing*. Addison-Wesley Publishing Company Inc, Reading, Massachusetts, 1977.

[24] GUNILLA, B. Distance Transformations in Digital Images. *Computer Vision Graphics and Image Processing*, 34(3):344–371, June 1996.

[25] HECKBERT, P. Fundamentals of Texture Mapping and Image Warping. Master's thesis, U.C. Berkeley, California, USA, 1989.

[26] HECKENBERG, D. MIME: A Video Based Hand Gesture Interface. Master's thesis, Department of Computer Science and Electrical Engineering, University of Queensland, Queensland, Australia, 1999.

[27] HORN, B.K.P. TSAI's Camera Calibration Method Revisited. Capturado em: `http://www.ai.mit.edu/people/bkph/papers/tsaiexplain.pdf`. Último acesso em: dezembro 2004.

[28] InterSense Inc. The New Standard in Motion Tracking. Capturado em: `http://www.isense.com`. Último acesso: novembro de 2004.

[29] ISARD, M; MACCORMICK, J. Hand Tracking for Vision Based Drawing. Technical Report 01, Visual Dynamics Group, Dept. Science, University of Oxford, 2000.

[30] JONES, M.J.; REHG, J.M. Detecting Adult Images. HP Labs. Capturado em: `http://crl.research.compaq.com/projects/vision/adult-detection.htm`. Último acesso: novembro de 2004.

[31] U. KÜHNAPFEL, H. ÇAKMAK, B. CHANTIER, MAAB H., G. STRAUSSS, C. TRANTAKIS, J. NOVATIUS, E.and MEIXENSBERGER, K. LEHMANN, H.J. BUHR, M. LAWO, and B.G. BRETTHAUER. HapticIO: Haptic Interfaces Systems For Virtual Reality Training in Minimally-Invasive Surgery. Capturado em: `http://informatiksysteme.pt-it.de/vr-ar-3/projekte/hapticio/paper_HAPTICIO.pdf`. Último acesso em: novembro 2004.

[32] KOPPER, R.A.P; SANTOS, M.C.C; PROCHNOW, D; PINHO, M.S; LIMA, J.C. Projeto e Desenvolvimento de Dispositivos de Geração de Tato. In *Proceedings of the VII Symposium on Virtual Reality*, pages 65–75, São Paulo, Brazil, October 2004. (In Portuguese).

[33] LIAO, S.X. *Image Analysis by Moments*. PhD thesis, The Department of Electrical and Computer Engineering The University of Manitoba, Winnipeg, Canada, 1993.

[34] LICSÁR, A.; SZIRÁNYI, T. Hand Gesture Recognition in Camera-Projector System. In *Proceedings of the International Workshop on Human-Computer Interaction*, pages 83–93, Praque, Czech Republic, May 2004.

[35] LIU, N.; LOVELL, B.C. MMX-Accelerated Real-Time Hand Tracking System. In *Proceedings of the Image and Vision Computing New Zealand - IVCNZ*, pages 381–385, Dunedin, New Zealand, November 2001.

[36] MALASSIOTIS, S.; STRINTZIS, M.G.; AIFANTI, N. A Gesture Recognition System Using 3D Data. In *Proceedings of the 1st International Symposium on 3D Data Processing, Visualization and Transmission*, pages 190–193, Padova, Italy, June 2002.

[37] McALLISTER, G.; McKENNA S.J.; RICKETTS, I.W. Hand Tracking for Behaviour Understanding. *Image and Vision Computing Journal*, 20(12):827–840, October 2002.

[38] MicroScriber. MicroScriber Digitizer Overview. Capturado em: `http://www.immersion.com/digitizer/`. Último acesso em: novembro 2004. (In Portuguese).

[39] MORRIS, T.; ELSHEHRY, S.O. Hand Segmentation from Live Video. In *Proceedings of the International Conference on Image Science, Systems, and Technology*, pages 6–10, Las Vegas, USA, June 2002.

[40] MYERS, B.A. A Brief History of Human Computer Interaction Technology. *ACM Interactions*, 5(2):44–54, March 1998.

[41] NEW, J.R.; HASANBELLIU, E.; AGUILAR, M. Facilitating User Interaction with Complex Systems via Hand Gesture Recognition. Capturado em `http://ksl.jsu.edu/publications/NewEtAl-ACMSE2003.pdf`. Último acesso em: novembro 2004.

[42] NORTH, M.M.; NORTH, S.M. Virtual Environments And Psychological Disorders. *Electronic Journal of Virtual Culture*, 2(4):37–42, September 1994.

[43] NORTH, M.M.; NORTH, S.M.; COBLE, J.R. Application: Psychotherapy, Flight Fear Flees. *CyberEdge Journal*, 6(1):8–10, January 1996.

[44] PARKER, J.R.; BAUMBACH, M. Visual Hand Pose Identification for Intelligent User Interfaces. In *Proceedings of the Vision Interface 2003*, pages 11–13, Nova Scotia, Canada, June 2003.

[45] PINHO, M.S.; DIAS, L.L.; MAZZORANI, A.; DUARTE, L.M. Experiências do Laboratório de Realidade Virtual da PUCRS. In *Anais do CACIC'99 - Congresso Argentino de Computação*, Tandil, Argentina, 1999. (In Portuguese).

[46] PLACEK, J.; CIGER, J. The Hand as an Ultimate Tool. In *Proceedings of the 16th Spring Conference on Computer Graphics*, pages 137–143, Budmerice, Slovakia, May 2000.

[47] REHG, J.; KANADE, T. DigitEyes: A Vision-based Human Hand Tracking. Technical Report CMU-CS-93-200, School of Computer Science, Carnegie Mellon University, 1993.

[48] RIOS, H.V. Human-Computer Interaction Through Computer Vision. In *Proceedings of the Conference on Human Factors and Computing Systems*, pages 59–60, Washington DC, USA, April 2001.

[49] RIVA, G. *Virtual Environments in Clinical Psychology and Neuroscience*. Ios Press, Amsterdan, Netherlands, 1998.

[50] ROCHA, L.; VELHO, L.; CEZAR, P.; CARVALHO, P. Image Moments-based Structuring and Tracking of Objects. In *Proceedings of the XV Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'02)*, pages 99–105, Fortaleza, Brazil, October 2002.

[51] ROY, S. State of the Art of Virtual Reality Therapy VRT in Phobic Disorders. *PsychNology Journal*, 1(2):176–183, November 2003.

[52] SATO,Y.; SAITO, M.; KIOKE, H. Real-Time Input of 3D Pose and Gestures of a User's Hand and Its Applications for HCI. In *Proceedings of the IEEE Virtual Reality Conference (VR'01)*, pages 79–86, Virginia, USA, November 2001.

[53] SEGENAND, J.; KUMAR, S. Human-Computer Interaction Using Gesture Recognition and 3D Hand Tracking. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 188–192, Chicago, USA, October 1998.

[54] SHAPIRO, L.G.; STOCKMAN, G.C. *Computer Vision*. Prentice Hall, New Jersey, 2001.

[55] SUTHERLAND, I.E. Head-Mounted Three-Dimensional Display. In *Proceedings of the AFIPS Conference Proceedings*, pages 757–764, Washington, DC, USA, December 1968.

[56] SZENBERG, F. *Acompanhamento de Cenas com Calibração Automática de Câmeras*. PhD thesis, Departamento de Informática, PUC Rio, Rio de Janeiro, Brazil, 2001. (In Portuguese).

[57] TOMAZ, F.; CANDEIAS, T.; SHAHBAZKIA, H. Improving Automatic Skin Detection in Color Images. In *Proceedings of the VIIth Digital Image Computing: Techniques and Applications - DICTA*, pages 419–427, Sydney, Austrália, December 2003.

[58] UNR Computer Vision Laboratory. Virtual GloveBox. Capturado em: `http://www.cse.unr.edu/~aerol/projecthome/vgx.htm`. Último acesso em: julho de 2004.

[59] VEZNHEVETS, V.; SAZONOV, V.; ANDREEVA, A. A Survey on pixel-Based Skin Color Detection Techniques. In *International Conference on Computer Graphics - GraphiCon 2003*, pages 85–92, Moscow, Russia, September 2003.

[60] WOLBERG, G. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA, 1990.

[61] WU, A; LOBO, N.A; SHAH, M. A Virtual 3D Blackboard: 3D Finger Tracking using a Single Camera. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 536–543, Grenoble, France, March 2000.

[62] ZELINSKY, A.; O'HAG, R. Finger Track - A Robust and Real-Time Gesture Interface. *Lecture Notes In Computer Science*, 1342:475–484, 1997.

[63] ZHU, X.; YANG, J.; WAIBEL, A. Segmenting Hand of Arbitrary Color. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 446–453, Grenoble, France, March 2000.