UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

CAROLINE CONTI

Rastreamento de movimentos da mão humana usando Visão Computacional

São Carlos
2010

Caroline Conti

Rastreamento de movimentos da mão humana usando Visão Computacional

Trabalho de Conclusão de Curso apresentado
à Escola de Engenharia de São Carlos, da
Universidade de São Paulo

Curso de Engenharia Elétrica com ênfase em
Eletrônica

ORIENTADOR: Professor Doutor Adilson Gonzaga.

São Carlos
2010

Ficha catalográfica preparada pela Seção de Tratamento
da Informação do Serviço de Biblioteca – EESC/USP

C762r

Conti, Caroline
Rastreamento de movimentos da mão humana usando Visão Computacional / Caroline Conti ; orientador Adilson Gonzaga. -- São Carlos, 2010.

Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2010.

1. Rastreamento. 2. Segmentação de vídeo. 3. Gestos de mão. I. Título.

Dedico todo meu esforço nestes anos de estudo e aprendizado à minha família.

Agradecimentos

Agradeço primeiramente aos meus familiares que me acompanharam e apoiaram em todas as fases sem cobranças, com compreensão e carinho. À Camila pelos auxílios em todas as horas, seja psicológico ou com seus desenhos e correções. Aos amigos que estiveram presentes nos momentos em que eu precisava de um ombro amigo para desabafar. Às meninas das Repúblicas Mãe Joana e Monti e ao pessoal do alojamento USP São Carlos que foram a minha família nos anos de faculdade. Ao Professor Dr. Adilson Gonzaga pela orientação e atenção ao trabalho.

E a todos que de certa forma contribuíram para a conclusão deste trabalho.

"É melhor atirar-se à luta em busca de dias melhores, mesmo correndo o risco de perder tudo, do que permanecer estático, como os pobres de espírito, que não lutam, mas também não vencem, que não conhecem a dor da derrota, nem a glória de ressurgir dos escombros. Esses pobres de espírito, ao final de sua jornada na Terra não agradecem a Deus por terem vivido, mas desculpam-se perante Ele, por terem apenas passado pela vida". (Bob Marley)

RESUMO

Este trabalho propõe um método baseado em visão computacional para realizar o rastreamento do movimento dos dedos de uma mão pela utilização de uma câmera de baixa resolução. Após a captação da imagem, a segmentação ocorre por meio da subtração desta pelo fundo. A decisão por este método de segmentação foi baseada em um estudo comparativo de vários algoritmos aplicados a um banco de dados de imagens. A detecção da mão considera seu contorno, as pontas dos dedos e o centro da palma. O reconhecimento do contorno é baseado na aproximação pelo polígono de mínimo perímetro. As pontas dos dedos são encontradas pelo algoritmo de k-curvatura. Para determinar o centro da mão utiliza-se o conceito de Transformada da Distância Euclidiana. O rastreamento considera a posição das pontas dos dedos em relação ao centro da palma da mão e a partir disto, é possível prever o movimento de abrir e fechar os dedos. Os resultados demonstraram que é possível de forma simples implementar um sistema base de interação humano-computador.

Palavras-chave: rastreamento, segmentação de vídeo, gestos de mão

ABSTRACT

This paper proposes a computer vision system for hand-fingers tracking from images captured by a low resolution webcam. The segmentation algorithm was based on segmentation by background subtraction. The decision by this method was based on a comparative study of various algorithms applied to an image database The hands detection finds its contours, the fingertips and palm center. Hand contour recognition is obtained by approximation for minimum-perimeter polygon. The k-curvature algorithm is used to find fingertips and applies the concept of the Euclidean Distance Transform to determine the palm center. The tracking regards the distance from fingertips to center, in order to predict the motion of opening or closing fingers. The results demonstrated that it is possible to implement a base system for human-computer interation using simple resources.

Key words: tracking, video segmentation, hand gesture.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO 1

1 INTRODUÇÃO

A Interação Humano-Computador (IHC) é considerada um conceito novo, mas vem aumentando rapidamente e de forma constante nas últimas três décadas, atraindo profissionais de várias áreas e incorporando diversos conceitos e abordagens (CARROLL, 2009).

A inclusão digital bem como as grandes inovações nesta área abriu várias opções para que esta interação não ficasse limitada à utilização tradicional do monitor, teclado e *mouse*. Neste contexto, o conceito de visão computacional, rastreamento e reconhecimento de voz possibilitaram uma interação mais natural (TRUYENQUE, 2005) e flexível entre o homem e o computador.

A utilização da mão e de seus gestos é a maneira mais intuitiva para interação do homem com o computador. Atualmente os métodos mais eficientes utilizam alguns dispositivos baseados em sensores ópticos, magnéticos ou mecânicos ligados ao computador que são, em sua maioria, sofisticados (*data gloves*), necessitam de ambientes especiais para serem operados e são comercializados a um preço elevado.

É possível e interessante que esta interação pelas mãos seja cada vez mais natural e independente de dispositivos que a pessoa deva vestir ou manipular, portanto, novas interfaces devem ser desenvolvidas.

Recentemente há um crescente interesse em introduzir a visão computacional como meio de interação humano-computador. O rastreamento de gestos humanos representa importantes possibilidades e fatores como a proliferação atual de computadores de alta capacidade, as câmeras de boa qualidade a baixo custo, além da crescente necessidade de análise automatizada de vídeo, geraram um maior interesse de desenvolver algoritmos de rastreamento (BEN-ISRAEL, 2007).

Alguns desafios, entretanto, devem ser superados. A recuperação de um movimento da mão, com todos os seus graus a partir de imagens auto-ocludidas inevitáveis é um problema de grande motivação para ser resolvido, uma vez que cria a possibilidade de aplicar estes algoritmos à operação em tempo real de equipamentos à distância, algumas vezes instalados em locais inseguros e perigosos e sem o contato humano (GONZAGA, 2010).

1.1 OBJETIVO

Este trabalho propõe como objetivo principal um método eficiente de reconhecimento e rastreamento de gestos dos dedos de uma mão humana, com o intuito de ser utilizado como base interação humano-computador. Neste método poucas considerações serão feitas em relação a aspectos como iluminação e cor de objetos, de maneira que esta interação seja realizada de forma mais natural possível. Porém, deve mostrar também que é possível, com o uso de um sistema simples, realizar o rastreamento com eficiência e precisão.

Com este intuito, o trabalho contempla um problema de visão computacional e sua solução desde a aquisição de imagens, segmentação, reconhecimento, rastreamento de gestos e interpretação dos resultados.

A aquisição de imagens considera a utilização de dispositivos de baixo custo como câmeras Web com baixa resolução e, por algumas vezes, serão realizados ajustes baseados nas características dos ambientes. Tais ajustes, entretanto, são pequenos e feitos numa fase de instalação, de forma a não comprometer a usabilidade do sistema.

A segmentação, neste trabalho, considera o reconhecimento de uma mão limpa. Isto significa que não são utilizados dispositivos (mecânico, magnético ou óptico) ou marcadores colocados na mão para interagir com o computador. Além disto, esta fase do projeto foca a avaliação de métodos de segmentação, a fim de obter um bom desempenho no resultado final do trabalho, isto é, que responda de maneira eficiente e gere parâmetros confiáveis a serem utilizados na próxima fase do projeto.

Vale destacar que o algoritmo de reconhecimento é baseado somente em contornos, portanto rápido o bastante para se trabalhar em tempo real, e este trabalho estará restrito aos gestos executados apenas pelas mãos, mais especificamente, pela mão direita. A primeira preocupação, portanto, é implementar algoritmos que tenham um bom desempenho em tempo real sem um prévio treinamento para comparações de cor de pele e de formas de mãos.

O rastreamento tem por objetivo mostrar que através do uso de diferentes técnicas de de Visão Computacional combinadas é possível reconhecer e classificar corretamente o posicionamento dos dedos em uma sequência de vídeo. Para isto, desenvolveu-se algumas aplicações a fim de mostrar o sistema em funcionamento.

Algumas etapas são definidas como base e levam em consideração alguns aspectos do trabalho realizado por Ribeiro (2006), para que se dê continuidade aos estudos contemplados no projeto Kanguera. Estas etapas são apresentadas a seguir:

1. Aquisição de imagem quadro a quadro por uma webcam;
2. Processamento de cada quadro utilizando segmentação para selecionar os pixels pertencentes à mão;
3. Pós processamento para melhorar a a imagem segmentada;
4. Extração das características da mão pertencente a cada quadro que serão utilizadas como parâmetros para o rastreamento;
5. Normalização do movimento da mão;
6. Determinação dos parâmetros utlizados para caracterizar o rastreamento.
7. Rastreamento do movimento de cada dedo;
8. Tratamento de oclusões;

1.2 ESTRUTURA DO DOCUMENTO

O presente trabalho está organizado segundo a sequência de capítulos relacionados que se segue:

Capítulo 1: Introdução, Apresenta uma introdução ao tema desta monografia;

Capítulo 2: Rastreamento de Objetos. Neste capítulo são apresentados alguns conceitos sobre o rastreamento de objetos em cenas de vídeos e sua utilização na interação humano-computador. Alguns trabalhos realizados envolvendo rastreamento de mão humana são apresentados de forma a contextualizar o objetivo proposto.

Capítulo 3: Segmentação de Imagens. Este capítulo trata do conceito de processamento da imagem pela segmentação, apresentando alguns tipos de segmentação e o conceito de pós-processamento por meio da utilização de operadores morfológicos.

Capítulo 4: Materiais e Métodos. Este capítulo contém o desenvolvimento do trabalho, isto é, apresenta a metodologia adotada bem como o material utilizado para que os objetivos propostos fossem alcançados.

Capítulo 5: Resultados e Conclusões. Este capítulo apresenta alguns resutados conseguidos pela implementação da metodologia, bem como a análise desses resultados e considerações de possíveis melhorias e conclusões.

CAPÍTULO 2

2 RASTREAMENTO DE OBJETOS

Como já dito, o rastreamento de gestos humanos propicia importantes mecanismos para o conceito de Interação Usuário-Computador (IHC). Além disto, com a tecnologia atual de computadores de alta capacidade e câmeras de boa qualidade a baixos custos, surge um maior interesse em algoritmos de rastreamento (BEN-ISRAEL, 2007). Destaca-se aqui, três passos para análise em um vídeo:

(a) Detecção do objeto de interesse;

(b) Rastreamento do objeto quadro a quadro;

(c) Análise do objeto rastreado para interpretar seu comportamento.

A utilização de rastreamento é pertinente às funções de reconhecimento baseado em movimento e reconhecimento de gestos para controlar as entradas de dados para os computadores.

O problema do rastreamento de um objeto pode ser definido como o problema de estimar a trajetória de um objeto no plano da imagem enquanto se move ao redor de uma cena (YILMAZ et al., 2006). Em outra palavras, o algoritmo de rastreamento atribui um nome ou *label* para cada objeto rastreado na cena e o reconhece em cada quadro do vídeo.

No caso da mão humana, os principais desafios no rastreamento são:

- A diferenciação entre as áreas de fundo e o primeiro plano;
- A diferenciação entre os objetos monitorados e outros objetos em movimento na mesma cena e extração de características relevantes ao rastreamento;
- Mudanças na iluminação, que provocam mudanças nos objetos monitorados;
- Normalização dos objetos que mudam com a distância da câmera;
- Oclusões entre os objetos presentes na cena.

O rastreamento pode ser simplificado através da imposição de restrições ao movimento e aparência dos objetos. Quase todos os algoritmos de rastreamento supõem que o movimento do objeto é constante e sem alterações bruscas. Pode-se restringir ainda mais considerando a velocidade ou a aceleração constante do movimento. Além disto, o conhecimento prévio sobre

o número e o tamanho dos objetos, ou a aparência e forma de um objeto também podem ser usados para simplificar o problema (YILMAZ et al., 2006).

Várias abordagens para controle de objeto têm sido propostas. Estas diferem umas das outras principalmente com base na maneira como abordam questões sobre qual objeto de representação é adequado para monitoramento, quais características da imagem devem ser usadas e como deve ser o movimento, a aparência e a forma do objeto a ser modelado. As respostas dependem do contexto, do ambiente em que o monitoramento é realizado e do uso final para o qual as informações de monitoramento estão sendo procuradas.

A primeira questão é definir uma representação adequada do objeto, descrever como deve ser caracterizado o objeto em cada cena. A questão seguinte é a seleção de características da imagem que serão utilizadas como parâmetros para o rastreamento. E finalmente, a questão é escolher uma estratégia para o rastreamento.

2.1 REPRESENTAÇÃO DOS OBJETOS

Em um cenário de rastreamento, um objeto pode ser definido por qualquer característica que seja interessante para a posterior análise. Por exemplo, no caso das pessoas andando em uma estrada, pode ser importante a utilização de um template para caracterizar o formato do corpo em movimento, por isso os objetos podem ser representados por suas formas e aparências.

Nesta seção, é descrita a forma do objeto por meio de representações comumente empregadas para o acompanhamento. As formas de representação mais comumente usadas são (YILMAZ et al., 2006):

- Pontos: O objeto é representado por um ponto, isto é, o centróide, como mostrado na Figura 2.1(a), ou por um conjunto de pontos, Figura 2.1(b). Em geral, a representação de ponto é adequada para acompanhar os objetos que ocupam pequenas regiões em uma imagem;
- Formas geométricas: a forma do objeto é representada por um retângulo, elipse ou outra abstração geométrica, como exemplificado na Figura 2.1(c), no caso de simples objetos rígidos e também para monitorar objetos não-rígidos;
- Silhueta e contorno do objeto. O objeto pode ser caracterizado pelos pontos que definem o contorno de um objeto, como na Figura 2.1(g) e 2.1(h). A região no interior do contorno é chamada silhueta do objeto, como mostra a Figura 2.1(i). Silhueta e

representações de contorno são adequados para o acompanhamento de formas nao rígidas complexas, como por exemplo a mão humana;

- Modelos de forma articulada: Objetos articulados são compostos pelas partes do corpo que são unidas pelas juntas. Por exemplo, o corpo humano é um objecto articulado com tronco, pernas, mãos, cabeça e pés ligados por articulações, como mostra a Figura 2.1(e). A relação entre duas partes é governada por modelos cinemáticos de movimento, por exemplo, ângulo articular. Desta maneira, a mão também poderia ser representada por suas porções articuláveis e as partes constituintes poderiam ser modeladas por cilindros ou elipses.
- Modelos de esqueleto. Este modelo é comumente usado como uma forma de representação para reconhecer objetos e pode ser utilizado para modelar objetos articulados e rígidos, veja exemplo de sua representação na Figura 2.1(f).

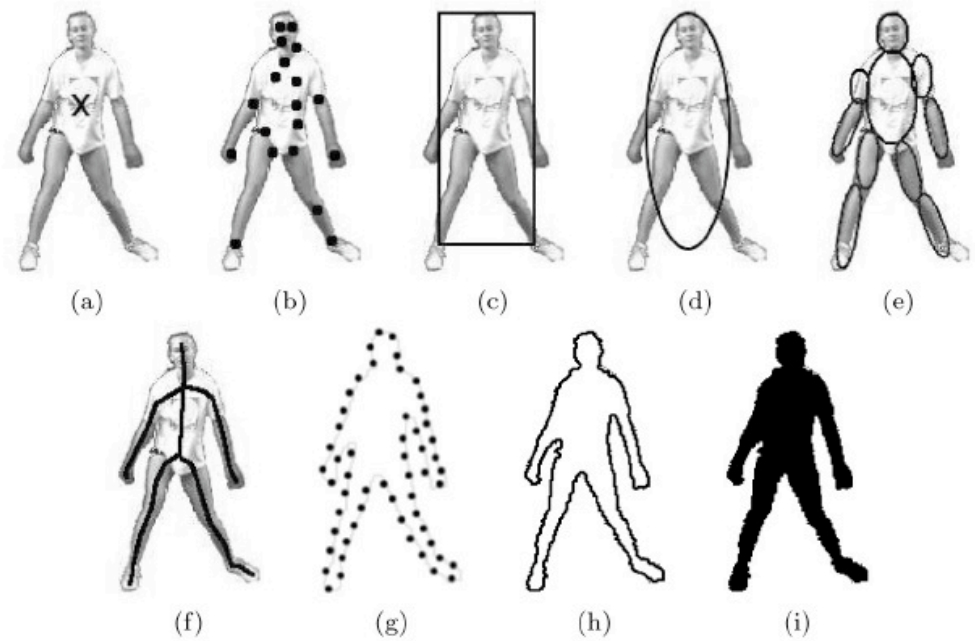

Figura 2.1 Formas de representação de objetos. (a) e (b) Pontos;(c) e (d) Forma Geométrica; (e) Forma articulada; (f) Esqueleto; (g) e (h) Contorno e (i) Silhueta. Fonte: (YILMAZ et al., 2006).

2.2 DETECÇÃO DOS OBJETOS

A detecção do objeto para o rastreamento é uma atividade crítica para seu controle. Em geral, a singularidade é uma propriedade desejável para que os objetos sejam facilmente distinguíveis na cena (YILMAZ et al., 2006). Esta detecção está intimamente relacionada com a

representação escolhida para determinado objeto. Por exemplo, na representação do contorno utiliza-se normalmente como base a detecção das bordas do objeto. Em geral, os algoritmos de rastreamento podem recorrer a uma combinação de características para conseguir monitorá-lo.

Cada método de rastreamento exige um mecanismo de detecção em cada quadro a partir do momento que o primeiro objeto apareça no vídeo.

Uma abordagem comum para a detecção de objetos é a utilização de informações em um único quadro, no entanto, alguns métodos de detecção de objetos fazem uso da informação temporal a partir de uma seqüência de quadros para reduzir o número de falsas detecções. Esta informação temporal é geralmente sob a forma de diferenciação de quadros, o que evidencia mudança em quadros consecutivos. Dados os objetos em um quadro, é tarefa do algoritmo realizar a correspondência deste em relação ao próximo quadro para caracterizar o rastreamento.

A detecção do objeto é portanto a escolha do método de segmentação da imagem. Alguns dos diferentes métodos de segmentação serão discutidos na Seção 3.

2.3 RASTREAMENTO (*TRACKING*)

O objetivo de um algoritmo de rastreamento é gerar a trajetória de um objeto ao longo do tempo a partir de sua posição em cada quadro do vídeo. Pode-se também fornecer a região completa que é ocupada pelo objeto na imagem em cada instante do tempo, ou alguma característica que seja relevante para a aplicação do rastreamento. As tarefas de detectar o objeto e estabelecer a correspondência entre as instâncias de objetos entre quadros pode ser realizada individualmente ou em conjunto (YILMAZ et al., 2006). Em uma correspondência individual, os objetos são obtidos por meio de um algoritmo de detecção de objetos a fim de que que o algoritmo realize a correspondência dos objetos com o quadro anterior. No caso de ser realizada em conjunto, a correspondência é estimada de forma interativa pela localização e informações do objeto em um conjunto de quadros anteriores.

Em qualquer abordagem de monitoramento, os objetos são representados com a forma e modelos de aparência. O modelo selecionado para representar a forma do objeto limita o tipo de movimento ou deformação que pode sofrer. Por exemplo, se um objeto é representado como um ponto, então apenas um modelo de translação pode ser usado. No caso de representação por uma forma geométrica modelos paramétricos como movimento afim ou transformações projetivas são adequados. Essas representações podem aproximar o movimento de objetos

rígidos na cena. Para um objeto não-rígido, como a mão humana, silhueta ou contorno são as representações mais descritivas e paramétricas para especificar o seu movimento.

Em vista disto, algumas das categorias de rastreamento mais utilizadas são listadas abaixo.

Rastreamento de pontos

Os objetos detectados em quadros consecutivos são representados por pontos, e a associação dos pontos é baseada no estado do objeto anterior, que pode incluir a posição do objeto e seu movimento. Essa abordagem requer um mecanismo externo para detectar objetos em cada quadro. Um exemplo de correspondência objeto é mostrado na Figura 2.3 (a).

A utilização do rastreamento dos pontos pode ser um problema complicado, especialmente na presença de oclusões, falsas detecções e quando ocorrem entradas e saídas dos objetos nas cenas.

Em geral, os métodos de correspondência ponto mais utilizados são os chamados métodos determinísticos. Estes métodos definem um custo de de associar a cada objeto no quadro $k-1$ um único objeto no quadro k usando um conjunto de restrições de movimento. A minimização do custo de correspondência é formulado como um problema de otimização combinatória. A solução pode ser interpretada como o problema de encontrar o caminho de mínimo custo em um grafo entre todas as associações possíveis dos objetos presentes em cada cena, como exemplifica a Figura 2.2. Um algoritmo muito utilizado para esta associação é chamado algoritmo Hungarian , no qual o custo de correspondência é geralmente definido usando a combinação das restrições abaixo (YILMAZ et al., 2006):

- **Proximidade**: assume que a posição do objeto não pode mudar muito de um quadro para outro (ver Figura 2.3 (a)).

- **Velocidade máxima:** define um limite superior de velocidade do objeto e que as correspondências ocorram em uma região circular ao redor do objeto (ver Figura 2.3 (b)).

- **Movimentos suaves (*smooth motion*):** assume que a direção da velocidade do objeto não muda drasticamente (ver Figura 2.3 (c)).

- **Movimento comuns:** assume que a velocidade de objetos contidos em uma pequena área deve ser semelhante (ver Figura 2.3 (d)). Essa restrição é adequada para objetos representados por vários pontos.

- **Rigidez:** assume que os objetos são rígidos, portanto, a distância entre dois pontos sobre o objeto real permanecerá inalterada (ver Figura 2.3 (e)).

- **Uniformidade Proximal (p*roximal uniformity*):** é a combinação das restrições de proximidade e de *smooth motion*.

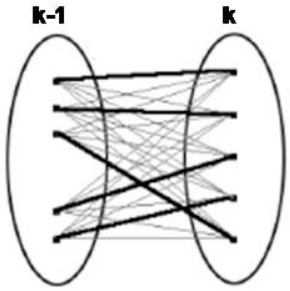

Figura 2.2 Solução para correspondência entre pontos em diferentes quadros utilizando conceito de grafos. Fonte: (YILMAZ et al., 2006).

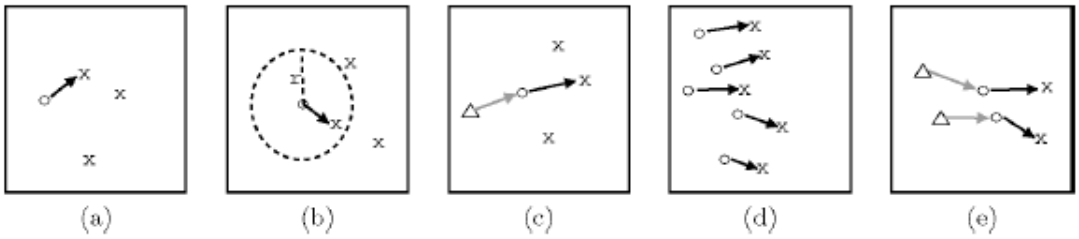

Figura 2.3 Critérios de restrição para definir custo em rastreamento de pontos. Fonte: (YILMAZ et al., 2006).

Rastreamento por *Kernel*

Kernel refere-se à forma do objeto e a aparência. Por exemplo, o *kernel* pode ser um modelo retangular ou uma forma elíptica ou um *template*. Os objetos são monitorados pelo cálculo do movimento do kernel em consecutivos quadros (Figura 2.4). Este movimento é geralmente sob a forma de uma transformação paramétrica, como translação, rotação e afim. Sua utilização é mais comum quando se considera um objeto rígido.

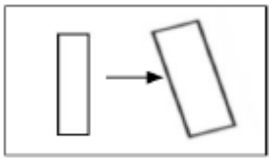

Figura 2.4 Representação do movimento em modelos de rastreamento por *Kernel*.

Rastreamento pela silhueta ou contorno

Os objetos podem ter formas complexas, por exemplo, a cabeça, as mãos e ombros portanto não podem ser descritos por formas geométricas simples. Para tanto, o método de rastreamento da silhueta fornece uma forma de descrição precisa para esses objetos. O

objetivo de um rastreador baseado em silhueta é encontrar o objeto em cada quadro por meio de um modelo de objeto gerado nos quadros anteriores.

A partir do modelo de objeto, sua silhueta é rastreada por correspondência de forma (*matching*) ou pela evolução de contorno. Exemplos de evolução do contorno são mostrados na Figura 2.5.

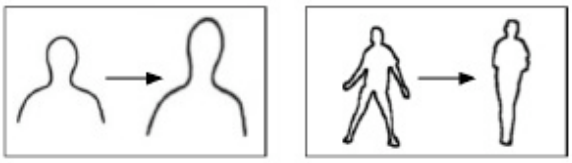

Figura 2.5 Exemplo de evolução de contorno em rastreamento. Fonte: (YILMAZ et al., 2006).

2.4 PROBLEMAS DE OCLUSÃO NO RASTREAMENTO

O problema da oclusão em um quadro pode ser classificado em três categorias: auto-oclusão, oclusão inter-objetos e oclusão com fundo da imagem.

A auto-oclusão ocorre quando uma parte do objeto oclui outra. Esta situação surge mais frequentemente em rastreamento de objetos articulados, como a mão humana em que os dedos podem ocluir entre eles ou com a palma da mão.

Oclusão inter-objeto ocorre quando dois objetos que estão sendo monitorados ocluem-se mutuamente.

Da mesma forma, a oclusão do fundo ocorre quando uma estrutura no fundo obstrui os objetos monitorados, como por exemplo, quando um objeto oclui com sua própria sombra em movimento. A oclusão parcial de um objeto por uma estrutura de cenário é difícil de detectar, pela dificuldade de diferenciar se o objeto mudou a sua forma ou se está obstruído.

A chance de ocorrer a oclusão pode ser reduzida por uma seleção adequada de posição da câmera. Por exemplo, se as câmeras são montadas acima da cena, ou seja, quando uma vista panorâmica está disponível, oclusões entre os objetos no chão, não ocorrem. No entanto, em câmeras de visão oblíqua é provável encontrar oclusões de múltiplos objetos e por isso necessitam de um mecanismo de tratamento de oclusão. A utilização de múltiplas câmeras para visualizar a mesma cena também pode ser usada para resolver este problema durante o rastreamento (YILMAZ et al., 2006).

2.5 INTERAÇÃO HUMANO-COMPUTADOR (IHC)

A Association for Computing Machinery (ACM), definiu a interação humano-computador como uma disciplina que diz respeito ao projeto, avaliação e implementação de sistemas de computador interativos para uso humano (CARVALHO, 2003).

Esta interface pode ser considerada um meio de comunicação, pois é tanto um meio para a interação quanto uma ferramente que oferece os instrumentos para esta comunicação.

A interação possui componentes de *software* e *hardware* (DE SOUZA et al., 1999).

Os componentes de *hardware* compreendem os dispositivos com os quais o usuário realiza as atividades motoras e perceptivas. Os mais tradicionais são a tela, o teclado e o *mouse*.

O *software* da interface implementa os processos computacionais necessários para controle dos dispositivos de hardware e para a interpretação dos comandos dos usuários.

O conceito de usabilidade de um sistema refere-se à qualidade da interação de sistemas. Para classificar um sistema em relação à sua usabilidade são considerados alguns fatores (DE SOUZA et al., 1999), dentre eles ressalta-se a facilidade de uso que avalia o esforço físico e cognitivo do usuário durante o processo de interação, medindo a velocidade com que é realizado e o número de erros cometidos durante a execução de uma determinada tarefa.

Em interfaces de interação tradicionais é necessário um dispositivo físico para traduzir os comandos dos usuários para o computador (TRUYENQUE, 2005), o que determinada a necessidade de um esforço físico maior que em um sistema que utiliza Visão Computacional. Assim, o desafio de usabilidade está no projeto de novas tecnologias que buscam explorar ao máximo as capacidades dos usuários na criação de ambientes de trabalho mais eficazes e produtivos (DE SOUZA et al., 1999).

As interfaces baseadas em Visão Computacional eliminam a necessidade de dispositivos físicos como intermediários e permitem uma interação mais natural e direta pela utlilzação de partes do corpo para controlar o computador, por exemplo, o dedo indicador da mão poderia ser utilizado para substituir o *mouse*.

A interação humano-computador utilizando visão computacional é caracterizada pela utilização de câmeras para capturar imagens, e técnicas de processamento de imagens e reconhecimento de padrões para o rastreamento dos objetos (RIBEIRO, 2006) e para interpretar um movimento ou gesto.

2.5.1 Utilização da mão para a interação humano-computador

Existem vários dispositivos que possibilitam que os usuários utilizem a mão para interagir com o computador. Desde os mais tradicionais, como teclado, *mouse*, *Joystick* e até outros mais sofisticados como *Power Glove, P5 VR Glove*, *Wii Remote.* Muitos desses dispositivos são populares e funcionam de maneira confiável e mais eficiente comparado aos sistemas de visão computacionais atuais, porém os avanços dos últimos anos, como por exemplo os recentes e *Gesture-cube* e *Gesture Tek*, demonstram que melhores resultados serão alcançados em um fututo próximo.

As Figuras de 2.6 e 2.7 exemplificam alguns sistemas para interação humano-computador que utilizam a mão com dispositivos para conseguir rastrear seu movimento. Por outro lado, as Figuras 2.8 e 2.9 mostram dois sistemas atuais de interação humano-computador aplicando tecnologia de visão computacional.

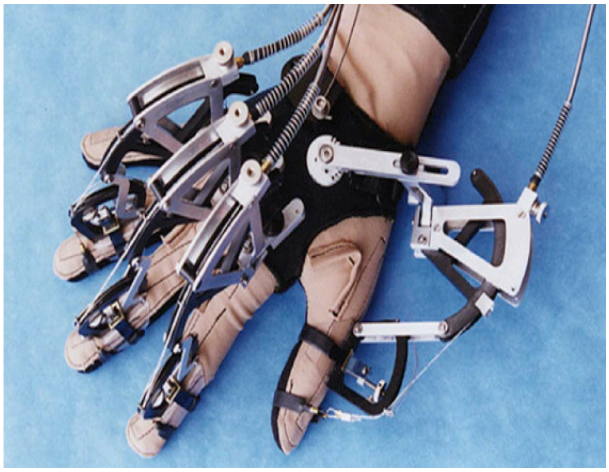

Figura 2.6 *Data glove*

Figura 2.7 *Wii Remote*

Figura 2.8 *Gesture-cube*

Figura 2.9 *Gesture Tek*

Muitas vantagens podem ser citadas nos sistemas que utilizam visão computacional. Primeiramente, em relação à versatilidade. Enquanto alguns dispositivos tem suas funções limitadas a seu uso com o PC (*personal computer*) e com *Games,* como o *mouse* e *Joystick*, a Visão Computacional possibilita uma ampla gama de aplicações para controle de monitores, notebooks, telefones celulares, televisores, projetores e aparelhos de som (*Gesture Cube*) apenas pela integração de uma câmera ao aparelho.

Outra vantagem importante está no fato de que o uso da câmera substitui a necessidade de interligação do sistema por cabos e elimina a necessidade de manipulação de aparelhos, o que possibilita ao usuário controlar máquinas de maneira livre, sendo muito útil para controlar máquinas em locais inseguros e perigosos à distância.

2.6 DIFICULDADES NO RASTREAMENTO DE OBJETOS

O problema de detecção de uma mão em movimento em um fundo constante pode parecer simples, mas se considerarmos o processo realizado em nosso cérebro, mesmo se as câmeras tivessem a mesma capacidade de captura de informação que a retina , os computadores não possuem o enorme poder de processamento em paralelo que possui o nosso cérebro (TRUYENQUE, 2005), por isso somente alguns processos básicos são implementados na maioria dos sistemas em tempo real baseados em Visão.

Uma outra dificuldade da Visão é a instabilidade causada por mudanças de iluminação, oclusões entre os objetos e ruídos nos equipamentos de captura.

As soluções destes problemas são, em sua maioria, limitadas a algumas aplicações, já que integram características intrínsicas do sistema para ficar mais robustas. Por estas razões é difícil construir um sistema de Visão Computacional de propósito geral que seja capaz de trabalhar com toda classe de objetos e em todos os ambientes e é preciso restringir o campo de ação e construir sistemas mais específicos (TRUYENQUE, 2005).

2.7 TRABALHOS RELACIONADOS

Fujii et al. (2002) utilizaram uma câmera de infra-vermelho e uma estrutura contendo um espelho inclinado para detectar o movimento em três dimensões dos dedos. A utilização do infra-vermelho foi baseada no fato de que com esta câmera as condições de iluminação e de cor do objeto não eram tão relevantes. Neste caso, o posicionamento dos dedos é dado pela

combinação das imagens da mão captadas de forma direta e refletida. A extração do fundo da imagem é realizada de maneira simples pela utilização de um limiar.

Letessier e Bérard (2004) propõe um algoritmo para rastreamento dos dedos da mão baseado em Segmentação por diferenciação da imagem (Image Differencing Segmentation) e Filtros de Rápida Rejeição (Fast Rejection Filters). Eles utilizam critérios geométricos para caracterizar as pontas dos dedos e a partir de um filtro de forma (*shape filtering*) conseguiram extrair a posição dos dedos. Cada ponta de dedo detectada em cada quadro foi associada a uma ponta do quadro anterior utilizando como custo a restrição de velocidade máxima, isto é, as pontas correspondentes em cada quadro não poderiam se distanciar de uma reginão circular ao redor do objeto com um raio máximo definido. Segundo os autores este algoritmo possuía a vantagem de ser insensível à orientação dos dedos e insensível a eventuais sombras existentes na imagem. Duca et al. (2007) propuseram um algoritmo de rastreamento em tempo real dos movimentos dos dedos da mão em três dimensões. O algoritmo proposto é baseado no uso de apenas uma *webcam* combinada a utilização de códigos para reconhecer a palma da mão e os dedos. Os dedos eram codificados por diferentes cores e com uma linha que dividia cada dedo e identificada o eixo de cada um, como mostra a Figura 2.10, assim cada dedo era reconhecido facilmente em cada quadro. Utilizando uma figura quadrada na palma da mão e em sua face oposta eles conseguiram reconhecer o movimento de rotação da mão pela variação da área do quadrado. Esta solução, segundo os autores, tem a vantagem de rápido processamento em relação a outros algoritmos em três dimensões, e de necessitar de recursos mínimos, porém fica dependente da utilização destes marcadores, não sendo uma solução .de grande usabilidade.

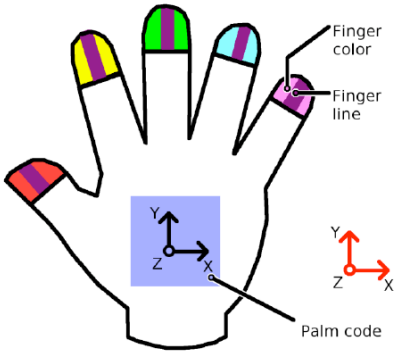

Figura 2.10 Modelo de codificação dos dedos para realização de rastreamento.*Fonte Duca et al. (2007)*

2.8 CONSIDERAÇÕES FINAIS

Este capítulo apresentou o conceito de rastreamento de objetos em cenas de vídeos focando-se no rastreamento de mão humana.

Os principais desafios no rastreamento da mão humana resumem-se na escolha de um método de detecção que consiga caracterizar de maneira eficiente um objeto articulado com forma complexa. Portanto, a forma recomendada para representar a mão é por meio de seu contorno, ou pela combinação da representação pelo contorno com alguma das outras formas de representação apresentada. Outra característica do rastreamento da mão é a necessidade de prever e tratar as oclusões que caracterizam um objeto articulado.

A eficiência da detecção da mão pode ser traduzida para a eficiência do método de segmentação utilizado. Por conseguinte, o capítulo 3 que se segue apresenta de forma mais detalhada os conceitos envolvidos na segmentação de imagens.

CAPÍTULO 3

3 SEGMENTAÇÃO E PÓS-PROCESSAMENTO DE IMAGENS

Segmentação é uma operação básica em imagem que equivale a encontrar regiões associadas com os objetos de interesse no domínio da imagem.

O nosso sistema visual executa esta tarefa de forma natural. Nós não temos nenhuma dificuldade para distinguir todos os tipos de objetos, quando olhamos para uma cena. No entanto, a segmentação é uma das operações mais desafiadoras em análise de imagem.

A obtenção de melhores resultados na análise de informações extraídas de uma imagem esta intimamente ligada à qualidade do tratamento feito sobre a fonte dos dados (SALDANHA, 2009). As técnicas de extração de informação utilizadas para o rastreamento e análise de imagens tem como passo inicial a realização de uma segmentação (GONZALEZ; WOODS, 1987).

O problema de segmentação é crucial na área de processamento de imagens e uma das tarefas mais difíceis dentro do processamento da imagem. Além de possibilitar a detecção de objetos, o processo de segmentação é importante para uma melhor visualização da imagem e uma quantificação dos elementos dentro da imagem (área, perímetro, volume, etc.).

Os objetos que compõem uma imagem possuem duas características básicas (SALDANHA, 2009):

1) Eles exibem alguma uniformidade interna em relação a uma propriedade da imagem;

2) Eles contrastam em relação à sua vizinhança.

Essas características são adotadas como referência para a elaboração de diversos algoritmos de segmentação.

A abordagem para a segmentação de imagens pode ser feita de baixo para cima (*bottom-up*) ou de cima para baixo *(top-down*).

A primeira toma por base princípios de continuidade, agrupando *pixels* da imagem de acordo com o grau de uniformidade dos níveis de cinza ou textura das regiões, bem como, a suavidade e a continuidade dos limites dos contornos.

Já a segunda, utiliza conhecimentos a priori sobre um objeto, tais como sua possível forma ou textura, para guiar a segmentação.

Não existe um modelo formal para a segmentação, o processo é essencialmente empírico e deverá se ajustar a diferentes tipos de imagem.

Esta etapa é, portanto, a mais difícil do processo e também a mais delicada porque todas as medidas serão realizadas sobre as regiões identificadas neste momento.

A maioria dos algoritmos são geralmente baseados em dois conceitos:

- Similaridade ou homogeneidade da região: a imagem deve ser dividida em regiões homogêneas com intensidade similar, cor, textura ou movimento;
- Descontinuidades ou transições: transições (por exemplo, pontos de extremidade) são detectadas nas imagens e usadas para estimar a borda do objeto.

Alguns dos métodos para segmentação que consideram estes conceitos serão apresentados nas seções subsequentes.

3.1 DETECÇÃO DE DESCONTINUIDADES

A descontinuidade em uma imagem pode ser:

- um ponto isolado;
- uma linha;
- a borda de uma objeto.

Essas feições sobressaem numa imagem, seja por possuir tons de cinza distintos da região na qual estão inseridas ou por assinalarem mudanças bruscas de tons de cinza entre regiões.

Neste contexto os algoritmos baseados em descontinuidades podem partir:

a) Da detecção de pontos: é a mais simples técnica de detecção. Um ponto terá uma mudança drástica do valor de cinza em relação aos seus vizinhos.
b) Da detecção de linhas: é o processo mais complicado, pois é necessário achar os pixels que são semelhantes e testá-los para verificar se são parte de uma linha comum.
c) Da detecção de bordas: é uma das técnicas básicas utilizadas pela visão humana no reconhecimento de objetos. É o processo de localização e realce dos *pixels* de borda, aumentado o contraste entre a borda e o fundo. Este processo verifica a variação dos valores de luminosidade de uma imagem.

Neste trabalho é dada ênfase apenas para os algoritmos de segmentação para detecção de bordas. Borda é o contorno entre um objeto e o fundo indicando o limite entre objetos sobrepostos. Entende-se por contorno uma linha fechada formada pelas bordas de um objeto.

Variações de intensidade complexas que ocorrem em uma região são geralmente chamadas de textura. Bordas são definidas como picos da magnitude do gradiente, ou seja, são variações abruptas que ocorrem ao longo de curvas baseadas nos valores do gradiente da imagem. Elas são regiões da imagem onde ocorre uma mudança de intensidade em um certo intervalo do espaço, em uma certa direção. Isto corresponde a regiões de alta derivada espacial, que contém alta frequência espacial.

Se uma borda é definida por uma mudança no nível de cinza, quando ocorre uma denscontinuidade na intensidade, ou quando o gradiente da imagem tem uma variação abrupta, um operador que é sensível a estas mudanças, operará como um detector de bordas.

Um operador de derivada faz exatamente esta função. Uma interpretação de uma derivada seria a taxa de mudança de uma função, e a taxa de mudança dos níveis de cinza em uma imagem é maior perto das bordas e menor em áreas constantes.

Se a partir dos valores da intensidade da imagem identificarmos os pontos onde a derivada é um ponto de máximo, nós teremos marcado suas bordas.

Dado que as imagens são em duas dimensões, é importante considerar mudanças nos níveis de cinza em muitas direções. Por esta razão, derivadas parciais das imagens são usadas com as respectivas direções X e Y.

Uma estimativa da direção atual da borda pode ser obtida usando as derivadas x e y como os componentes da direção ao longo dos eixos, e calcular o vetor soma. O operador envolvido é o gradiente, e se a imagem é vista como uma função de duas variáveis $I(x, y)$ então o gradiente é definido como na equação 1.

$$\nabla I(x, y) = \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \tag{1}$$

A maioria dos algoritmos de segmentação utilizam a convolução de uma máscara sobre os *pixels* da imagem para detecção de uma descontinuidade. Cada *pixel* e seus *pixels* vizinhos tem o valor do seu nível de cinza multiplicado por uma constante. A soma destes valores representa o novo valor de nível de cinza que representa a resposta daquele ponto.

O algoritmo mais conhecido para detecção de bordas é aquele que utiliza os operadores de Sobel e utiliza duas máscaras, para encontrar os gradientes vertical e horizontal das bordas. Estas máscaras são apresentadas pela equação 2.

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad e \quad Gy = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2}$$

Uma propriedade importante das bordas é que elas são menos sensíveis às mudanças de iluminação em relação às características de cor.

3.2 DETECÇÃO DE SIMILARIDADES

O fundamento para a detecção por similaridades é a observação do interior dos objetos e não as fronteiras que os delimitam. Para tanto, parte da idealização de que os *pixels* que compõe um objeto têm propriedades similares enquanto que *pixels* de objetos distintos têm propriedades distintas (SALDANHA, 2009).

As principais abordagens baseiam-se em:

Limiarização: A limiarização é a forma mais simples de segmentar imagem digitais, além disso, é uma das abordagens mais importantes para a segmentação (GONZALEZ; WOODS, 1987). A idéia que está por trás dessas técnicas é a que um objeto pode ser entendido como uma região formada por *pixels* contíguos que tenham em comum uma faixa de intensidades. Dessa forma, a limiarização usa a intensidade dos *pixels* para distinguir as regiões. Esta abordagem será tratada com mais detalhes na seção 3.2.1.

Crescimento de regiões: O princípio do funcionamento da técnica de crescimento por regiões é agrupar *pixels* ou sub-regiões em regiões maiores (GONZALEZ; WOODS, 1987). Seu início se dá com a adoção de um conjunto de *pixels* "sementes", a partir dos quais é realizado o crescimento das regiões por meio da inclusão de pixels vizinhos que tenham atributos similares, tais como, intensidade, textura, cor, entre outros. O crescimento ocorre interativamente até que cada *pixel* seja processado e sejam formadas diferentes regiões cujas fronteiras possam ser definidas por polígonos fechados, e os *pixels* internos a essas compartilhem de certa similaridade.

Apesar de ter funcionamento simples, o crescimento de regiões esconde algumas dificuldades relacionadas à sua fundamentação.

Uma dessas dificuldades diz respeito justamente ao início do processo, que é a escolha de "sementes".

Outra dificuldade está ligada à natureza da aplicação e ao tipo de imagem disponível, e diz respeito à definição do critério de similaridade a ser usado além de apresentar problemas na escolha do critério de parada.

3.2.1 Limiarização

Matematicamente, a operação de limiarização pode ser descrita como uma técnica de processamento de imagens na qual uma imagem de entrada $I(x, y)$ de N níveis de cinza produz na saída uma imagem $l(x, y)$, chamada de imagem limiarizada, cujo número de níveis de cinza é menor que N. Normalmente, $l(x, y)$ apresenta dois níveis de cinza, sendo portanto uma imagem binarizada dada pela equação 3, onde os pixels rotulados com "1" correspondem aos objetos e os pixels rotulados com "0" correspondem ao fundo e T é um valor de tom de cinza predefinido denominado limiar.

Neste caso tem-se a limiarização simples.

$$l(x, y) = \begin{cases} 1, & I(x, y) \leq T \\ 0, & I(x, y) > T \end{cases} \tag{3}$$

A classificação é muitas vezes baseada no histograma de uma propriedade (em geral, o nível de cinza). Dado que o número de pixels numa imagem é geralmente muito elevado, pode-se considerar o histograma como uma boa aproximação à densidade de probabilidade da propriedade que ele representa.

Um caso simples de ser entendido é o de um histograma bimodal (PADILHA, 1998). Nessa situação o histograma apresenta dois picos separados por um vale entre eles, como mostra a Figura 3.1 (b).

Esses picos podem representar duas regiões distintas, como por exemplo, um fundo e um objeto. A Figura 3.1(c) exemplifica a binarização da imagem presente na Figura 3.1(a) por um limiar escolhido em seu histograma.

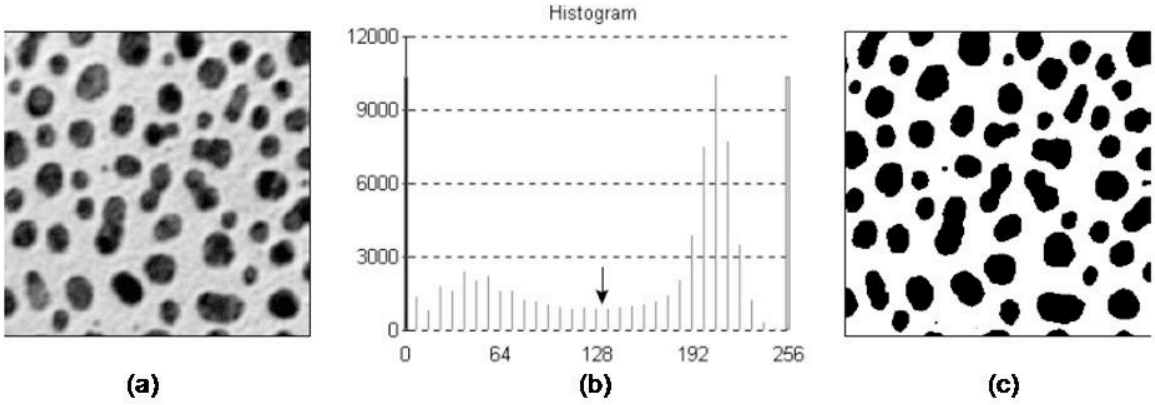

Figura 3.1 Exemplo de segmentação de imagem em escala de cinza por limiarização. *Fonte: (PADILHA, 1998).*

A localização do vale entre picos de histogramas bimodais não é, geralmente, muito fácil devido ao carácter discreto dos níveis do histograma e à presença de ruído nas imagens -

próximo do vale podem existir diversos mínimos locais, não sendo trivial encontrar automaticamente o melhor limiar.

Para resolver este problema, é frequentemente útil proceder a uma suavização da imagem antes da determinação do histograma (por exemplo, usando filtros de média), ou a uma pós-filtragem do histograma (interpretado como um "sinal" unidimensional).

Quando cada pixel é caracterizado por um conjunto de medidas relativas a N distintas bandas espectrais, por exemplo em imagens coloridas RGB, a segmentação pode efetuar-se no espaço N-dimensional.

Por exemplo, quando N é igual a 2, a separação entre classes pode efetuar-se por meio de segmentos de reta, como se procura ilustrar no diagrama de dispersão da Figura 3.2. Observe que se o histograma tivesse considerado apenas uma banda não apresentaria tão claramente três regiões que pudesse representar três classes diferentes.

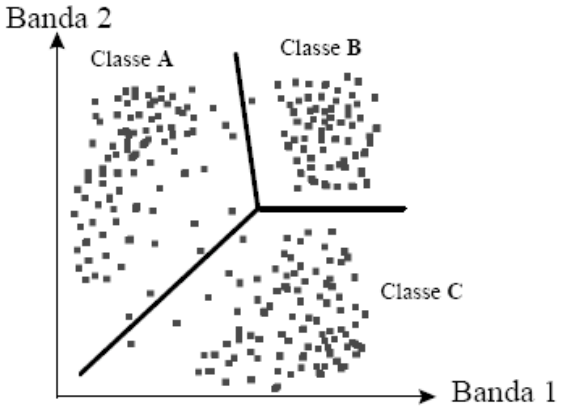

Figura 3.2 Exemplo de limiarização em duas dimensões. *Fonte: (PADILHA, 1998)*

Este conceito é utilizado para segmentação através de limiares em espaço de cor, e será aprofundado na seção que se segue.

3.3 SEGMENTAÇÃO DE IMAGENS COLORIDAS

Segmentação de imagens coloridas é um processo pelo qual se extraem, do domínio da imagem, uma ou mais regiões conectadas que satisfaçam o critério de uniformidade (homogeneidade), o qual é baseado em características derivadas de componentes do espectro.

Esses componentes são definidos em um modelo de espaço de cores escolhido. O processo de segmentação pode ser melhorado através de alguns conhecimentos adicionais sobre os objetos em cena, tais como as suas propriedades ópticas e geométricas.

Uma característica importante deste método de segmentação é a definição de região. Em uma classificação baseada em pixel a região é definida como um componente conectado de um grupo de pixels, especificado por uma função de pertinência de classes definida num espaço de cores. Por exemplo:

- Cor do pixel está num determinado quadrante definido por um plano.;
- Cor do pixel está no poliedro dado;

Portanto, é necessário primeiramente escolher um espaço de cor a ser considerado para a segmentação. Alguns espaços de cores são apresentados na seção seguinte.

3.4 ESPAÇOS DE COR

O objetivo de um modelo de cores é facilitar a especificação das cores em uma forma normalizada e aceita genericamente. Alguns dos espaços de cores mais conhecidos são apresentados nesta seção.

3.4.1 RGB (*Red Green Blue*)

No sistema RGB, cada cor é definida pela quantidade de vermelho (*Red* em inglês), verde (*Green* em inglês) e azul (*Blue* em inglês) que a compõem.

Por conveniência, a maioria dos arquivos digitais atuais usam um byte para especificar estas quantidades (255 possiblidades). O número 0 indica ausência de intensidade e o número 255 indica intensidade máxima.

Neste contexto, cada cor no sistema RGB é identificado por uma tripla ordenada (R, G, B) de números inteiros, com $0 \leq R \leq 255$, $0 \leq G \leq 255$ e $0 \leq B \leq 255$. Sendo assim, podemos associar cada cor do sistema RGB com pontos com coordenadas inteiras de um cubo com aresta de tamanho 255. Este modelo é apresentado na Figura 3.3.

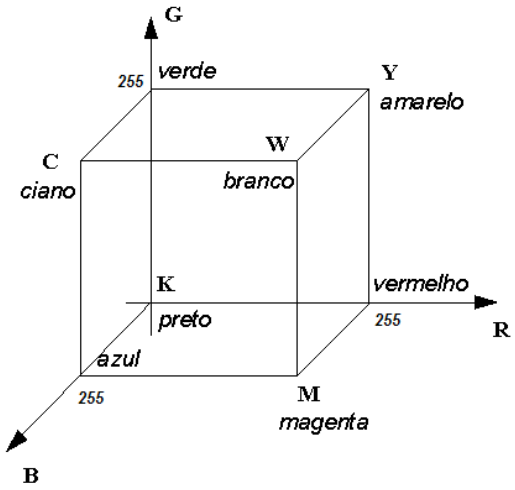

Figura 3.3 Modelo do cubo RGB.

Cada pixel é representado por três bytes, um para cada componente de cor: vermelho, verde e azul. Esses bytes são todos empacotados em um inteiro. Uma vez que cada componente é armazenado como um byte, cada um pode representar 256 diferentes intensidades da cor correspondente.

O modelo RGB é amplamente usado na computação porém possui uma desvantagem em relação à definição de cores baseando-se no sistema de percepção visual humano, pois não garante que cores próximas dentro do espaço RGB sejam próximas em termos de percepção visual (RIBEIRO, 2006). Para contornar este tipo de problema surgiram outros espaços de cores como veremos nas seções a seguir.

3.4.2 RGB normalizado

O RGB normalizado é uma representação obtida através dos valores RGB após um procedimento de normalização. Os valores r, g e b são determinados respectivamente pelas equações 4, 5 e 6.

$$r = \frac{R}{R+G+B} \tag{4}$$

$$g = \frac{G}{R+G+B} \tag{5}$$

$$b = \frac{B}{R+G+B} \tag{6}$$

Como a soma dos três componentes normalizados é conhecida ($r + g + b = 1$), um dos três componentes pode ser omitido com o intuito de reduzir a dimensão espacia, já que não mantém nenhuma informação significativa na resolução deste sistema, e l.

Os componentes restantes são chamados de “cores puras”, pois a normalização diminui a dependência dos componentes r e g ao brilho do espaço de cores RGB.

Uma propriedade notável deste espaço de cores é que, quando ignoramos a luz do ambiente em superfícies opacas, o RGB normalizado é invariável às mudanças de orientação das fontes de luz na superfície (RIBEIRO, 2006). Esta característica em conjunto com a simplicidade na transformação ajuda este espaço de cores a ganhar popularidade entre os pesquisadores.

3.4.3 HSV (*Hue Saturation Vaue*)

Ao contrário do RGB, o espaço o HSV (*Hue Saturation Value* - Matiz Saturação e Brilho) é perceptualmente uniforme. Neste espaço, as cores são formadas também por três valores, a saber:

- **Matiz**: Verifica o tipo de cor, podendo ser vermelho, amarelo ou azul. Atinge valores de 0 a 360, mas para algumas aplicações, esse valor é normalizado de 0 a 100%.
- **Saturação**: Também chamado de "pureza". Quanto menor esse valor, a imagem aparecerá com mais tons de cinza. Quanto maior o valor, mais "pura" é a imagem. Atinge valores de 0 a 100%.
- **Brilho**: Define o brilho da cor. Atinge valores de 0 a 100%.

O espaço HSV é caracterizado por ser uma transformação não-linear do sistema de cores RGB. A transformação do espaço RGB para o espaço de cores HSV é efetuada pelas equações 7, 8 e 9.

$$H = \arccos\left(\frac{\frac{1}{2}\cdot\left((R-G)+(R-B)\right)}{\sqrt{\left((R-G)^2+(R-B)\cdot(G-B)\right)}}\right) \tag{7}$$

$$S = 1 - 3\cdot\frac{\min(R,G,B)}{R+G+B} \tag{8}$$

$$V = \frac{1}{3}\cdot(R+G+B) \tag{9}$$

O espaço de cores HSV trabalha-se com a representação de um hexacone (um cone com base hexagonal), no qual um ângulo H com respeito ao eixo horizontal determina a matiz de cor desejada, a distância perpendicular do centro até a borda determina a saturação S e a distância vertical, determina a luminosidade V, como mostra a Figura 3.4.

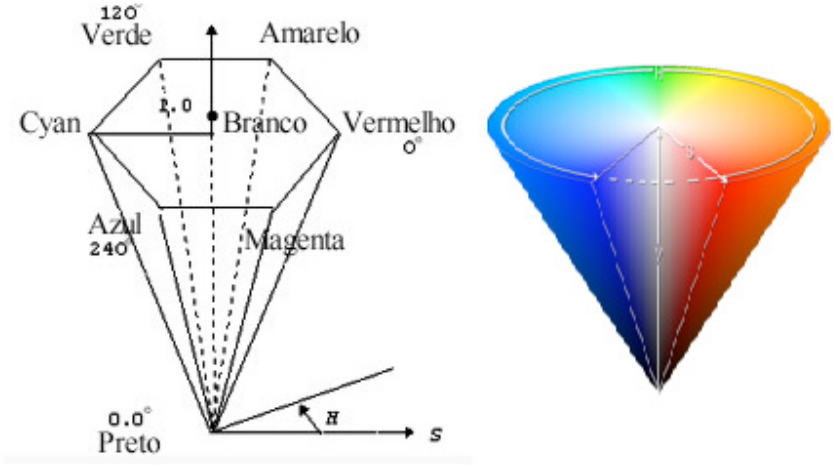

Figura 3.4 Modelo de hexacone HSV.

3.4.4 YCbCr

YCbCr é um espaço de cor usado dentro de sistemas de vídeo e fotografia digital. A cor é representada por brilho ou luma, que é a luminância (Y) calculada de RGB não linear, criados com pesos na soma dos valores de RGB, e diferença de duas cores Cr e Cb que são formados pela subtração de luma dos componentes de vermelho (R–Y) e azul (B–Y). Este espaço pode ser equacionado a partir do modelo RGB pelas equaçoes 10, 11 E 12. YCbCr não é espaço de cor absoluto, é uma maneira de codificar a informação RGB.

$$Y = 0{,}299R + 0{,}587G + 0{,}114B \tag{10}$$

$$Cr = (R - Y)0{,}713 + 128 \tag{11}$$

$$Cb = (B - Y)0{,}564 + 128 \tag{12}$$

A figura 3.5 mostra graficamente o espaço de cores YCbCr. É possível observar como a crominância se espalha pelos eixos Cb e Cr e a variação da luminância pelo eixo Y . Para Y mínimo a cor mostrada é preta para qualquer crominância e, para Y máximo a cor mostrada é branca para qualquer crominância.

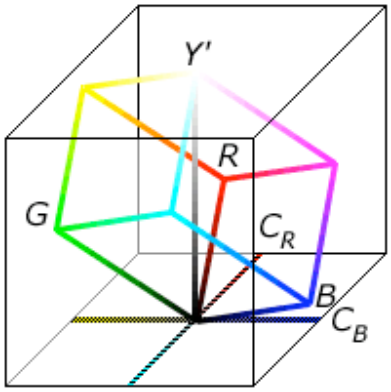

Figura 3.5 Modelo do espaço de cor YCbCr.

3.5 SEGMENTAÇÃO DE PELE

Em uma primeira análise, a seleção do espaço de cores parece ser crucial para as técnicas de detecção de pele que utilizam a cor como elemento principal.

Uma questão importante é escolher um espaço de cor adequado para detecção de pele, e a discussão que se tem é se há um espaço de cores ótimo para classificação de pele.

Muitos artigos relacionados à detecção e classificação de pele não apresentam uma justificativa para a escolha do espaço de cores. Isso provavelmente ocorre pela possibilidade de se obter, em quase todo espaço de cores, resultados aceitáveis na detecção de pele (quando uma base de dados limitada é utilizada).

A detecção de pele por cor é caracterizada por ser sensível a vários fatores como (KAKUMANU et al., 2007):

1. Iluminação: As mudanças de fontes de iluminação e tipos de iluminação (interna, externa, com sombras, etc) produz mudanças na cor da pele na imagem. Este problema é chamado como problema da constância da cor e é o mais importante problema nos sistemas atuais de detecção de pele que degrada seriamente sua performance.
2. Características das câmeras: mesmo sobre uma mesma condição de iluminação a distribuição da cor de pele de uma pessoa difere de uma câmera para outra dependendo das características de seu sensor e de sua sensibilidade.
3. Etnia: A cor da pele varia em pessoas de diferentes etnias e entre pessoas de diferentes regiões, por exemplo pessoas provenientes da Ásia, Africa, Caucasianos e Latinos possuem variações nas gradações de cores.
4. Características individuais: A idade da pessoa, sexo e a parte do corpo a ser detectada possue características individuais que afetam a aparência da cor da pele.

5. Outros fatores: Diferentes fatores podem influir na aparência da pele e dificultar a detecção de cor de pele, como por exemplo a cor do fundo da imagem, sombras presentes e o movimento na cena.

3.6 SUBTRAÇÃO DE BACKGROUND

A Figura 3.6 mostra a evolução de um pixel de cor em uma sequência de imagens em certo intervalo de tempo. Na maioria das vezes é possível representar esta intensidade $I(x, y, t)$ como a soma da intensidade média (ou a mediana) $\bar{I}(x, y)$ mais um fator ruidoso gaussiano $\omega(x, y, t)$ onde este ruído representa o movimento na cena. Assim a imagem é dada pela equação 13.

A imagem dada pela média ou pela mediana $\bar{I}(x, y)$ pode ser computada por um método robusto de uma sequência de vídeo. Esta imagem é chamada de imagem de fundo (*background*), isto é possível pois a mediana é insensível.

$$I(x, y, t) = \bar{I}(x, y) + \omega(x, y, t) \tag{13}$$

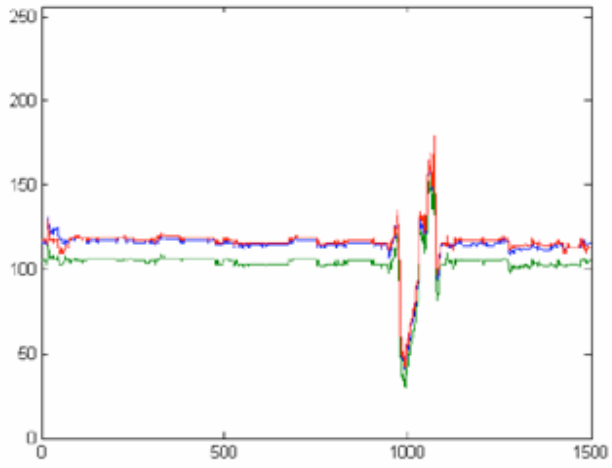

Figura 3.6 Evolução de um pixel em uma sequência de imagens.

Utilizando este modelo pode-se detectar movimentos na cena a partir da comparação do erro absoluto com um valor de limiar, como mostrado na equação 14.

$$\left| I(x, y, t) - \bar{I}(x, y) \right| > T \tag{14}$$

Este método é conhecido como método de subtração de fundo (*background subtration*) e sua uilização é muito popular para operações de rastreamento em tempo real por ser simples

e rápido. No entanto, tem dois grandes inconvenientes. Ele não pode lidar com ruídos intermitentes (por exemplo, mudanças de iluminação) e pode gerar objetos fantasmas.

Modelos mais sofisticados são usados para lidar com este problema. Por exemplo, as intensidades de pixel são muitas vezes modeladas por misturas de gaussianas que são capazes de representar várias distribuições modais. Isto é muito importante para representar os padrões de movimento mais complexos como os gerados pelo movimento no fundo da imagem. Os parâmetros de mistura são freqüentemente atualizados durante a operação do sistema para lidar com as mudanças de iluminação.

Por outro lado, a classificação independente dos pixels, pela equação 14, muitas vezes leva a pequenos grupos de pixels ativos que não são interpretados como objetos. Portanto, a imagem binária produzida pelo detector de movimento muitas vezes é processada por filtros morfológicos para remover pequenas regiões e preencher os buracos. Normalmente, estes filtros computam todos os componentes conectados e eliminam pequenas regiões.

3.7 FILTROS MORFOLÓGICOS

O estudo morfológico concentra-se na estrutura geométrica das imagens. A morfologia pode ser aplicada em diversas áreas de processamento de imagens, como realce, filtragem, segmentação, esqueletização e outras afins.

Esta teoria diz que é possível fazer transformações entre reticulados completos, os quais são chamados de operadores morfológicos.

Na morfologia matemática existem duas classes básicas de operadores: dilatação e erosão, chamadas de operadores elementares.

A base da morfologia consiste em extrair de uma imagem desconhecida a sua geometria através da utilização da transformação de uma outra imagem completamente definida, ou seja, consiste em extrair as informações relativas à geometria e à topologia de um conjunto desconhecido (no caso uma imagem) pela transformação através de outro conjunto bem-definido, chamado elemento estruturante. Com isso torna importante ao contexto a utilização da Teoria dos Conjuntos, pois esta é a base utilizada na morfologia.

Desta forma, é com esta teoria que será descrita e apresentada uma imagem. Por exemplo a definição de um vetor bidimensional onde será exposta as coordenadas (x, y) para sua representação gráfica.

3.7.1 Dilatação Binária

A dilatação é uma transformação morfológica que combina dois conjuntos usando adição vetorial. Seu símbolo é $\oplus$. Como o nome diz, o resultado será uma imagem "engordada".

A dilatação de um conjunto A por um conjunto B é definida pela equação 15. Onde A representa a imagem sendo operada e B é um segundo conjunto onde é chamado elemento estruturante e sua composição define a natureza especifica da dilatação, sendo assim a dilatação expande uma imagem. Esta pode ser representada também pela união entre esses dois conjuntos.

$$A \oplus B = \left(c \mid c = a + b, a \in A, b \in B\right) \tag{15}$$

Um exemplo da operação de dilatação pode ser visto na Figura 3.7.

Figura 3.7 Representação da operação morfológica de dilatação.

3.7.2 Erosão Binária

A erosão basicamente encolhe uma imagem e pode ser vista como uma transformação morfológica que combina dois conjuntos usando vetores de subtração. Ela é expressa como a interseção de A e B e seu símbolo é dado por Θ.

A erosão da imagem A pelo elemento estruturante B pode ser definida a partir da equação 16 ou 17.

$$A \Theta B = \left(x \mid x + b \in A, \quad para \quad todo \quad b \in B \right) \tag{16}$$

$$A \Theta B = \left(c \mid B_c \supseteq A\right) \tag{17}$$

Assim define-se que a erosão é o conjunto de todos os pixels, em o elemento estruturante B transladado por c está contido no conjunto de pixel em A.

Pode-se visualizar a erosão através do exemplo dado pela Figura 3.8. Assim, verifica-se que cada elemento de B é um elemento de A ou seja B esta contido em A

Figura 3.8 Representação da operação morfológica de erosão

3.7.3 Abertura

A abertura em geral suaviza o contorno de uma imagem, quebra estreitos e elimina proeminências delgadas, a operação de abertura é usada também para remover ruídos da imagem. A abertura de um conjunto A por elemento estruturante B e denotado $A \circ B$ e definida como dado na equação 18.

$$A \circ B = (A \Theta B) \oplus B \tag{18}$$

A aplicação de uma erosão imediatamente seguida de uma dilatação usando o mesmo elemento estruturante é uma operação de abertura, ela tende a abrir pequenos vazios ou espaços entre objetos próximos numa imagem. Em outras palavras, uma abertura é uma erosão seguida de uma dilatação usando um mesmo elemento estruturante.

Vale relembrar que a erosão acha todos os lugares onde o ajuste do elemento estruturante está dentro da imagem, mas isto somente marca esta posição à origem de um elemento.

A Figura 3.9 exemplifica a operação de abertura e mostra que as “sujeiras” ao redor de A foram eliminadas.

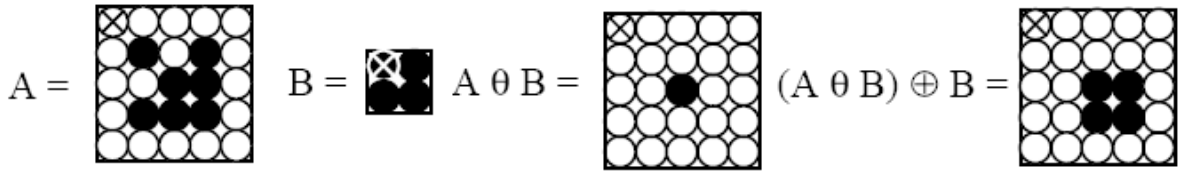

Figura 3.9 Representação da operação morfológica de abertura.

3.7.4 Fechamento

O fechamento funde pequenas quebras, alarga golfos estreitos e elimina pequenos orifícios. Se uma abertura cria pequenos vazios na imagem, um fechamento irá preencher ou fechar os vazios. Estas operações podem remover muitos dos pixels brancos com ruídos, ou seja basicamente ele é igual à abertura só que primeiramente é feita a dilatação seguida da erosão. Assim o fechamento é definido pela equação 19.

$$A \bullet B = (A \oplus B) \Theta B \tag{19}$$

Em outras palavras o fechamento trabalha de um modo oposto ao método abertura, já que ela remove todos os pixels onde o ajuste do elemento estruturante não esta dentro da imagem, e o fechamento enche todos os lugares onde o elemento estruturante não se iria ajustar na imagem. Entretanto, operações inversas, abertura e fechamento não irão estabelecer a imagem original.

A Figura 3.10 exemplifica esta operação. Observe que os “furos” existentes na imagem A foram eliminados.

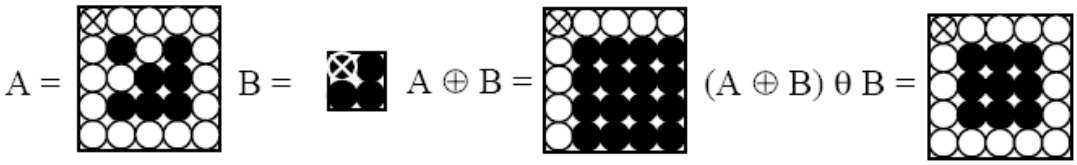

Figura 3.10 Representação da operação morfológica de fechamento.

3.8 CONSIDERAÇÕES FINAIS

Este capítulo focou-se no conceito de segmentação de imagem, apresentando alguns algoritmos popularmente utilizados que serviram como base para o desenvolvimento discutido no capítulo 4.

A segmentação da imagem pode considerar o critério de detecção de descontinuidades ou o critério de detecção de similaridades, o que caracteriza uma abordagem *bottom-up* ou *top-down*.

As teorias de segmentações por cor de pele e por subtração de fundo da imagem foram apresentadas demostrando a possibilidade de utilizar estas informações de cor e movimento para detectar uma imagem da mão humana.

A classificação independente dos pixels dada pela segmentação por limiar de cor e por subtração de background muitas vezes resulta em uma imagem contendo, além da mão segmentada, pontos ou *blobs* que podem dificultar na detecção da mão (objeto de interesse).

Portanto, a imagem binária produzida pela segmentação é processada por filtros morfológicos para remover pequenas regiões e preencher os buracos.

CAPÍTULO 4

4 MATERIAIS E MÉTODOS

Este trabalho propõe um sistema de visão computacional para reconhecimento e rastreamento de gestos dos dedos de uma mão humana à partir de imagens captadas por uma webcam de baixa resolução para ser utilizado como base em sistemas simples (em relação às ferramentas utilizadas) de interação humano-computador. Neste contexto, o trabalho contempla um problema de visão computacional e sua solução desde a aquisição de imagens, segmentação, reconhecimento, rastreamento de gestos até a interpretação dos resultados.

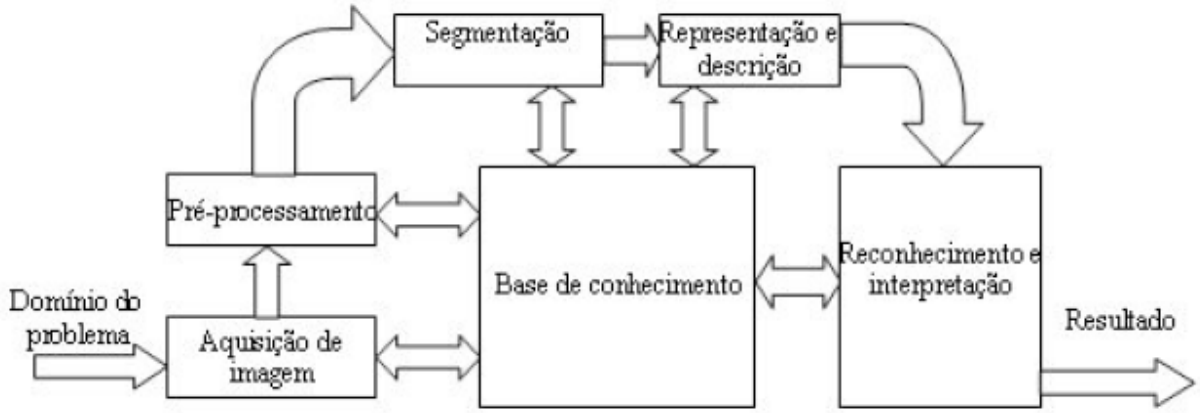

Figura 4.1 Diagrama de blocos de um sistema de visão computacional. *Fonte: Geisler (2006).*

Além de propor uma solução baseada no diagrama de blocos básico de um sistema de visão computacional, como visto na Figura 4.1, este projeto possui uma análise mais detalhada da fase de segmentação da imagem, contemplando um estudo comparativo entre vários métodos de segmentação com o objetivo de se escolher o método com melhor desempenho para o sistema em questão.

Portanto, o projeto foi dividido em duas fases, uma contendo o estudo comparativo dos métodos de segmentação, que será tratado na seção 4.2 e outra contendo a solução para rastreamento de gestos da mão, discutido em 4.3.

4.1 CONSIDERAÇÕES SOBRE O SISTEMA

O sistema desenvolvido utiliza uma *webcam* ligada a um PC para aquisição de imagens e tratamento dessas em tempo real com o intuito de rastrear o movimento de abrir e fechar a mão, o que significa abaixar e levantar os dedos.

O número de objetos presentes em cada quadro capturado é limitado a um, sendo este uma mão direita. O movimento da mão fica limitado à translação em três direções (nas duas dimensões do plano da imagem e a distância da mão à *webcam* sendo o terceiro eixo), os movimentos previstos da mão podem ser vistos na Figura 4.2. Nos movimentos dos dedos foram considerados os dois graus de liberdade, como mostrado na Figura 4.3. A *webcam* fica fixa e a mão fica a uma distância máxima de um metro em relação à lente da câmera, limitada por um painel de cor preta com o intuito de criar um *background* simples.

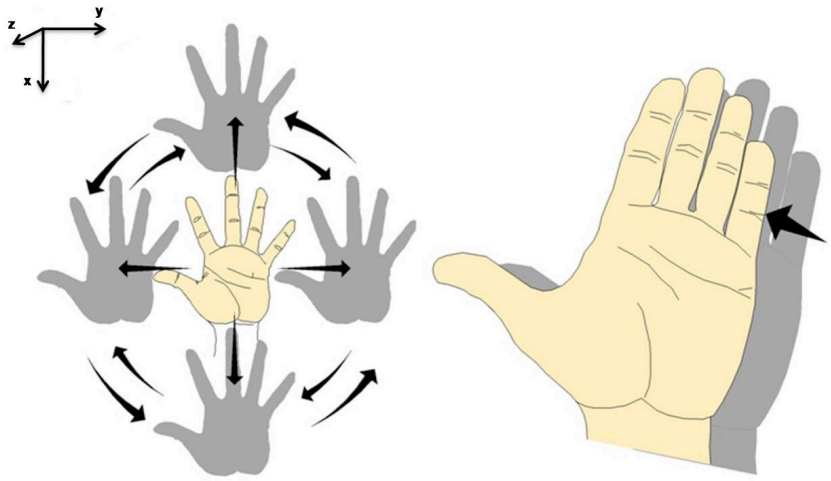

Figura 4.2 Movimento previsto da mão no sistema proposto de rastreamento.

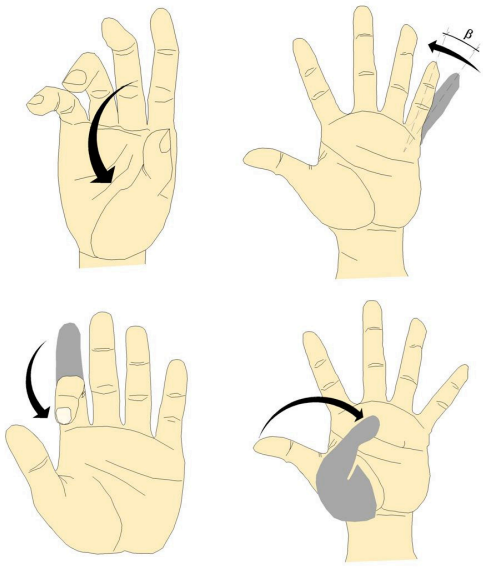

Figura 4.3 Movimento previsto dos dedos no sistema proposto de rastreamento**.**

Em relação à iluminação ambiente, foram considerados os casos de um ambiente fechado mas não isolado da iluminação externa, por exemplo um ambiente de laboratório com ou sem uma fonte de luz focada no objeto de interesse.

O sistema operacional e a configuração do PC ficam limitados aos que suportem a utilização do *software* MATLAB®, assim como as fases desde aquisição de imagem pela *webcam* até o processamento e interpretação de resultados ficam dependentes da utilização de ferramentas deste *software*. O esquema de configuração física proposta pode ser visto na Figura 4.4.

Figura 4.4 Esquema de configuração física proposta para o rastreamento do movimento dos dedos. *Fonte: (GEJGUS et al., 2004).*

4.2 ANÁLISE DA SEGMENTAÇÃO

Esta primeira fase do projeto foca-se em avaliar métodos de segmentação de imagem a partir de aspectos considerados relevantes para um bom desempenho dessa segmentação no resultado final do trabalho, isto é, que respondam de maneira eficiente e gerem parâmetros confiáveis a serem utilizados na próxima fase do projeto.

A análise quantitativa da performance de métodos de segmentação que melhor responda para determinado problema é uma estratégia utilizada por muitos autores no processo de visão computacional.

Milsztajn e Geus (2001) utilizaram campos aleatórios de Markov e Algoritmos Genéticos para segmentar imagens do cérebro humano e classificar anomalias, e para a validação dos resultados da segmentação realizaram a comparação visual das imagem segmentadas automaticamente, pela utilização do algoritmo proposto, e manualmente, no qual utilizaram um limiar (*threshold*) escolhido empiricamente.

A comparação deu-se a partir do grau de similaridade entre as duas segmentações, utilizando o coeficiente de Jaccard (MILSZTAJN, 2001) que mede a similiaridade de uma imagem à imagem de referência.

Sawangsri et al. (2005) propuseram um algoritmo para segmentar em cor de pele da face humana e compararam o resultado em relação à segmentação por limiares nos espaços de cores HSV, RGB e YCbCr.

Um trabalho muito completo de comparação de várias técnicas de segmentação de pele humana, utilizando informação de cor, foi realizado por Kakumanu et al. (2007) que analisaram a resposta em pessoas com diferentes cores de pele, assim como variadas condições de iluminação e diferentes tipos fundo (*backgrounds*), sendo que, para a validação, utilizaram vários Bancos de Dados de imagens contendo pixels de pele e de não pele, um artifício muito comum em artigos de análise de segmentação.

Vários são os Bancos de Dados disponíveis na internet, e estes contém um conjunto de centenas ou milhares de imagens em escala de cinza ou coloridas de peles de homens e mulheres de diferentes raças.

De forma parecida Vezhnevets et al. (2003) compararam a exatidão de alguns algoritmos de segmentação de face humana nos espaços de cores RGB, YCbCr, HSV e pela combinação destes três métodos, a validação quantitativa foi realizada a partir das taxas de detecção, chamado DR (*Detection Rate*), e a partir de falsos alarmes, chamados FAR (*False Alarm Rate*) (SAWANGSRI, 2005).

No presente trabalho, foram selecionados alguns algoritmos de segmentação em cor de pele nos espaçoes de cores RGB, HSV, YCbCr, RGB normalizado e também utilizando subtração de fundo simples. Estes serão analisados de forma quantitativa pela utilização dos parâmetros de similiaridade, taxa de falsos alarmes e taxa de detecção para que seja possível selecionar o que responda de maneira eficiente ao problema proposto de segmentação de uma mão humana.

Um bom classificador deve ser eficiente na classificação de diferentes tipos de pele e variadas condições de iluminação.

Para a análise da segmentação foram obtidos alguns vídeos em diferentes situações de iluminação e com diferentes fundos, contendo mão feminina e masculina como único objeto da cena.

A partir desses vídeos, foram selecionadas algumas imagens com diferentes posições da mão que remetessem ao seu movimento de translação e ao ato de abrir e fechar os dedos.

Essas imagens foram definidas como um banco de dados de imagens de referência para que os valores dos parâmetros considerados para a comparação fossem encontrados a partir delas.

Para que fosse possível encontrar os valores desses parâmetros foi necessário definir também um método de segmentação manual, por isso utilizou-se as imagens de referência para criar um outro banco de dados, contendo imagens correspondentes encontradas por esse método de segmentação manual, que será apresentado posteriormente.

Por essa análise dos diferentes algoritmos foi possível especificar o que seria utlizado na segunda fase do projeto, levando em consideração os valores de similiaridade, taxa de falsos alarmes, taxa de detecção e também uma análise visual das imagens para auxiliar na decisão.

4.2.1 Banco de dados das imagens

Na fase de aquisição de imagens procurou-se selecionar imagens de maneira cuidadosa, já que esta decisão influenciará diretamente no desempenho final do sistema.

Com o objetivo de comparar o desempenho de diferentes métodos de segmentação de imagem, foram realizados alguns vídeos para formar um banco de dados de imagens (quadros), as quais foram segmentadas e analisadas.

Estes vídeos foram gravados simulando algumas situações de ambientes internos habituais, para analisar a resposta de cada segmentação à variação dos parâmetros que influem na decisão de cada pixel ser objeto ou não.

No caso de segmentação utilizando a cor da pele, o resultado da iluminação pode influenciar negativamente no desempenho da segmentação, por exemplo, iluminação por lâmpadas coloridas podem agir como um filtro na imagem adquirida pela câmera (GOMEZ; M. SANCHEZ, 2002).

A variação seja pela influência da iluminação externa ao ambiente (por exemplo, vinda de uma janela), sombras geradas pelo objeto de interesse (que ocorrem com freqüência em imagens com o objeto à frente de fundos claros e dependendo do posicionamento da fonte de iluminação) e diferentes níveis de iluminação mostram-se como os principais e mais importantes problemas no processo de detecção de pele e degradam seriamente a performance de um sistema de segmentação.

Outra característica que se relevou na realização dos vídeos foi a escolha de alguns fundos diferentes, parâmetro que influencia muito no desempenho de um método de segmentação em cor de pele, já que objetos pertencentes ao fundo da imagem (*background*) com cores próximas à da pele podem ser facilmente confundidos com a mão.

Os vídeos selecionados continham as seguintes situações:

Vídeo 1: Ambiente interno com iluminação ambiente (superior e influência de iluminação externa) e Fundo Preto;

Vídeo 2: Ambiente interno com iluminação ambiente (superior e influência de iluminação externa) e Fundo Branco;

Video 3: Ambiente interno com fonte luminosa diretamente focada no objeto, com presença de sombras e com Fundo Branco;

Video 4: Ambiente interno com iluminação ambiente (superior e influência de iluminação externa) e com imagem de fundo fixo complexo.

A Tabela 1 mostra as características de cada um dos vídeos realizados para a análise da segmentação. Imagens exemplificando as situações de cada vídeo podem ser vistas na Figura 4.5.

Tabela 1 Relação dos vídeos realizados para análise de segmentação.

	Iluminação	**Background**	**Tamanho**	**Quadros**
Vídeo 1	Interna	Preto	352x288	400
Vídeo 2	Interna	Branco	352x288	400
Vídeo 3	Interna com sombra	Branco	352x288	400
Vídeo 4	Interna	Complexo	352x288	400

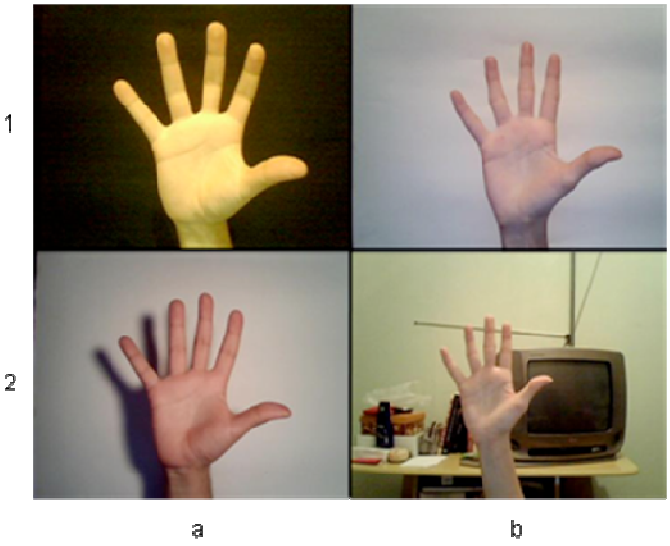

Figura 4.5 Figura com as diferentes representações da imagem de cada sequência de vídeo utilizado para análise da segmentação. Em (1a) representa o vídeo 1 da Tabela 1, (1b) representa o vídeo 2 da Tabela 1, (2a) representa o vídeo 3 da Tabela 1 e (2b) representa o vídeo 4 da Tabela 1.

Cada vídeo continha uma sequência de uma mão direita feminina e masculina realizando movimentos sincronizados de abrir e fechar todos os dedos ou apenas alguns deles. Alguns dos movimentos realizados nas sequencias são apresentados na Figura 4.6.

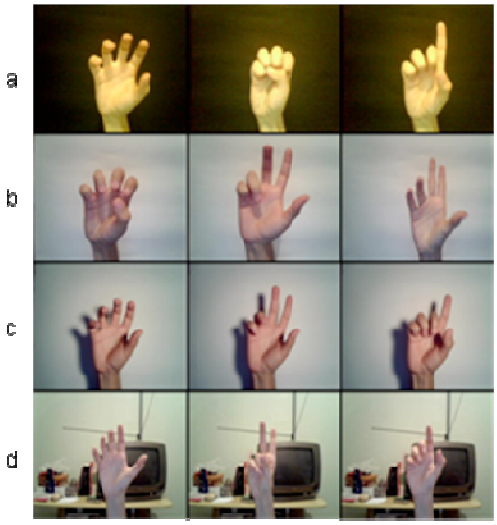

Figura 4.6 Figura contendo imagens que exemplificam o movimento nos vídeos utilizado para análise da segmentação. Na linha (a) representa o vídeo 1 da Tabela 1, (b) representa o vídeo 2 da Tabela 1, (c) representa o vídeo 3 da Tabela 1 e (d) representa o vídeo 4 da Tabela 1.

Um conjunto de imagens foi selecionado para formar uma base de dados de interesse especialmente voltada para o problema proposto, por isso funciona como um estudo de caso. Para tanto, este conjunto contém uma porcentagem das imagens de cada vídeo completando 250 imagens para abordar as diferentes condições que influem no desempenho da segmentação da mão.

A partir desse banco de dados inicial foi criado um segundo banco de dados formado pelas imagens do primeiro banco segmentadas manualmente.

A segmentação manual foi realizada com o auxílio do *software* Adobe® Photoshop®, utilizando a ferramenta de seleção para separar visualmente a região da mão e deixá-la branca e o resto da imagem de cor preta. Essas imagens passaram ainda por um refinamento pelo *software* MATLAB® para transformá-la em imagem binária, através de um limiar em escala de cinza muito próximo do valor de branco. A Figura 4.7 mostra um exemplo de imagem do banco de dados e sua segmentação manual.

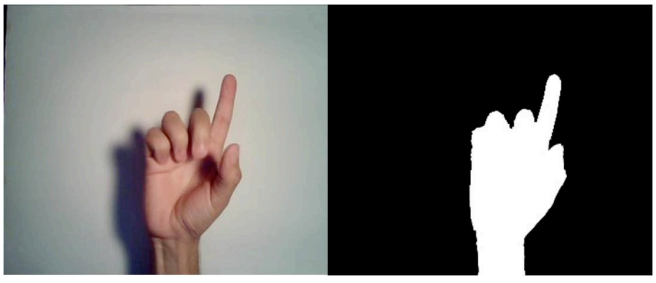

Figura 4.7 Exemplo de Imagem segmentada de maneira manual

4.2.2 ALGORITMOS CONSIDERADOS E CRITÉRIOS PARA ANÁLISE

Os aspectos considerados importantes na escolha da segmentação foram aqueles que respondam bem em relação à sensibilidade às condições de iluminação. Outra questão diz respeito à boa definição do contorno da mão, pois o algoritmo para reconhecimento e rastreamento dos dedos necessita dos pontos do contorno da ponta dos dedos bem definidos para que a interpretação do movimento seja consideravelmente confiável.

Com este objetivo, seis algorítmos de segmentação foram utilizados para se realizar um estudo de caso com ambiente, fundo da imagem e objeto especificados.

Os seis algorítmos são baseados em segmentação por limiar de cor de pele e subtração de fundo. Cada imagem resultante foi comparada com a respectiva imagem segmentada de forma manual. A comparação utiliza conceitos de falsos positivos, taxa de detecção e similiaridade para avaliar seu desempenho. A especificação de cada algoritmo bem como dos critérios de desempenho serão abordados nos itens subsequentes.

4.2.2.1 Algoritmo baseado em espaço RGB

O algorítmo de detecção de pele escolhido para o caso de se utilizar o espaço de cores foi o desenvolvido por Kovac, Peer e Solina (2003) para a condição de iluminação uniforme do dia (*uniform daylight*), segundo a equação 20 que utiliza a operação lógica *AND*.

$$\begin{aligned}&(R > 45)AND\ (G > 40)AND\ (B > 20)AND\ (máx(R,G,B) - mín(R,G,B) < 15)AND\\&(|R - G| > 15)AND\ (R > G)AND\ (R > B)\end{aligned} \tag{20}$$

Este algoritmo também foi utilizado por Ribeiro (2006) para auxiliar no processo de segmentação utilizando GMM.

O algoritmo foi executado para os quatro vídeos realizados no item 4.2.1 e alguns exemplos de resultados para a segmentação podem ser vistos na Figura 4.8, colunas 2, 3 e 4.

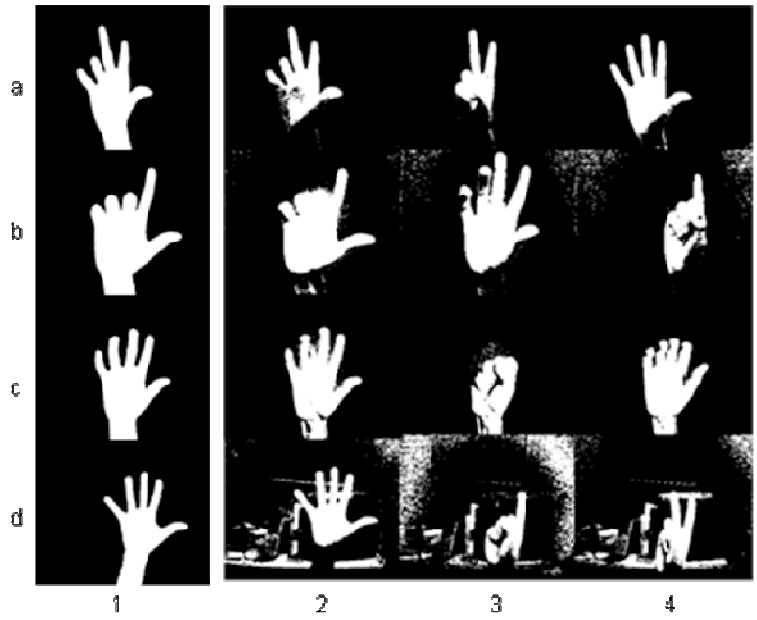

Figura 4.8 A Coluna 1 contém imagens segmentadas manualmente. As colunas 2, 3 e 4 contém exemplos de imagens segmentadas pelo algorítmo RGB de cada vídeo da Tabela 1: Na linha (a) representa o vídeo 1, (b) representa o vídeo 2, (c) representa o vídeo 3 e (d) representa o vídeo 4.

As imagens contidas no Banco de Dados foram então segmentadas e os pixels que satisfizessem a condição da equação 20 foram considerados como pixels de pele humana. As imagens resultantes dessa segmentação foram guardadas para se avaliar posteriormente a performance deste método juntamente com os restantes.

Observa-se que o algoritmo respondeu de forma similar, mostrando uma característica vantajosa de invariância à iluminação e tipos de pele.

4.2.2.2 Algoritmo baseado em espaço RGB normalizado

Um algoritmo utlizado por Kürl e Silva (2004) representado pela equação 21 foi utilizado para exemplificar a utilização do espaço RGB normalizado (rgb) na avaliação de métodos de segmentação.

$$(r \geq 0.35)AND(r \leq 0.465)AND(g \geq 0.28)AND(g \leq 0.363)AND(V \geq 0.25)AND(V \leq 1),$$
$$onde \quad V = \frac{máx(R,G,B)}{255} \tag{21}$$

As imagens do banco de dados foram computadas a partir desse algoritmo e suas respectivas imagens binárias resultantes fomaram um conjuto de comparação a ser também considerado.

Exemplos de alguns quadros segmentados por este algoritmo são mostrados na Figura 4.9.

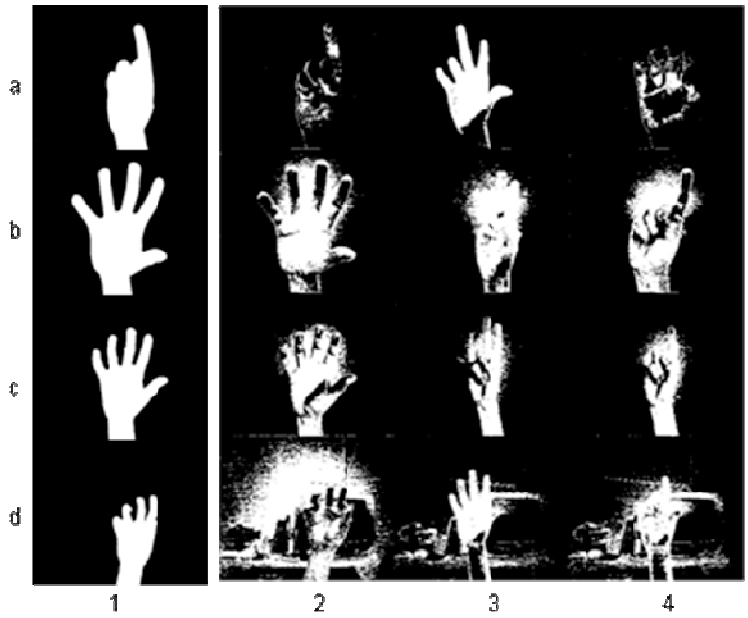

Figura 4.9 A Coluna 1 contém imagens segmentadas manualmente. As colunas 2, 3 e 4 contém exemplos de imagens segmentadas pelo algorítmo RGB normalizado de cada vídeo da Tabela 1: Na linha (a) representa o vídeo 1, (b) representa o vídeo 2, (c) representa o vídeo 3 e (d) representa o vídeo 4.

Este algoritmo apresentou variação de resultados principalmente diante da variação de iluminação, já que em algumas imagens de uma mesma mão poucos pontos foram reconhecidos. No caso de fundo complexos este algoritmo apresentou pior resultado em relação ao que utiliza RGB.

4.2.2.3 Algoritmo baseado em espaço HSV

O algoritmo de segmentação pelas características do espaço de cor HSV foi o utilizado por Sobottka e Pitas (1996). Eles definiram os pixels de uma imagem através de um intervalo nos valores de matiz (*H*) e saturação (*S*) segundo a equação 22.

$$(H \geq 0^{\circ})AND(H \leq 50^{\circ})AND(S \geq 0.23)AND(S \leq 0.68) \quad (22)$$

Para segmentar os quadros dos vídeos, foi necessário converter cada quadro do espaço de cor RGB para HSV.

Nesta operação utlizou-se o comando *rgb2hsv* do MATLAB, porém, este normaliza a matiz, sendo um valor entre zero (0) e um (1), por isso deve-se multiplicar esse valor por 360 para ter a representação em graus.

A Figura 4.10 mostra alguns quadros resultantes desta segmentação para os quatro vídeos selecionados.

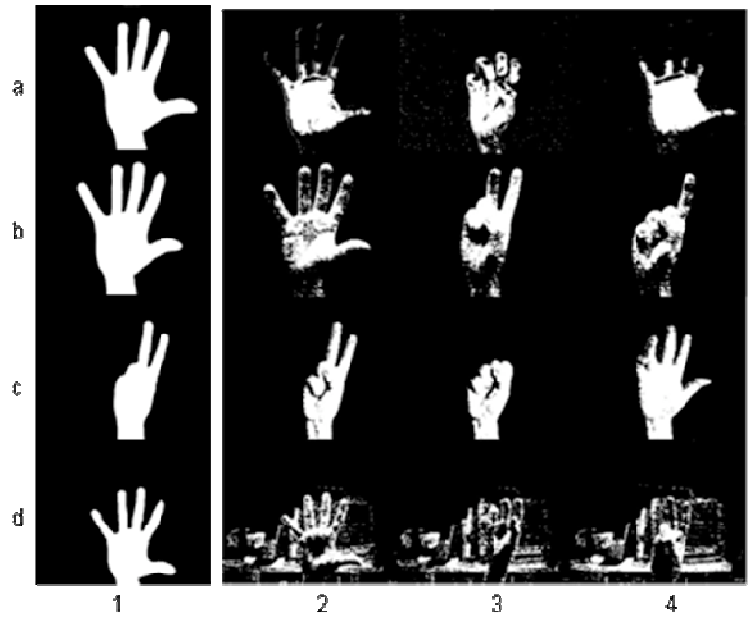

Figura 4.10 A Coluna 1 contém imagens segmentadas manualmente. As colunas 2, 3 e 4 contém exemplos de imagens segmentadas pelo algorítmo HSV de cada vídeo da Tabela 1: Na linha (a) representa o vídeo 1, (b) representa o vídeo 2, (c) representa o vídeo 3 e (d) representao vídeo 4.

As imagens do banco de dados foram também segmentadas e reservadas para que se realizasse o posterior estudo comparativo. Pode-se observar que este algoritmo apresentou uma menor invariância à variação de iluminação.

4.2.2.4 Algoritmo baseado em espaço YCbCr

No caso do espaço de cor YCbCr, o algoritmo de segmentação de face utilizado por Chai e Ngan (1998) foi o escolhido para ser também avaliado. Estes autores fixaram um intervalo no plano de CbCr, segundo a equação 23.

$$(Cb \geq 77)AND(Cb \leq 127)AND(Cr \geq 133)AND(Cr \leq 173) \quad (23)$$

Alguns exemplos de imagens segmentadas por este método, pertencentes aos quatro vídeos utilizados, podem ser vistas na Figura 4.11.

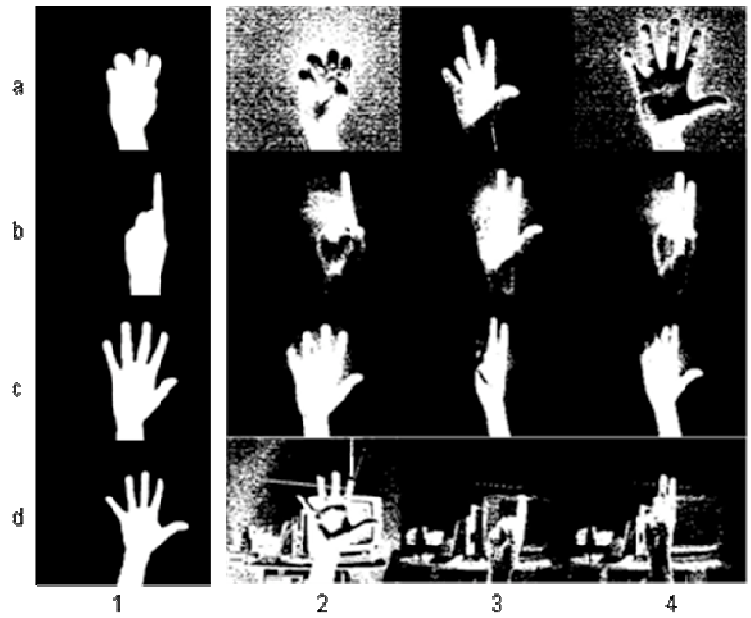

Figura 4.11 A Coluna 1 contém imagens segmentadas manualmente. As colunas 2, 3 e 4 contém exemplos de imagens segmentadas pelo algorítmo YCbCr de cada vídeo da Tabela 1: Na linha (a) representa o vídeo 1, (b) representa o vídeo 2, (c) representa o vídeo 3 e (d) representao vídeo 4.

A partir da Figura 4.11 é possivel observar a variação dos resultados pela condição de iluminação, já que em algumas imagens mais pixels do fundo são reconhecidos como pele humana.

4.2.2.5 Algoritmo baseado em espaço rgb e HSV

Um algoritmo que utiliza dois espaços de cores para definir um píxel como pertencente à pele humana foi definido por Wang e Yuan (2001 apud KAKUMANU et al., 2007,p.1109) e também utilizado neste estudo de desempenho de segmentação.

Segundo Wang e Yuan um pixel é definido como pele humana se obedecer à equação 24.

$$(r \geq 0.36) AND (r \leq 0.465) AND (g \geq 0.28) AND (g \leq 0.363) AND (H \geq 0º) AND \\ (H \leq 50º) AND (S \geq 0.2) AND (S \leq 0.68) AND (V \geq 0.35) AND (V \leq 1) \quad (24)$$

A Figura 4.12 contém algumas imagens, nas condições dos vídeos selecionados, segmentadas segundo este algoritmo. As imagens do banco de dados foram também segmentadas e incluídas na análise quantitativa posterior.

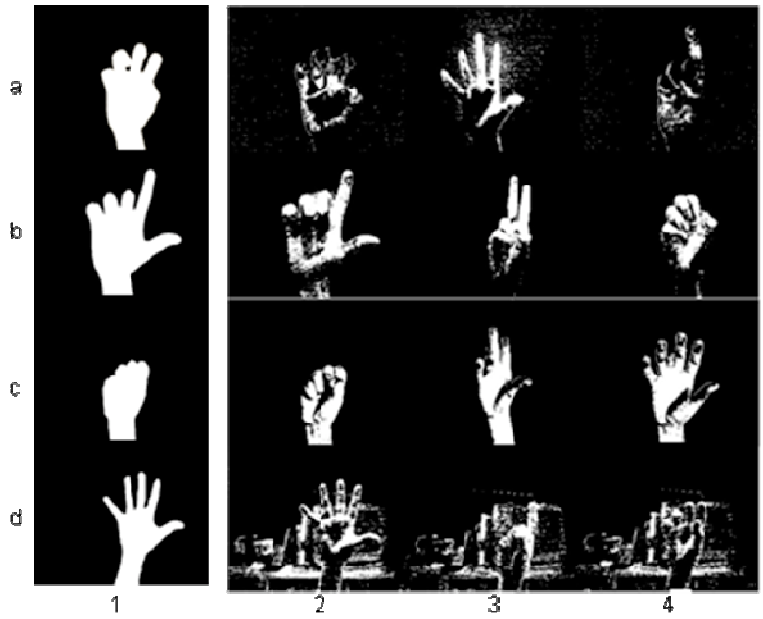

Figura 4.12 A Coluna 1 contém imagens segmentadas manualmente. As colunas 2, 3 e 4 contém exemplos de imagens segmentadas pelo algorítmo proposto por Wang e Yuan (2001 apud KAKUMANU et al., 2007,p.1109) de cada vídeo da Tabela 1: Na linha (a) representa o vídeo 1, (b) representa o vídeo 2, (c) representa o vídeo 3 e (d) representao vídeo 4.

4.2.2.6 Algoritmo baseado em subtração de fundo

Para se realizar a subtração pelo fundo (*Background*), optou-se por se encontrar a imagem de fundo nos primeiros 15 quadros do vídeo, pela mediana dos planos R, G e B separadamente (BEN-ISRAEL, 2007).

A Figura 4.13 mostra a imagem *backgound* em RGB dos quatro vídeos capturados para a análise de segmentação. Este processo irá retirar o movimento da cena e possíveis variações de iluminação serão detectados.

Figura 4.13 Figura contendo as imagens de fundo RGB de cada vídeo selecionado.

A partir dessa imagem, a subtração de cada quadro pelo fundo foi realizada nos planos R, G e B, onde utilizou-se um limiar pela combinação linear de cada plano com mesmo peso, como pode ser observado na equação 25, onde R_I, G_I e B_I representam a imagem e R_B, G_B e B_B as componentes do fundo.

Nesta equação a operação de adição poderia ser substituida pela união. O valor de limiar foi encontrado empiricamente pela observação de que se o valor de limiar fosse menor que o especificado, a imagem tendia a ser mais ruidosa, aumentando a quantidade de pixels classificados como objeto erroneamente. Porém, se o limiar aumentasse de maneira exacerbada, o contorno da mão ficava prejudicado pois buracos apareciam na palma da mão e nos dedos.

A Figura 4.14 (a) mostra uma imagem segmentada a partir de um limiar menor que o escolhido. Já a Figura 4.14 (c) mostra uma imagem segmentada a partir de um limiar maior que o escolhido e a Figura 4.14 (b) apresenta uma imagem resultante da segmentação utilizando o limiar escolhido.

$$\frac{\left(\left|R_I - R_B\right| + \left|G_I - G_B\right| + \left|B_I - B_B\right|\right)}{3} > 25; \qquad (25)$$

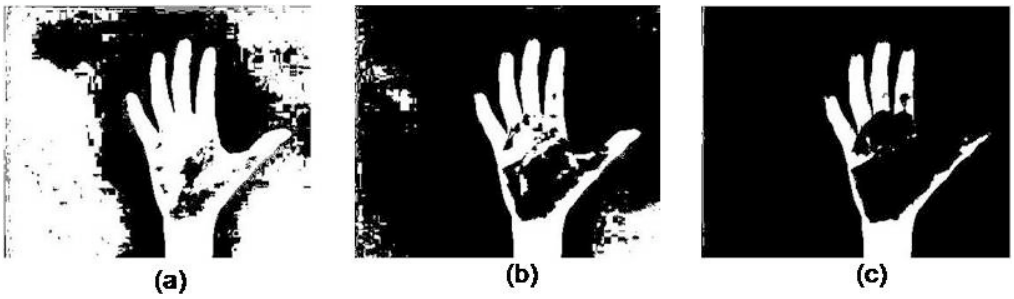

Figura 4.14 Imagem segmentada a partir de: (a) um limiar menor que o escolhido; (b) apresenta uma imagem resultante da segmentação utilizando o limiar escolhido e (c) demonstra uma imagem segmentada a partir de um limiar maior que o escolhido.

Alguns resultados da segmentação dada pela equação 25 podem ser vistos na Figura 4.15. As imagens do banco de dados foram também segmentadas e a imagem binária resultante de cada quadro guardadas para análise de desempenho.

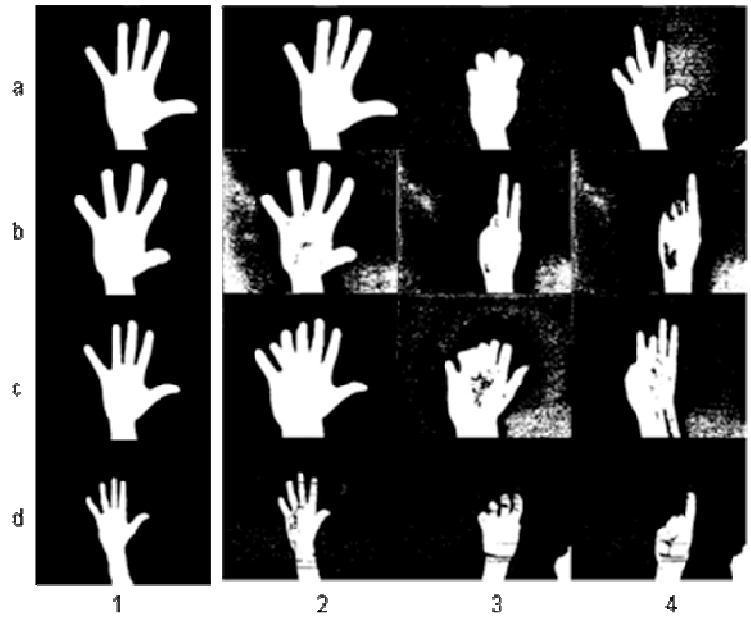

Figura 4.15 A Coluna 1 contém imagens segmentadas manualmente. As colunas 2, 3 e 4 contém exemplos de imagens segmentadas pelo algorítmo de subtração de fundo de cada vídeo da Tabela 1: Na linha (a) representa o vídeo 1, (b) representa o vídeo 2, (c) representa o vídeo 3 e (d) representao vídeo 4.

Observa-se que este algoritmo responde bem independentemente do tipo de pele, já que os objetos contidos na cena são identificados pelo movimento. Porém, esta característica tem grande desvantagem em cenas que contenham sombra como mostra a Figura 4.15 coluna c.

Nestes casos a imagem segmentada identificará um objeto maior que o real, no caso da mão observa-se isso nas imagens que contêm um número maior que cinco dedos.

4.2.2.7 Critérios de desempenho

O desempenho de cada algoritmo de segmentação foi comparado quantitativamente a partir de um conjunto de imagens de referência. A acurácia de cada método foi avaliada por três parâmetros:

Similiaridade de Jaccard

O coeficiente de Jaccard mede a similiaridade entre dois conjuntos, no caso entre duas imagens, sendo que uma delas é considerada a imagem padrão e outra a imagem que se deseja concluir a similiaridade.

A equação 26 contém a definição deste coeficiente, na qual a imagem I_1 será a imagem binária segmentada manualmente para formar o banco de dados padrão de segmentação e a imagem I_2 representa cada imagem segmentada pelos diversos algoritmos aqui considerados.

O coeficiente é então interpretado como similiaridade com a imagem padrão pois encontra a quantidade de pixels que a imagem resultante de um método de segmentação tem em comum com esta imagem, normalizada pela quantidade total de pixels considerados de pele humana na uniaõ das duas imagens.

$$Jaccard = \frac{I_1 \bigcap I_2}{I_1 \bigcup I_2} \tag{26}$$

Taxa de detecção (DR):

Para se determinar a taxa de detecção, do inglês *detection rate* (DR), de certo algoritmo de segmentação considera-se as imagens segmentadas manualmente como imagens de referência para determinar os píxels como sendo de pele ou não-pele.

A taxa de detecção é expressa pela equação 27, onde TP representa a quantidade de píxels *true positive* em um quadro, isto é, que foram corretamente interpretados como píxels de pele e FN representa a quantidade de píxels interpretados como não-pele por certo algoritmo mas que na verdade eram pele, portanto *false negative*.

A figura 4.16 exemplifica pixels *false negative* (FN) e true positive (TP).

$$DR = \frac{TP}{TP + FN} \tag{27}$$

Taxa de falsos alarmes (FAR):

A taxa de falsos alarmes, também chamada *false alarm rate* (FAR) é expressa pela equação 28 onde TP representa o número de pixels true positive, e FP representa o número de pixels considerados de maneira errada como sendo de pele humana, portanto falsos positivos (*false positive*).

A Figura 4.16 ilustra também a interpretação de falsos positivos (FP).

$$FAR = \frac{FP}{TP + FP} \tag{28}$$

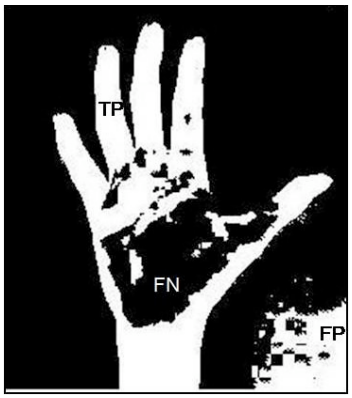

Figura 4.16 Representação de TP (*true positive*), FP (*false positive*) e FN (*false negative*).

Para se determinar os parâmetros TP (*true positive*), FN (*false negative*) e FP (*false positive*) foram ulizadas operações lógicas entre as imagens segmentadas manualmente e as segmentadas automaticamente pelos algoritmos selecionados para o estudo. As equações 29, 30 e 31 expressam essas operações, onde I_1 representa uma imagem segmentada manualmente e I_2 uma segmentada automaticamente (por algoritmo).

$$TP = I_1 \quad AND \quad I_2 \tag{29}$$

$$FN = I_1 \quad AND\left(NOT\left(TP\right)\right) \tag{30}$$

$$FP = \left(I_1 \mid I_2\right) AND\left(NOT\left(I_1\right)\right) \tag{31}$$

4.3 METODOLOGIA

Após a escolha pelos critérios de desempenho do algorítmo de segmentação que melhor respondesse às condições de considerações feitas para o sistema, pôde-se desenvolver uma metolodogia para alcançar o objetivo principal deste trabalho: rastrear os movimentos de dedos de uma mão humana.

Para isso, algumas etapas foram definidas como base e serão abordadas aqui. Estas etapas levaram em consideração alguns aspectos do trabalho realizado por Ribeiro (2006). As etapas são apresentadas abaixo:

1. Aquisição de imagem quadro a quadro por uma webcam;
2. Processamento de cada quadro utilizando segmentação para selecionar os pixels pertencentes à mão;
3. Pós processamento para melhorar a a imagem segmentada;

4. Extração das características da mão pertencente a cada quadro que serão utilizadas como parâmetros para o rastreamento;
5. Normalização do movimento da mão;
6. Determinação dos parâmetros utlizados para caracterizar o rastreamento.
7. Rastreamento do movimento de cada dedo;
8. Tratamento de oclusões;

4.3.1 Aquisição de imagens

Para a aquisição das imagens da mão *online* foram consideradas as imagens capturadas por uma *webcam* via USB ou um arquivo de vídeo com padrão AVI, ambos utilizando as ferramentas de captura de vídeo disponíveis no *Image Acquisition Toolbox*^{TM} do MATLAB®.

A aquisição ocorreu em espaço RGB e com resolução de 352x288 e com uma taxa de 15 ou 22,5 quadros por segundo.

Para a aquisição das imagens utilizadas na seção de resultados e conclusões sobre a solução proposta foram utilizados arquivos de vídeos em formato AVI. Os vídeos foram realizados em ambiente com iluminação interna e superior ao sistema.

4.3.2 Segmentação de imagens

A partir da comparação dos resultados apresentados na fase de determinação do algoritmo de segmentação a ser utilizado, escolheu-se utilizar o algoritmo de segmentação por subtração do fundo em espaço de cor RGB, como apresentado na seção 4.2.2.6. As análises para se chegar a esta decisão podem ser vistas na seção de Resultados 5.1.

Cada quadro capturado foi separado em sua porção R, G e B e processado a partir da equação 25, resultando em uma imagem binária contendo os objetos em movimento na cena, no caso a mão.

Um exemplo de imagem processada a partir deste algorítmo pode ser visto na Figura 4.17.

Figura 4.17 Exemplo de segmentação da imagem (a) por segmentação por subtração de fundo, resutando em (b).

4.3.3 Pós-processamento

Após a segmentação de cada quadro, a imagem binária resultante pode ainda conter pixel falsos, como por exemplo, quando o objeto da imagem binária contém buracos ou o fundo da imagem apresenta objetos ruidosos, como mostra a Figura 4.18(a).

É necessário realizar, então, um pós-processamento para corrigir ou amenizar estes efeitos indesejados.

Para isso, é aplicado um filtro morfológico com as operações de dilatação e erosão (CASTLEMAN, 1996) para melhorar a imagem segmentada. As funções *imdilate* e *imerode* do MATLAB® realizam a filtragem necessária.

Para refinar a segmentação, *blobs* de dimensões reduzidas contidos na mão são eliminados considerando-se como "objeto segmentado" apenas o de maior área. Este procedimento é um filtro mais seletivo que retira objetos que sejam maiores do que um determinado tamanho, mas que não fazem parte da mão.

Para isso, foi utilizada a função do MATLAB® *regionprops*. Após o tratamento com os filtros morfológicos a imagem não apresentou porções da mão separadas, ficando esta completamente conectada.

A Figura 4.18 exemplifica este pós processamento, mostrando uma imagem resultante da segmentação (Figura 4.18 (a)) e outra correspondente após ser filtrada (Figura 4.18 (b)).

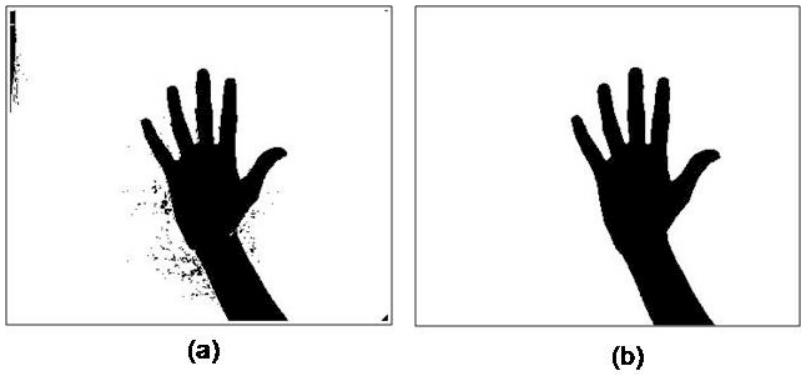

Figura 4.18 Exemplo de (a) Imagem resultante da segmentação (b) Imagem correspondente filtrada.

4.3.4 Extração das características das mãos

Cada imagem resultante do pós processamento continha apenas a mão como objeto, portanto a partir deste ponto foi necessário caracterizá-la, considerando também os seus dedos para que fosse possível realizar o rastreamento de seus movimentos.

Os parâmetros escolhidos para caracterizar a mão foram sua silhueta, as posições das pontas de cada dedo, do centro da mão, a distância do ponto de cada dedo ao centro da mão e o valor do raio da circunferência contida na palma da mão.

A Figura 4.19 apresenta uma mão com os seus parâmetros em vermelho que irão caracterizá-la.

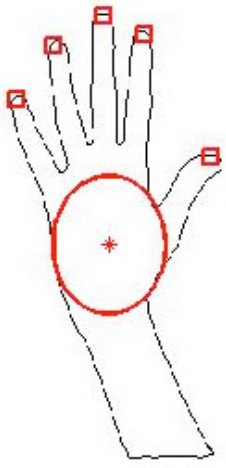

Figura 4.19 Parâmetros que caracterizam a mão para o rastreamento.

As seções que se seguem apresentam a forma que estes parâmetros foram definidos e encontrados em uma imagem contendo uma mão segmentada

4.3.5 Identificação das pontas dos dedos.

Foram consideradas apenas as pontas dos dedos para identificar sua localização, em vez de incluir também as junções entre eles, pois observou-se que se dois dedos ficassem mais próximos, a segmentação das porções inferiores dos dedos não era confiável e não possuiriam valores confiáveis para identificar a posição dos dedos.

Isto pode ser observado na Figura 4.20, na qual o contorno encontrado não conseguiu considerar a junção correta entre os dedos. As pontas de cada dedo podem ser encontradas a partir do contorno da mão, por isso a decisão de um método para encontrar o contorno teve que considerar que o valor destes parâmetros fosse o mais confiável possível.

A decisão do método para encontrar o contorno da mão esteve diretamente ligada à escolha do algoritmo que encontra os pontos de localização das pontas dos dedos, devido a isso esta decisão será discutida posteriormente na seção 4.3.6.

Figura 4.20 Comparação entre contorno real e contorno encontrado da mão

O problema de encontrar os dedos de uma imagem da mão pode ser interpretada como o de encontrar pontos de picos e vales da mão, como exemplificado na Figura 4.21.

A curvatura de cada ponto na curva é definido como a taxa de mudança no ângulo da tangente à curva naquele ponto. Esta curvatura em um contorno pode ser positivo em regiões onde os objetos são convexos, caracterizando um vale, e negativo onde o objeto é côncavo, caracterizando um pico (CASTLEMAN, 1996).

A utilização de derivadas não é uma boa solução na detecção de picos e vales em imagens segmentadas de contorno, já que estas possuem muito ruído e aplicando a derivada muitos pontos do contorno seriam considerados erroneamente como um ponto de máximo ou mínimo local (TRUYENQUE, 2005).

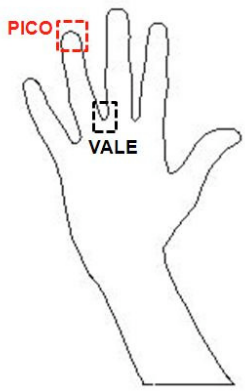

Figura 4.21 Representação do contorno com picos e vales para encontrar as pontas dos dedos

Lopez (2006) utiliza em seu trabalho algumas das soluções existentes para se identificar os picos e vales em uma imagem.

Uma delas utiliza uma técnica chamada Curva de Distância. Esta técnica determina a distância de cada ponto do contorno ao ponto central da mão e constrói uma nova curva com esse valores de distância. Os pontos nesta curva de distância considerados mínimos locais são identificados como um vale na imagem e os pontos máximos locais são os picos da imagem.

Outro algorítmo selecionado no trabalho de Pinho(2006) para encontrar picos e vales é chamado k-curvatura ou k-curva.

Neste algorítmo, proposto primeiramente por Segenand e Kumar (1998 *apud* Lopez, 2006, p. 48) e também utilizado por Truyenque (2005), o cálculo da curvatura em cada ponto do contorno é encontrado a partir do ângulo entre dois vetores mostrados na equação 32, onde $P(i)$ representa um píxel do contorno da mão, $P(i-k)$ representa um pixel pertencente ao contorno e anterior em k pontos ao ponto $P(i)$ de referência, e $P(i+k)$ representa um ponto do contorno posterior em k pontos do ponto de referência.

A Figura 4.22 mostra a porção de um contorno com alguns píxels para exemplificar os três pontos considerados do contorno e como o ângulo entre dois vetores pode ser utilizado para caracterizar a curvatura de um contorno.

Para que seja possível implementar este algorítmo é necessário que o contorno da mão seja fechado, já que em um vetor contendo os pontos do contorno para i de 1 (um), o primeiro ponto guardado no vetor, até i igual ao último ponto do contorno é necessário identificar quais são os k pontos anteriores do vetor.

$$\vec{v}_1 = [P(i-k), P(i)]$$
$$\vec{v}_2 = [P(i+k), P(i)] \quad (32)$$

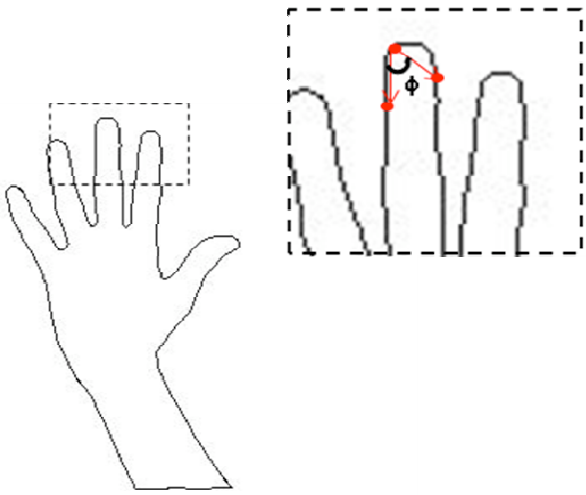

Figura 4.22 Ângulo de curvatura do contorno da mão utilizando k-curvatura.

Pode-se observar que em uma curvatura ruidosa, o algorítmo consegue ajustar, pelo número de pontos k considerados, a sua sensibilidade a estes ruídos, já que aumentando o intervalo de pontos a se considerar, os pontos pertencentes às pequenas imperfeições na imagem não serão considerados, portanto sua curvatura não será encontrada.

O ângulo entre os dois vetores $\vec{v}_1$ e $\vec{v}_2$ pode ser econtrado utilizando a definição de produto escalar.

O produto escalar entre dois vetores $\vec{v}_1$ e $\vec{v}_2$ é definido pela equação 33, onde ϕ é o ângulo formado por eles, portanto, o ângulo entre dois vetores pode ser encontrado pela equação 34 e é definido em intervalo de zero (0) a 180 graus.

$$\vec{v}_1 \bullet \vec{v}_2 = \left|\vec{v}_1\right| \cdot \left|\vec{v}_2\right| \cdot \cos(\phi) \quad (33)$$

$$\phi = \arccos\left(\frac{\vec{v}_1 \bullet \vec{v}_2}{\left|\vec{v}_1\right| \cdot \left|\vec{v}_2\right|}\right). \qquad (34)$$

Na identificação das inflexões (picos ou vales) da imagem, que representassem os dedos, utilizou-se um valor ângulo de curvatura *threshold* T_ϕ tal que para cada ponto do contorno da mão foi calculado o ângulo formado entre os dois vetores $\vec{v}_1$ e $\vec{v}_2$ pelas equações 32 e 34.

Se este ângulo fosse menor que T_ϕ era considerado como um ponto candidato à inflexão, isto porque todos os pontos pertencentes às pontas dos dedos ou às suas junções apresentam ângulo de curvatura parecidos, como mostra a Figura 4.23 que representa em forma gráfica os ângulos de curvatura de todos os pontos pertencentes ao contorno (em azul) e uma reta (em vermelho) com o valor de threshold T_ϕ. Nesta curva pode-se identificar nove porções abaixo da curva threshold, por isso nove conjuntos de pontos candidatos à inflexão.

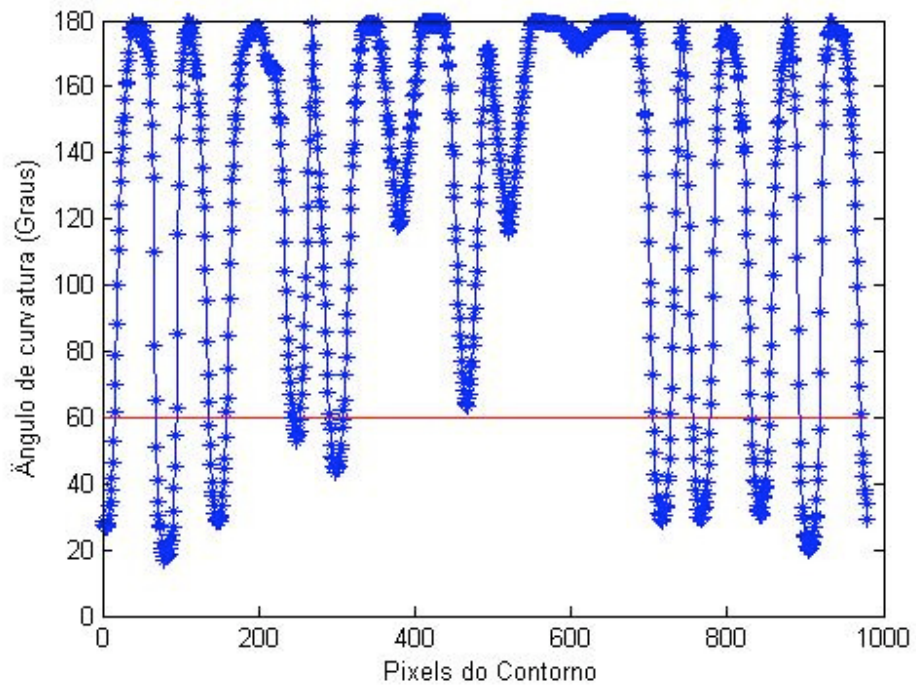

Figura 4.23 Ângulo de curvatura de cada pixel do contorno da mão.

Para identificar apenas um ponto que caracterize cada inflexão do contorno, os pontos candidatos à inflexão que são adjacentes entre si são guardados juntamente com os seus

respectivos ângulos de curvatura formando um mesmo conjunto, até que não haja mais candidatos adjacentes.

O ponto de inflexão é definido como o ponto pertencente a um mesmo conjunto de pontos candidatos adjacentes e que possuam a menor curvatura.

Observando novamente a Figura 4.23, percebe-se que a última porção do gráfico foi unida ao grupo dos primeiros pontos, já que o algorítmo considera uma curva fechada.

Ajustou-se os valores de k e T_ϕ em uma imagem contendo uma mão aberta para que identificasse todos os dedos e se chegou a um valor de k igual a 30 e T_ϕ 60 graus. A Figura 4.24 apresenta o contorno utilizado para encontrar estes valores de k e T_ϕ com todos os pontos de inflexão.

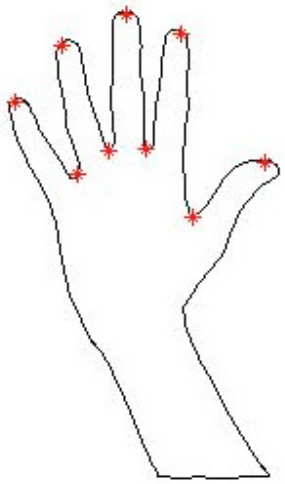

Figura 4.24 Contorno da mão com picos e vales detectados.

Como os ângulos de curvaturas variam de zero a 180 graus, este algorítmo não diferencia os pico dos vales. Para que apenas os picos fossem identificados, incluiu-se a consideração da direção dos vetores $\vec{v_1}$ e $\vec{v_2}$ para fazer a distinção.

Como demostra a Figura 4.22, a direção dos vetores em coordenadas cartesianas para valores de um pico terá sentido para aumento dos valores de *x* e no caso dos vales o vetor terá sentido para diminuição de *x*. Assim, pode-se considerar o valor do vetor em *x* e, se este for negativo classifica um vale, caso contrário um pico. Assim, apenas os pontos classificados como pico são considerados.

A Figura 4.25 exemplifica a ação do algoritmo k-curvatura em uma imagem contendo a mão, onde apenas as pontas dos dedos são identificadas pelos retângulos em vermelho.

Figura 4.25 Contorno da mão com as cinco pontas dos dedos detectadas.

Foi criada uma função chamada Kcurva para realizar as operações descritas,que pode ser vista na Figura 4.26. Esta função é apresentada também no Apêndice A.

```
Para cada ponto no cortorno Pi ,
        Vetor 1 recebe [Pi-k,Pi]
        Vetor 2 recebe [Pi+k,Pi]
        Se fila de candidatos adjacentes vazia
                Se ângulo entre vetor 1 e 2 menor que Threshold
                Fila de candidatos adjacentes recebe Pi
                Fim Se
        Senão
                Se ângulo entre vetor 1 e 2 menor que Threshold
                        Fila de candidatos adjacentes recebe Pi
                Senão
                        Encontro Pi de menor curvatura na fila
                        Se Pi for Pico
                                Selecione Ponto
                        Fim Se
                Fim Se
        Fim Se
Fim Para
```

Figura 4.26 Algoritmo da função Kcurva.

4.3.6 Identificação do contorno da mão.

A decisão do método para encontrar o contorno da mão esteve diretamente ligada ao método para encontrar as pontas dos dedos, pois a qualidade do contorno possibilitava maior confiabilidade na classificação de sua curvatura para encontrar os pontos de interesse.

Inicialmente o contorno foi encontrado utilizando os operadores de Sobel (CASTLEMAN, 1996) que encontram pontos onde o gradiente da imagem é máximo. Para isto, utiliza-se o comando MATLAB *edge.*

A imagem de contorno resultante desta solução tem a característica, em alguns casos possuir muitas imperfeições e ser muito ruidosa, como exemplifica a Figura 4.27 (a).

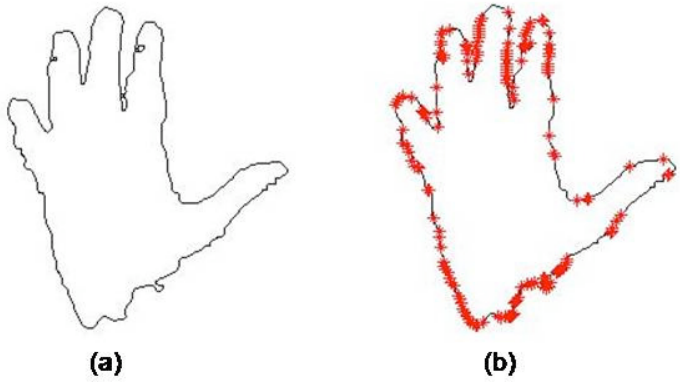

Figura 4.27 Exemplo de contorno ruidoso com pontos detectados pela k-curvatura.

Quando aplicado o algorítmo k-curvatura em imagens que possuiam esta característica, o resultado era a identificação de muitos pontos de falsa inflexão, o que impossibilitou o uso dos operadores de sobel para encontrar o contorno, já que o algoritmo k-curvatura possui a característica de filtrar os ruídos menores.

Entretanto, as imagens possuiam imperfeições maiores imunes a esta filtragem, e para que estes pontos fossem retirados, seria necessário aumentar a seletividade do algoritmo de k-curvatura pelo aumento do parâmetro k, mas neste caso, tinha-se o risco de o algorítmo ignorar também as pontas dos dedos.

A Figura 4.27 (b) exemplifica uma situação de contorno em que nos valores de limiar especificados na seção 4.3.5 de k igual a 30 e T_ϕ 60 graus muitos pontos foram considerados erroneamente como ponta dos dedos.

A solução para este de problema foi encontrar o contorno através do conceito de MPP (*minimum-perimeter polygon*), utilizando um algoritmo desenvolvido por Gonzalez, Woods e Eddins (2003) a partir da função chamada *minperpoly* que faz parte da biblioteca DIPUM (*Digitam Image Processing Using Matlab*), disponível na internet para utilização juntamente com o MATLAB.

Este algoritmo encontra um polígono aproximado de uma região ou um contorno de uma imagem, considerando apenas polígonos simples.

Portanto, se a região tiver áreas internas, estas são desconsideradas(GONZALEZ; WOODS; EDDINS, 2003), característica interessante para a proposta deste trabalho, pois mesmo que restassem buracos na imagem após passar pelos filtros morfológicos estes não são considerados nesta fase, restando apenas o contorno da mão.

A Figura 4.28 exemplifica a utilização da função *minperpoly* no contorno da Figura 4.27 (a), neste caso, os pontos que representam as inflexões do contorno, os asteriscos vermelhos da mesma figura, foram encontrados corretamente.

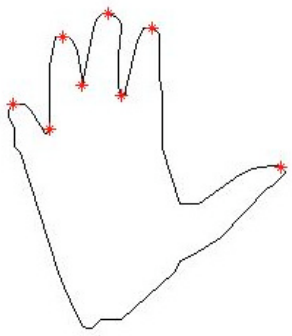

Figura 4.28 Contorno encontrado pela aproximação pelo polígono de mínimo perímetro.

4.3.7 Normalização do movimento da mão

Para que o rastreamento do movimento da mão fosse independente da distância desta em relação à webcam e também de diferentes tamanhos das mãos, utilizou-se a localização do centro da mão e o raio de uma circunferência interna à mão e com centro igual ao da mão.

A técnica para localizar o centro da palma da mão baseia-se na Transformada da Distância Euclidiana (TDE) (RIBEIRO, 2006), que converte uma imagem binária em uma outra imagem que contém em seu valor de intensidade de cada pixel o valor da mínima distância ao limite do contorno da mão.

O centro da palma será definido como sendo o pixel de maior intensidade na imagem resultante desta transformação, que representa o pixel da mão com maior distância em relação ao contorno.

Esta distância foi definida como o raio de uma circunferência interna à mão. A Figura 4.29 exemplifica uma imagem de intensidades encontrada a partir da transformada de distância euclidiana.

Figura 4.29 Imagem de intensidades resultante da Transformada da Distância Euclidiana.

A transformada da distância euclidiana foi encontrada a partir do comando MATLAB *bwdist*. O algoritmo para definir o centro e raio de uma imagem binária contendo a mão segmentada e pós-processada foi implementado da seguinte maneira:

1. Encontrar transformada da distância da imagem segmentada (*bwdist*);
2. O raio é dado pelo máximo valor da transformada;
3. Encontrar ponto com máxima intensidade na imagem resultante da transformada;
4. O centro da mão terá posição deste ponto.

A independência dos movimentos dos dedos em relação aos movimentos da mão foi possível considerando sua posição em relação ao centro da mão, isto é, sendo representada pelo ponto $P_{dedo} = (x_D, y_D)$ e o centro por $P_{centro}(x_C, y_C)$ e r o raio interno à mão, a posição do dedo normalizada será definida a partir da equação 35, isto é, a posição de um dedo será definida pela distância euclidiana entre o ponto de pico que representa a sua ponta pelo centro da palma da mão, normalizada pelo raio da circunferência interna à ela.

$$Posição_{dedo} = \frac{\sqrt{(x_D - x_C)^2 + (y_D - y_C)^2}}{r} \tag{35}$$

4.3.8 Rastreamento do movimento dos dedos

O problema do rastreamento de um objeto pode ser definido como o problema de estimar a trajetória de um objeto no plano da imagem enquanto se move ao redor de uma cena

(YILMAZ et al., 2006). Para isto, um algoritmo rastreador necessita identificar alguns atributos do objeto para serem monitorados em cada quadro de um vídeo.

O objeto a ser rastreado pode ser definido a partir de qualquer característica que seja interessante para posterior análise de seu movimento.

Neste trabalho, a mão humana é definida por sua silhueta de onde são extraídos, pelo algoritmo k-curvatura, os pontos que identificam a ponta dos dedos, pela localização de seu centro e tamanho do raio da circunferência interna à ela, com o intuito de caracterizar.

Após a detecção dos parâmetros que caracterizam a mão, o rastreamento consiste em encontrar a trajetória dos seus dedos para identificar o movimento de abaixá-lo e levantá-lo. Portanto, este movimento deve ser independente do movimento de translação da mão como um todo. Esta independência foi garantida pela normalização do movimento dos dedos, apresentado na seção 4.3.7.

Para encontrar a correspondência do conjunto de pontos que localizam as pontas dos dedos de um quadro k, imagem $I(k)$, com o conjunto do quadro posterior, imagem $I(k+1)$, definiu-se como custo a velocidade de cada ponto do quadro k em relação a todos os pontos do quadro $k+1$. Assim, sendo $P_k = (x_k, y_k)$ um ponto do dedo de um quadro k a velocidade em relação a um ponto $P_{k+1} = (x_{k+1}, y_{k+1})$, localizado no quadro $k+1$, será dada pela equação 36. A correspondência entre os pontos do quadro k e $k+1$ será encontrada quando houver a da minimização deste custo.

$$Velocidade = \frac{\sqrt{(x_{k+1} - x_k)^2} + (y_{k+1} - y_k)^2}{(k+1) - k} \tag{36}$$

Na inicialização do rastreamento dos dedos foi considerado um valor de referência em que a mão estivesse totalmente estendida e que todos os dedos fossem reconhecidos, para que posteriormente fosse possível comparar os movimentos. Quando o primeiro quadro para o rastreamento é definido, um objeto é criado para guardar as informações que caracterizam este objeto, portanto este objeto contém uma matriz em que cada linha representa um dedo, e armazena o nome do dedo, dado por um número de um a cinco, a posição deste na imagem e sua distância em relação ao centro da mão. As posições dos dedos são ordenadas e numeradas a partir do dedo anelar que se encontra o mais extremo à esquerda da imagem e é nomeado o dedo “1”. Isto repete-se até que o dedo polegar é nomeado como dedo “5”. A Figura 4.30 exemplifica um quadro que inicializa o rastreamento, reconhecendo os cinco dedos e com a mão completamente estendida e a Tabela 2 .apresenta o objeto armazenado deste quadro.

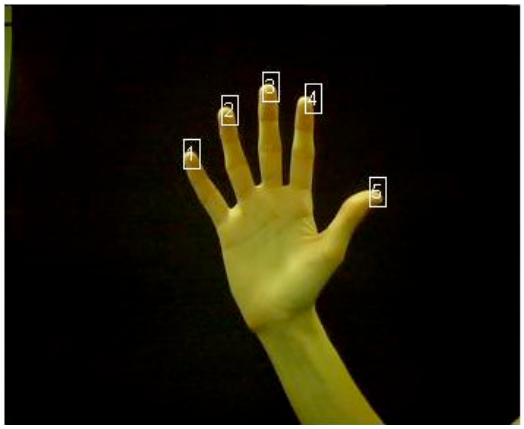

Figura 4.30 Imagem de intensidades resultante da Transformada da Distância Euclidiana.

Tabela 2 Exemplo de matriz objeto que representa a mão em cada quadro.

DEDO	POSIÇÃO x	POSIÇÃO y	DISTÂNCIA NORMALIZADA
1	96	125	2,7696
2	67	149	3,2864
3	51	182	3,655
4	61	210	3,5157
5	127	253	2,6649

4.3.9 Tratamento de oclusões

A partir do primeiro quadro rastreado garantiu-se que fossem reconhecidos cinco pontos picos no contorno pelo algoritmo de k-curvatura que representaria as pontas dos cinco dedos, porém do segundo quadro em diante poderia acontecer que nenhum ponto ou um número menor que cinco pontos fossem reconhecidos, o que é explicado por algumas hipóteses:

- Oclusão entre dois dedos, caso estes se aproximem o suficiente que não consiga ser reconhecido o contorno entre eles, caso exemplificado pela Figura 4.31 que mostra a imagem capturada onde dois dedos estão próximos o suficientes para não serem reconhecidos dois picos pelo algoritmo k-curvatura;
- Oclusão de um dedo ou todos os dedos com a palma da mão, que representa o movimento de abaixar os dedos o mais próximo possível da palma, ou fechar a mão.;

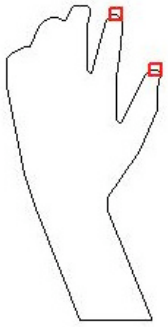

Figura 4.31 Exemplo de oclusão entre os dedos com apenas dois pontos detectados pelo algoritmo de k-curvatura.

É essencialmente necessário que o algoritmo resolva a segunda hipótese, para caracterizar por completo o rastreamento ao abaixar e levantar os dedos.

Se a posição dos dedos não é reconhecida ou considerada nos quadros em que não são encontrados picos no contorno da mão, não é possível dar continuidade confiável ao movimento dos dedos.

Por consequência, foi inserido no algoritmo de rastreamento uma porção que tratasse o problema de oclusão dos dedos. Esta solução é conseguida considerando-se que o rastreamento ocorre, primeiramente pela correspondência entre os pontos encontrados a partir da k-curvatura com os pontos do quadro anterior, e estes são guardados na matriz com o nome que o equivale pela minimização do custo.

Se o número dos pontos encontrados é menor que cinco nas linhas da matriz onde o ponto nao é reconhecido, as informações deste dedo são marcadas para que a parte do algoritmo de tratamento de oclusão reconheça esta situação.

O tratamento de oclusão reconhece os dedos marcados e encontra a distância entre a posição deste ponto no quadro anterior por todos os pontos pertencentes ao contorno da mão no quadro atual e considera a posição do ponto com a menor distância encontrada como sendo a posição mais provável deste dedo Isto é, sendo a posição no quadro anterior P_{k-1} de um dedo nao reconhecido, P_i o i-ésimo ponto do contorno da mão e n a quantidade de pontos de contorno, sua posição P_k no quadro atual é dada pela equação 37.

$$P_k = \min\left(\left|P_i - P_{k-1}\right|, i = 1..n\right) \tag{37}$$

Portanto, o algoritmo de rastreamento do movimento dos dedos poder ser resumido da seguinte maneira:

1. Encontrar no quadro atual pontos de picos pelo algoritmo de k-curvatura;

2. Achar equivalência destes pontos com os dedos encontrados no quadro anterior;
3. Se o número de pontos for menor que cinco pular para passo 5, senão próximo passo;
4. Econtrar dedos ainda não identificados pela menor distância de sua posição no quadro anterior ao contorno do quadro atual;
5. Guardar informações de cada dedo reconhecido com seu nome correspondente na matriz de objeto do quadro atual.

4.4 MATERIAL

O computador utilizado para os experimentos deste trabalho possuia processador Intel® CoreTM 2 Duo T5550, memória (RAM) de 3061MB, disco rígido de 160GB e com sistema operacional Windows VistaTM Home Premium 32 bits. As imagens foram capturadas por uma câmera *webcam* com resolução de 352x288 e com uma taxa de 22,5 quadros por segundo.

O algoritmo foi desenvolvido utilizando MATLAB® e suas funções pertencentes ao *Image Aquisition Toolbox 3.0®* e *Image Processing Toolbox™*, além do DIPUM *Toolbox* que acompanha o livro *Digital Image Processing Using MATLAB*, primeira edição (GONZALEZ; WOODS; EDDINS, 2003).

O DIPUM *Toolbox* primeira versão possui um conjunto de 70 funções no formato MATLAB® para processamento de imagem, que abrangem:

- Transformações Geométricas;
- Análise de imagem (segmentação, descrição, reconhecimento);
- Compressão de imagem;
- Filtros lineares e não-lineares;
- Processamento de imagens coloridas;
- Operações utilizando arrays (GONZALEZ; WOODS; EDDINS, 2003)

4.5 CONSIDERAÇÕES FINAIS

Este capítulo apresenta o problema baseado em visão computacional para rastrear os movimentos dos dedos de uma mão humana e propõe uma metodologia para solucioná-lo desde sua fase de aquisição de imagens até o tratamento de possíveis oclusões presentes nos quadros.

As imagens foram captadas a partir de uma *webcam*, foram segmentadas por subtração de fundo e tratadas num processo de pós-processamento utilizando filtros morfológicos para remover pequenas regiões e preencher os buracos.

O algoritmo caracteriza-se pela combinação de várias metodologias abordadas na visão computacional, como a utilização da Transformada de Distância Euclidiana (TDE), para encontrar o centro da mão e o raio da palma, e do algoritmo k-curvatura para detectar as pontas dos dedos da mão.

A fase de segmentação foi analisada de forma mais completa por um estudo comparativo de algoritmos de segmentação baseados em limiares de com e algoritmo utilizando subtração de badckground em espaço de cor RGB.

CAPÍTULO 5

5 RESULTADOS E CONCLUSÕES

Este capítulo apresentará separadamente os resultados referentes , na seção 5.1, à fase de avaliação dos métodos de segmentação de imagem, e na seção 5.2, ao algoritmo proposto para rastreamento dos movimentos dos dedos.

5.1 MÉTODOS DE SEGMENTAÇÃO

Os resultados da análise quantitativa de cada um dos seis algoritmos de segmentação considerados para o estudo de desempenho foi organizado na Tabela 3 e representam os valores médios encontrados de similiaridade, taxa de detecção e taxa de falsos alarmes, bem como valor médio de tempo de processamento de cada algorítmo.

Os métodos que apresentaram melhores desempenhos foram os baseados em *threshold* em espaço de cor RGB (KOVAC; PEER; SOLINA, 2003) e o algoritmo proposto de subtração de *background*, já que apresentaram melhor taxa de detecção e maior similiaridade com as imagens padrões de segmentação e com a razão de a taxa de falsos alarmes consideravelmente pequena.

Os métodos de segmentação em espaço de cor RGB normalizado (KÜRL; SILVA, 2004) e HSV (SOBOTTKA; PITAS, 1996) apresentaram pequena taxa de falsos alarmes, porém observando seus baixos valores de taxa de detecção e de similiaridade, percebe-se que este parâmetro não é suficiente para caracterizar um método de segmentação. No caso, estes valores baixos de similiaridade e taxa de detecção poderiam caracterizar um método não muito robusto em relação à mudanças de iluminação ou na detecção de peles de diferentes raças.

O algoritmo que utiliza o espaço de cor YCbCr (CHAI; NGAN, 1998) apresentou um valor de taxa de detecção alta, por outro lado possuia alta taxa de falsos alarmes o que explica o resultado baixo de similiaridade e justifica o uso destes três parâmetros para caracterizar os algoritmos.

Tabela 3 Tabela com os valores dos parametros de comparacao e tempo de execução dos algoritmos de segmentação analisados.

MÉTODO	Similiaridade Média	Taxa de Detecção Média	Taxa de Falsos Alarmes Média	Tempo Médio (s)
RGB	0,6988	0,7691	0,1109	0,0190
RGB normalizado	0,4680	0,5899	0,2053	0,0284
HSV	0,4825	0,5343	0,1943	0,0895
YCbCr	0,5476	0,7420	0,3219	0,0221
RGB normalizado e HSV	0,4450	0,5123	0,1581	0,0507
Subtração de *Background*	0,7126	0,9717	0,2740	0,0254

Em relação aos tempos de execução dos algoritmos, o algoritmo em espaço HSV possui processamento mais demorado possivelmente devido à necessidade da conversão da imagem do espaço RGB para HSV e, no caso do algoritmo que utiliza dois espaços de cores é esperado que seja mais lento já que realiza um passo a mais em seu processamento.

Em uma segunda fase, realizou-se uma análise visual da ação dos algoritmos de segmentação nos videos que continham o fundo negro, escolhido para realizar a segunda fase de rastreamento do movimento dos dedos.

Algumas imagens dos vídeos apresentados na Tabela 1 da seção 4.2.1 são apresentadas na Figura 5.1, assim como as imagens resultantes da segmentação, para exemplificar esta análise.

Nesta mesma figura, a coluna **a** contém as imagens originais dos vídeos, a coluna **b** sua imagem correspondente resultante da segmentação pelo método usando *threshold* em espaço RGB definido por Kovac, Peer e Solina (2003) e a coluna **c** contém as imagens segmentadas pelo algoritmo baseado em espaço RGB normalizado utilizado por Kürl e Silva (2004). A coluna **d** contém o algoritmo de segmentação pelas características do espaço de cor HSV utilizado por Sobottka e Pitas (1996); a coluna **e** o algoritmo de segmentação em espaço de cor YCbCr utilizado por Chai e Ngan (1998); a coluna **f** algoritmo baseado nos espaços de cores RGB normalizado e HSV definido por Wang e Yuan (2001) e finalmente a coluna **g** contém o algoritmo baseado em subtração de *background* em espaço RGB.

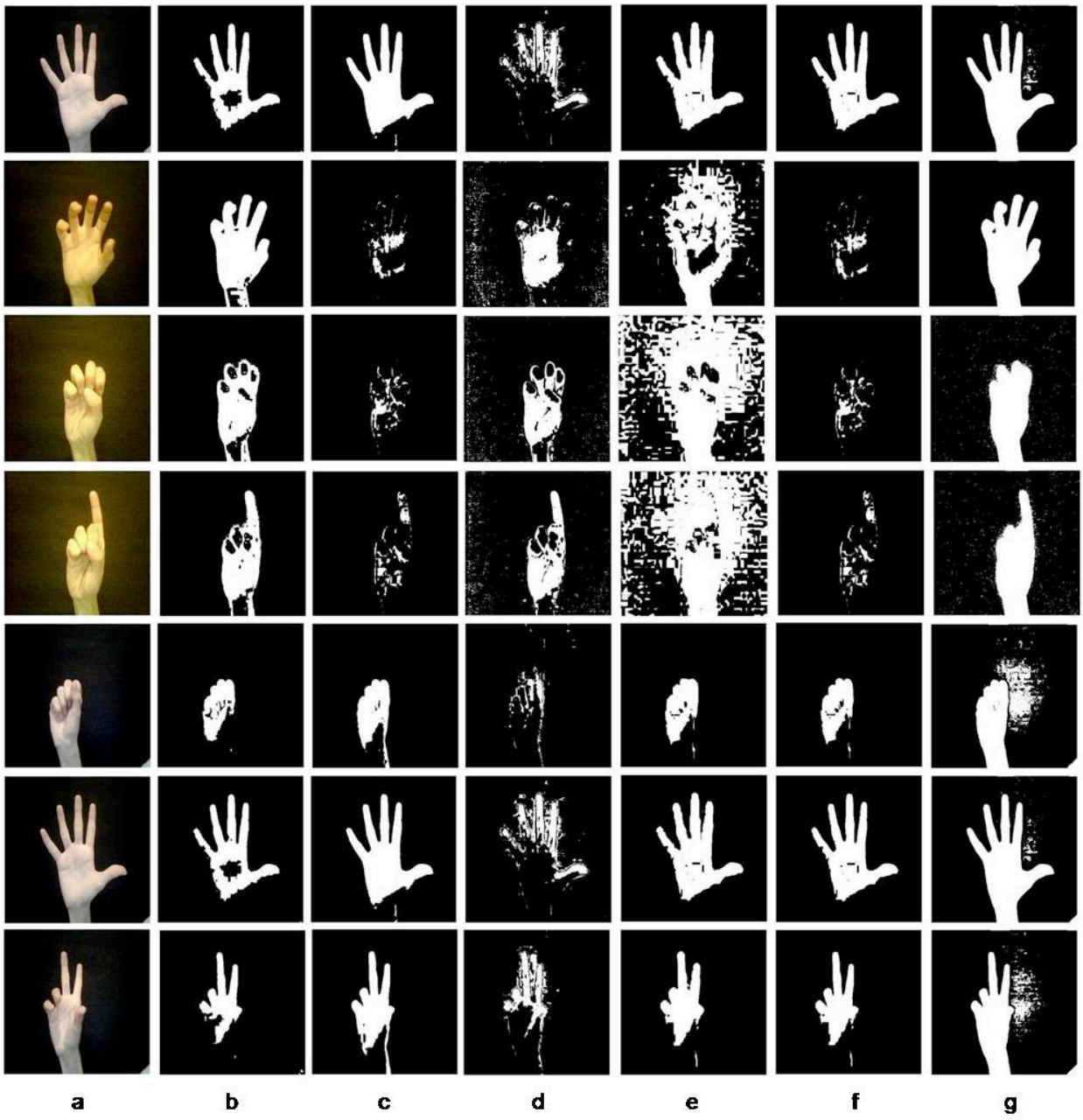

Figura 5.1 A coluna (a) contém imagens utilizadas para exemplificar a análise visual dos algoritmos de segmentação (b) Imagens correspondentes segmentadas pelo algoritmo baseado em limiar RGB; coluna (c) contém as imagens segmentadas pelo algoritmo baseado em espaço RGB normalizado; coluna (d) imagens resultantes da segmentação pelo algoritmo no espaço de cor HSV; na coluna (e) o algoritmo de segmentação em espaço de cor YCbCr; a coluna (f) algoritmo baseado nos espaços de cores RGB normalizado e HSV definido por Wang e Yuan (2001) e finalmente a coluna g comtém o algoritmo baseado em subtração de *background*.

Observa-se que o algoritmo proposto por Kovac, Peer e Solina (2003) revelou-se mais robusto nos casos de variação de iluminação nas cenas ou para mãos de diferentes pessoas. Porém, de forma geral as imagens segmentadas pelo algoritmo de subtração de *background* apresentaram melhor definição na silhueta e sua resposta à variação de iluminação foi menos prejudicial para a detecção da mão.

Portanto foi escolhido o algoritmo de subtração de *background* para realizar a segunda fase deste trabalho e também para garantir invariância para diferentes pessoas. Sua desvantagem é mais relevante em cenas que continham sombra, limitando-se à utilização de fundo negro sem foco de luz horizontal na mão pôde-se sanar esta dificuldade.

5.2 RASTREAMENTO DO MOVIMENTO DOS DEDOS

Os resultados são abordados primeiramente pela análise visual utilizando um vídeo que representa o movimento de fechar e abrir as mãos, a fim de verificar como o algoritmo se porta tantos nos casos de rastreamento dos pontos de picos da imagem, bem como nos quadros em que é necessário proceder com o tratamento de oclusão.

As imagens foram capturadas numa velocidade de 22,5 quadros por segundo e o rastreamento utilizava imagens a cada dois quadros.

A Figura 5.2 apresenta alguns quadros capturados para exemplificar o movimento realizado no vídeo. A cada quadro foram reconhecidos cada dedo com sua nomeação correta.

Observou-se que no caso em que alguma porção da mão ficasse oclusa por muitos quadros, um número maior que 15, podem ocorrer erros no resultado do rastreamente devido à solução de oclusão. Um exemplo do tipo de erro observado é apresentado na Figura 5.3.

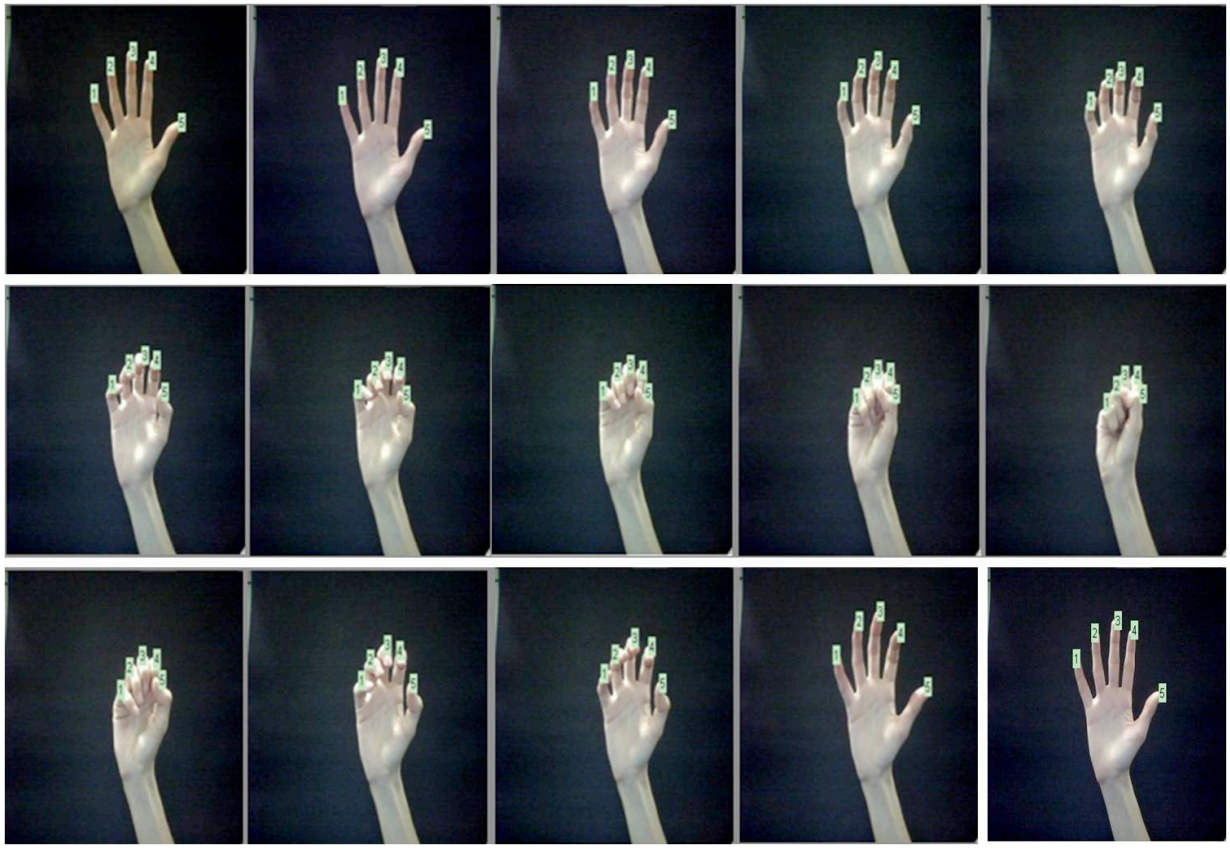

Figura 5.*23* Exemplo de quadros de sequência de vídeo com movimento de abrir e fechar a mão, utilizada para analisar o algoritmo de rastreamento.

Entretanto, analisando o tempo médio de execução do algoritmo em cada quadro, em um vídeo com 1000 quadros este tempo é de 0,6680 segundos. Portanto para a análise *online* a cada segundo aproximadamente 1,5 quadros seriam analisados e aconteceriam problemas se a pessoa ficasse mais que 10 segundos com a mão fechada, por exemplo. O que demostra que o algoritmo proposto responde consideravelmente bem no caso de oclusão.

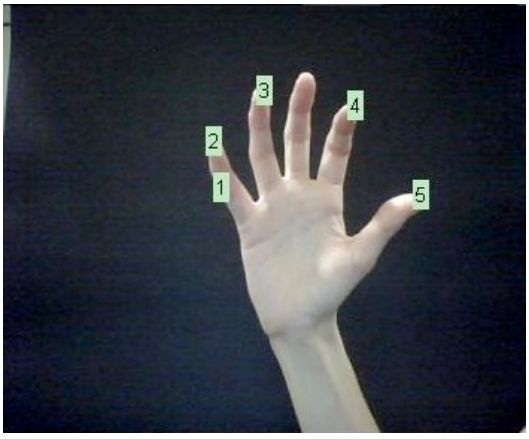

Figura 5.4 Exemplo de falha do tratamento de oclusão.

Observou-se que é possível a partir da análise do posicionamento de cada dedo em uma sequência de quadros, interpretar um gesto dinâmico da mão pela combinação destes posicionamentos nos cinco dedos.

A Figura 5.4 apresenta uma sequência de imagens capturadas de um vídeo em que a mão inicia completamente aberta e realiza movimento indicando o número dois e depois retorna a ficar aberta.

Observa-se que neste movimento os dedos nomeados “1”, “2” e “5” possuem movimentos similares, fechando e abrindo e os dedos “3” e “4” ficam aproximadamente estáticos.

Utilizando a informação de posição de cada dedo, encontrada a partir da equação 35 em cada quadro, foi possível construir as curvas de movimento dos dedos, representadas pelas Figuras 5.5 a 5.9.

Os valores de posicionamento nestas figuras são mostrados pela porcentagem de movimento em relação à máxima distância de cada dedo para o centro da mão, informação retirada no quadro que inicia o rastreamento, onde todos os dedos estão completamente estendidos.

As Figuras 5.6, 5.7 e 5.10 representam, respectivamente os dedos “1”, “2” e “5” e apresentam uma curva com maior amplitude e com a presença de um vale que indica o movimento de fechar estes dedos, como esperado.

As Figuras 5.8 e 5.9 apresentam uma curva sem valores abaixo de 100% aberto o que demostra que eles ficaram aproximadamente estáticos.

O aparecimento de valores maiores que 100% (que poderiam parecer impossíveis) são explicados pelo não controle total de que a mão esteja completamente aberta no início do rastreamento e também por possíveis variações das circunferências internas à mão.

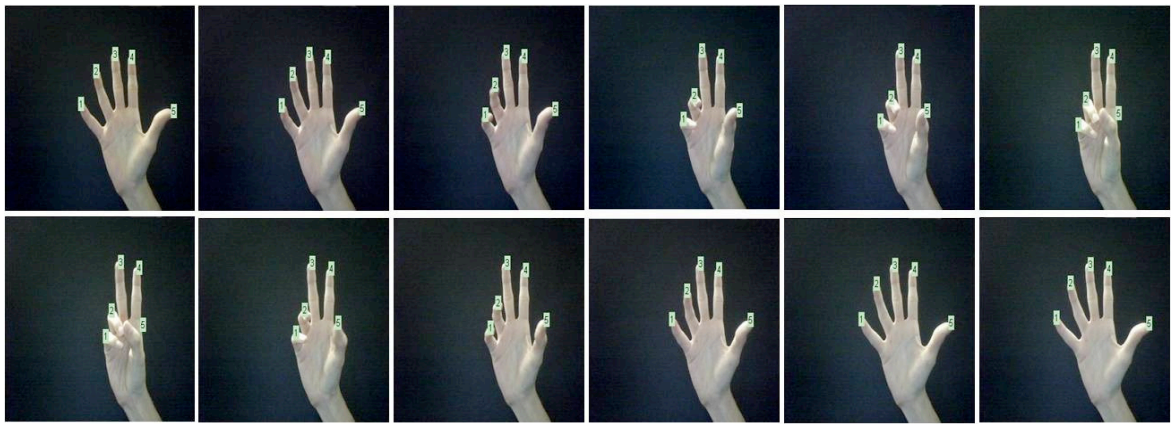

Figura 5.5 Exemplo de quadros de sequência de vídeo simulando movimento de indicar o número dois, utilizada para analisar o algoritmo de rastreamento.

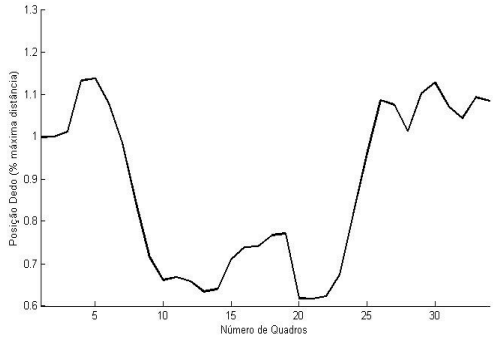

Figura 5.6 Curva do movimento do dedo mínimo em uma sequência de vídeo**.**

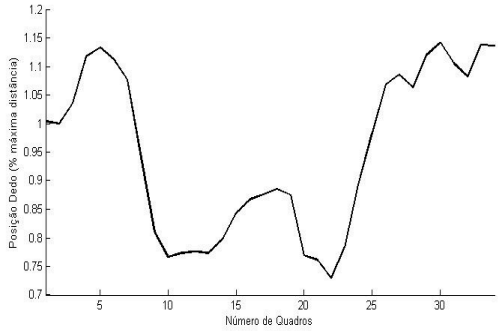

Figura 5.7 Curva do movimento do dedo anelar em uma sequência de vídeo

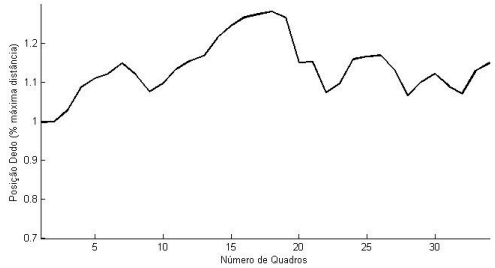

Figura 5.8 Curva do movimento do dedo médio em uma sequência de vídeo

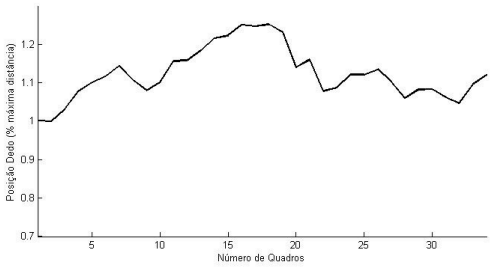

Figura 5.9 Curva do movimento do dedo indicador em uma sequência de vídeo

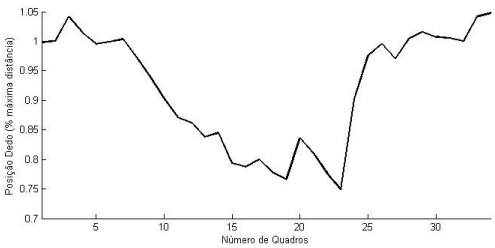

Figura 5.10 Curva do movimento do dedo polegar em uma sequência de vídeo

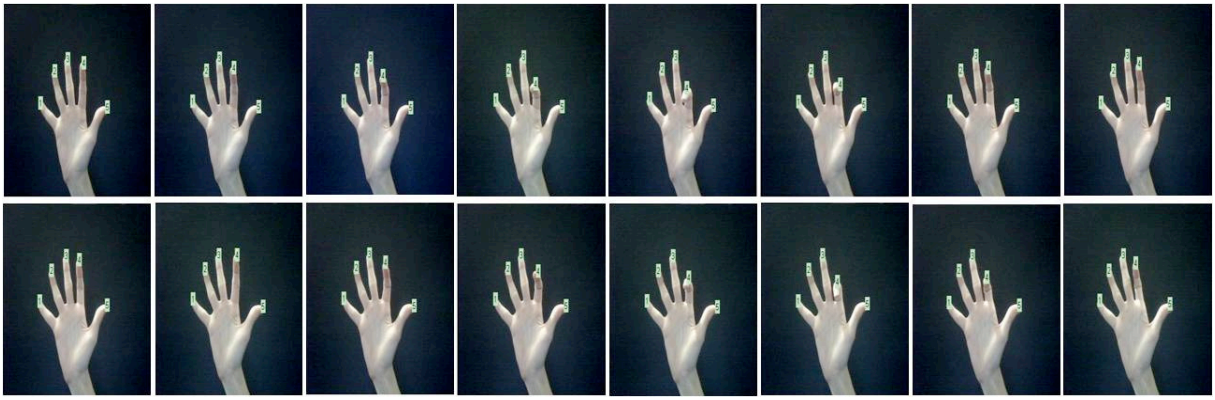

Figura 5.11 Exemplo de quadros de sequência de vídeo simulando movimento do dedo indicador, utilizada para estimar a velocidade do movimento.

Uma última análise é realizada para estimar a velocidade do movimento de um dedo. Para isto, foi realizado um vídeo contendo apenas movimento do dedo indicador e a partir de sua posição em cada quadro analisado foi construido a curva de seu movimento.

A Figura 5.10 apresenta alguns quadros deste vídeo para exemplificar o movimento. Esta curva é apresentada na Figura 5.11.

O vídeo utilizado para esta estimativa capturou imagens em uma taxa de 15 quadros por segundo e o algoritmo foi processado a cada dois quadros, portanto o tempo real para realizar o movimento será metade do calculado.

Observando esta curva do movimento percebe-se que o movimento de abaixar o dedo indicador ocorreu aproximadamente entre os quadros 7 e 16 por isso, aproximando esta curva neste intervalo a uma reta, como mostra a Figura 5.12, pode-se estimar o tempo em que o dedo fez o movimento, como mostrado na equação 38, onde Q_F representa o quadro onde o movimento cessou, Q_I o quadro onde o movimento iniciou e TQ a taxa de quadros capturados por segundo usados para construir o vídeo.

$$Tempo_{estimado} = \frac{(Q_F - Q_I) \cdot 2}{TQ} = 1{,}2s \tag{38}$$

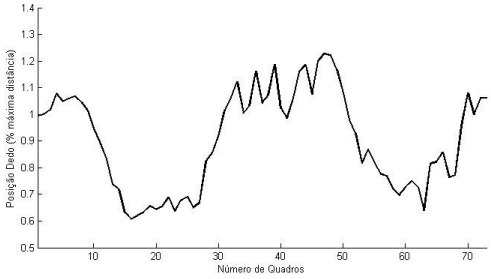

Figura 5.12 Curva do movimento do dedoindicador em uma sequência de vídeo

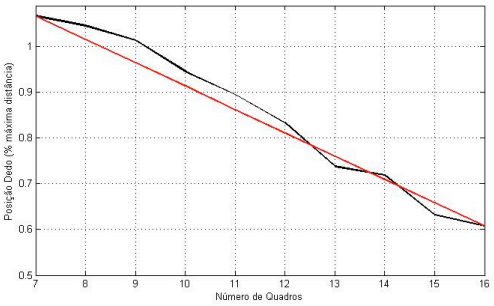

Figura 5.13 Aproximação por uma reta da curva de movimento de um dedo

5.3 CONCLUSÃO

Ao analisar a resposta de cada porção do método proposto, conclui-se que o método adotado para segmentação é limitado, considerando que o sistema utiliza um fundo simples preto para diminuir a sensibilidade por diferenças na iluminação ou por sombreamento, mas conseguiu responder de maneira eficiente para o sistema proposto.

O algoritmo de rastreamento funcionou muito bem na porção de reconhecimento e classificação das pontas de cada dedo, pois apenas nas imagens que continham oclusão entre os dedos ou com a palma é que o algoritmo de k-curvatura não conseguia encontrar as pontas de todos os dedos.

Em relação ao algoritmo de tratamento de oclusão, este funcionou consideravelmente bem nos casos em que uma oclusão não persistisse por mais que 10 segundos, portanto, consegue ser aplicado para movimentos menos complexos.

Os resultados obtidos graficamente pelas posições de cada dedo em cada quadro (Figuras de 5.5 a 5.9) a partir de uma sequência de video, demonstraram que é possivel prever os movimentos da mão pela análise da posição de cada dedo em relação ao seu centro. A análise destes movimentos rápidos (reconhecidos em alguns quadros separadamente) poderia ser aplicada em sistemas simples para prever movimentos da mão como um todo.

Assim, considerando-se o tempo de processamento do algoritmo, dentro de parâmetros compatíveis com a operação em tempo real, pode-se prever sua utilização na interface Humano-Computador por meio de comandos gestuais.

5.4 TRABALHOS FUTUROS

Embora o modelo de segmentação tenha sido robusto para as situações propostas do sistema, não apresenta bons resultados para os casos de variação brusca de iluminação e sombreamento. Esta característica é intrínsica da segmentação por subtração de fundo simples. Por consequência, pode ser abordado em projetos futuros o estudo e utilização de um algoritmo de segmentação que pudesse se adaptar dinamicamente às variações de iluminação, como por exemplo, algoritmos utilizando redes neurais.

Outra abordagem que se pode supor para continuar este trabalho, é a utilização do posicionamento dos dedos a cada quadro e interpretá-lo para prever seu movimento articulado. Isto poderia ser realizado de forma simples por uma calibração no sistema para que reconhecesse a cada diminuição, dada em porcentagem, do tamanho do dedo na imagem interpretasse uma posição angular de cada articulação.

REFERÊNCIAS*

BEN-ISRAEL, E. **Tracking of Humans Using Masked Histograms and Mean Shift:** Introductory Project in Computer Vision – Summary, [S.l.: s.n.], mar 2007. Disponível em: <http://www.scribd.com/doc/4020604/Tracking-of-Humans-Using-Masked-Histograms-and-Mean-Shift> . Acesso em: 15 de Fevereiro de 2010.

CARROLL, John M. **Encyclopedia entry on Human Computer Interaction (HCI)**. Disponível em: <http://www.interaction-design.org/encyclopedia/human_computer_interaction_hci.html>. Acesso em: 25 Maio 2010.

CARVALHO, J. O. F. O papel da interação humano-computador na inclusão digital. In: **Revista Transinformação**, V.15, N.3, edição especial, p.75-89, Campinas, 2003.

CASTLEMAN, K. R. **Digital Image Processing**, New Jersey:Prentice Hall, Englewood Cliffs, 1996.

CHAI, D.; NGAN, K.N. Face segmentation using skin-color map in video-phone applications, **IEEE Trans. Circuits Syst. Video Technol.,** v.9 n.4, 1999.

DE SOUZA, C. S.; LEITE, J. C.; PRATES, R.O.; BARBOSA, S.D.J. **Projeto de Interfaces de Usuário: Perspectivas Cognitiva e Semiótica**, Anais da Jornada de Atualização em Informática, XIX Congresso da Sociedade Brasileira de Computação, Rio de Janeiro, julho de 1999.

DUCA F.; FREDRIKSSON, J.; FJELD, M. Real-Time 3D Hand Interaction: Single Webcam Low-Cost Approach. In: **IEEE VR 2007 Workshop on Trends and issues in Tracking for Virtual Environments**, Gothenburg, Sweden, 2007.

* De acordo com:
ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6023**: informação e documentação: referências: elaboração. Rio de Janeiro, 2002.

FUJII, K.; SHIMAMURA , J.; ARAKAWA K.; ARIKAWA, T. Three-dimensional finger tracking using direct and reflected infrared images.In: **Conference on Human Factors in Computing Systems**, Ft. Lauderdale, Florida, USA, p. 848 – 849, 2002.

GEJGUS, P.; PLACEK, J.; SPERKA, M. Skin color segmentation method based on mixture of Gaussians and its application in Learning System for Finger Alphabet. In: **International Conference on Computer Systems And Technologies,** Rousse, Bulgaria, p. IIIA.1-IIIA.6.jun. 2004.

GOMEZ, G.; SANCHEZ, M.; SUCAR, L. On selecting an appropriate colour space for skin detection. In: **Mexican Int. Conf. on Artificial Intelligence**, Merida, Yucatan, Mexico, v.2313, p.70-79, 2002.

GONZAGA, A. **Kangûera:** Olho-Local Mão-Distante. Disponível em: <http://iris.sel.eesc.usp.br/weblab/default.html>. Acesso em: 22 maio 2010.

GONZALEZ, R. C.; WOODS, R. E. Digital image processing. New York: Addison-Wesley Publishing Company, Inc, 1987.

GONZALEZ, R. C; WOODS, R. E; EDDINS, S. L. **Digital Image Processing Using MATLAB,** Prentice Hall, 624 p., 2003.

KAKUMANU, P.; MAKROGIANNIS, S.; BOURBAKIS N. A survey of skin-color modeling and detection methods Pattern Recongnition Society. v. 40, n.. 3, p. 1106-1122, Mar 2007.

KÜHL, M. G.; SILVA, M. S. Nosso Hair: Sistema para simulação de coloração em tingimentos de cabelos. **Faculdade Mater Dei Revista de Informática**, Pato Branco, v.1, n., 2005, p.37-44.

LETESSIER, J.; BÉRARD, F. Visual Tracking of Bare Fingers for Interactive Surfaces. In: **ACM Symposium on User Interface Software and Technology (UIST)**, Santa Fe, New Mexico, USA, 2004.

LOPES, E. C., **Determinando a Posição e a Orientação da Mão Através de Imagens de Vídeo**, Porto Alegre, fevereiro de 2006.

MILSZTAJN F.; GEUS K. DE **Segmentação de Tecidos Cerebrais em Imagens Tridimensionais do** Cérebro. UFPR – Universidade Federal do Paraná, Curitiba, 2001.

NEVES S. C. M.; PELAES E. G. ESTUDO E IMPLEMENTAÇÃO DE TÉCNICAS DE SEGMENTAÇÃO DE IMAGENS. In: **Revista Científica da UFPA**, Belém, Pará, n. 02, outubro 2001.

PADILHA A. J. **Processamento e Análise de Imagem**, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, 1998. Disponível em: <http://paginas.fe.up.pt/~padilha/PAI/ficheiros/Cap4-ac.pdf >. Acesso em: 21 de abril de 2010.

PEER, P.; KOVAC, J.; SOLINA, F. Human skin colour clustering for face detection. In: **submitted to EUROCON 2003 – International Conference on Computer as a Tool**, v.2, p.144-148, 2003.

RIBEIRO, H. L. **Reconhecimento de gestos usando segmentação de imagens dinâmicas de mãos baseada no modelo de mistura de Gaussianas e cor de pele.** 2006. 144p**.** Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, SP, 2006.

SALDANHA M. F. S.; FREITAS, C. DA C. Segmentação de Imagens Digitais: Uma Revisão. In: **IX Workshop do Curso de Computação Aplicada - CAP Auditório Fernando de Mendonça**, LIT/INPE, outubro de 2009.

SAWANGSRI, T.; PATANAVIJIT, V., JITAPUNKUL, S. Face Segmentation Using Novel Skin-Color Map And Morphological Technique. In: **Proceedings of World Academy of Science**, Engineering and Technology, v.2, January 2005.

SOBOTTKA, K.; PITAS, I. Extraction of Facial Regions and Features using Color and Shape Information . In: **Proceedings of the International Conference Of Pattern Recognition (ICPR '96)**, 1996.

THE MATHWORKS Acelerating the pace of engineering and science. Disponível em: <http://www.mathworks.com/access/helpdesk/help/techdoc/>. Acesso em: 20 de novembro de 2009.

TRUYENQUE, M. A. Q. **Uma Aplicação de Visão Computacional que Utiliza Gestos da Mão para Interagir com o Computador**, Mestrado em Informática, PUC, Rio de Janeiro, 2005.

VEZHNEVETS V.; SAZONOV, V.; ANDREEVA, A. A Survey on Pixel-Based Skin Color Detection Techniques. In: **Proc. Graphicon-2003**, Graphics and Media Laboratory Faculty of Computational Mathematics and Cybernetics, Moscow State University, Moscow, Russia, 2003.

WANGENHEIM, A. V. Seminário Introdução à Visão Computacional, PPGCC – INE, UFSC. Disponível em: <http://www.inf.ufsc.br/~visao/>. Acesso em: 31 de janeiro de 2010.

YILMAZ, A.; JAVED, O.; SHAH, M. Object tracking: A survey. **ACM Comput. Surv**. v. *38*, n. 4, p. 13, 2006.

APÊNDICE A – Código fonte do algoritmo de rastreamento

```
function [ rowC columnC radiusC ] = HandCenter( imageSeg )
%HANDCENTER Encontra coordenadas do centro da mão, bem %como
seu raio.
%
%imageSeg - representa uma imagem binária segmentada
%rowC - representa coordenada em x  do centro da mão
%columnC - representa a coordenada em y do centro da mão
%radiusC - representa o raio da palma
%
%A função encontra a imagem de intensidades dada pela %transfomada
da distância euclidiana aplicada a imagem %segmentada e localiza as
coordenadas do centro da imagem %segmentada e o raio da
circunferêmcia interna à palma.
d=bwdist(~imageSeg);
maxRaio=max(d);
radiusC=max(maxRaio);
trasf_TDE=mat2gray(d);
imshow(~trasf_TDE);
auxmax=max(trasf_TDE);
maxdist=max(auxmax);
[cC,rC]=find(trasf_TDE==maxdist);
sizeC=size(cC,1);
sizeR=size(rC,1);
columnC=cC(uint16(sizeC/2));
rowC=rC(uint16(sizeR/2));
end
```

```
function [ Kptos ] = Kcurva( px,py,k,thetaref )
%KCURVADIST Encontra kcurva a partir da diferenca entre dois angulos.
% Pegar um vetor de pontos da curva (px,py) e percorre considerando k como o
intervalo para pelgar os outros dois pontos
% Determina a partir de thetaref os pontos candidatos e os pontos de picos
% da curva.
    sizeP=size(py);
    Kptos=[];
          Kcurva=[];
    m=1;
    j=0;
    for i=1:sizeP(1)
        if i<=k
            pkmenos=sizeP(1)+i-k;
        else
            pkmenos=i-k;
        end
        if i>sizeP(1)-k
            pkmais=k+i-sizeP(1);
        else
            pkmais=i+k;
        end
        v1=[px(pkmenos) py(pkmenos)]-[px(i) py(i)];
        v2=[px(pkmais) py(pkmais)]-[px(i) py(i)];
        theta=acosd(dot(v1,v2)/(norm(v1)*norm(v2)));
        if isempty(Kcurva)
           if theta<thetaref
               j=1;
               Kcurva(j,:)=[px(i) py(i) theta v1(1) v1(2)];
           end
        else
            if theta<thetaref
                j=j+1;
               Kcurva(j,:)=[px(i) py(i) theta v1(1) v1(2)];
            Else
%             Encontrar pontos candidatos
                indPto=find(Kcurva(:,3)==min(Kcurva(:,3)));
                sI=size(indPto);
                if sI(1)>1
                    sK=size(Kcurva);
                    indPto=uint16(sK(1)/2);
                end
                if Kcurva(indPto,4)>0
                    Kptos(m,:)=[Kcurva(indPto,1) Kcurva(indPto,2)];
                    m=m+1;
                end
                Kcurva=[];
            end
        end
    end
end
```

```
% main.m
main.m

% PRIMEIRA PARTE - AQUISIÇÃO, SEGMENTAÇÃO E FILTROS MORFOLÓGICOS

% VARIAVEIS PARA O TRACKING
w=1;
matriz=zeros();
tempo=zeros(1,180);
mm=1;
back=double(imread('backgrblack.jpg'));
% Cria video de entrada.
vid = videoinput('winvideo',1,'YUY2_352x288');
%Configura parâmetros do video
set(vid,'TriggerRepeat',Inf);
set(vid,'ReturnedColorSpace','rgb');
vid.FrameGrabInterval = 1;
vid_src = getselectedsource(vid);
set(vid_src,'Tag','motion detection setup');
figure;
% Começa a aquisição de imagem:
start(vid);
figh = figure('keypressfcn','set(gcbf,''userdata'',1)','userdata',0) ;
figure(figh) ;
while~get(figh,'userdata'),
    a frame_im = getsnapshot(vid);
    im=double(frame_im);
    imBack=((abs(im(:,:,1)-back(:,:,1))+ abs(imaux5(:,:,2)-back(:,:,2))+
abs(im(:,:,3)-back(:,:,3)))/3)>25;
    %FILTROS MORFOLOGICOS
    se=strel('disk',1,4);
    imfil=imdilate(imBack,se);
    imfil2=imerode(imfil,se);
    imfilt3=bwlabel(imfil2); % criar imagem de labels
    % MINPERPOLY NA REGIAO INTEIRA
    [Tlabel Tam]=bwlabel(imfilt3);
    Tarea=regionprops(Tlabel,'Area');
    Taux=[Tarea(1:Tam).Area];
    Tindice=find(Taux==max(Taux));
    for ma=1:Tam
        Tlbmenor= find(Tlabel(:)==ma);
        if ma~=Tindice
            Tlabel(Tlbmenor)=0;
        end
    end
    Ts=size(Tlabel);
    [Txp,Typ]=minperpoly(Tlabel,2);
    b3=connectpoly(Txp,Typ);
    Txmin=min(Txp(:));
    Tymin=min(Typ(:));
%   CONTORNO DA IMAGEM SEGMENTADA
    B3=bound2im(b3,Ts(1),Ts(2),Txmin,Tymin);
    xb=[b3(:,1)];
    yb=[b3(:,2)];
% PARTE TRACKING
(...)
```

```
(...)

% PARTE TRACKING
   % K-CURVA DO OBJETO DESTE FRAME
    Kptos = KcurvaDist(xb,yb,30,60);
    tamK=size(Kptos,1);
    if w==1
    % PARTE TRACKING
        if tamK<5 % TENHO SEMPRE QUE ENCONTRAR 5 PTOS INICIAIS
            fprintf('A encontrar ptos iniciais...\n');
            imshow(B3);
            if ~isempty(Kptos)
                figure(1)
                imshow(im);
                hold on
                for kapa=1:tamK(1)

text(Kptos(kapa,2),Kptos(kapa,1),num2str(kapa),'BackgroundColor',[.7 .9 .7]);
                end
                hold off
            end
        else % ENCONTREI CINCO PTOS INICIAIS
            if tamK==5 %garantir que tenha 5 ptos e mão aberta
                fprintf('IDENTIFICADAS AS PTAS DOS DEDOS\n');
                ptos=ordempto(Kptos);
                laba=bwlabel(B3);
                indilaba=find(laba(:)==1);
                sa  = regionprops(laba, 'FilledImage','BoundingBox');
                boundingbox=cat(1,sa.BoundingBox);
                imaFill=bwlabel(sa.FilledImage);
                [ c2 r2 raio2 ] = HandCenter( imaFill );
                r=double(c2+uint16(boundingbox(1)));
                c=double(r2+uint16(boundingbox(2)));
                for p=1:tamK,
                    matriz(p,1) = p; %nome do objeto
                    matriz(p,2:3) = ptos(p,1:2);% pta de um dedo
                    matriz(p,4) = r;%coordenada do centro da mão
                    matriz(p,5) = c;%coordenada do centro da mão
                    matriz(p,6) = raio2;%raio do centro da mão
                    v=[ptos(p,1) ptos(p,1)]-[r c];%vetor distancia ao centro
                    maxdist=norm(v)/raio2;% normaliza o movimento
                    matriz(p,7) = maxdist; %guardo valor máximo no primeiro frame
considerado
                end
                Objects{w}=matriz;%objetos do primeiro frame
                figure(1);
                imshow(im);
                hold on
                for t=1:size(Objects{w},1)

text(Objects{w}(t,3),Objects{w}(t,2),num2str(t),'BackgroundColor',[.7 .9 .7]);
                end
                hold off
                w=w+1;
            end
        end
    else %% A PARTIR DO SEGUNDO FRAME
(...)
```

```
(...)
    else %% A PARTIR DO SEGUNDO FRAME
        matriz_ant=matriz;
        x_ant=matriz_ant(:,2);
        y_ant=matriz_ant(:,3);
        matriz=zeros(5,7);
        laba=bwlabel(B3);
        indilaba=find(laba(:)==1);
        sa  = regionprops(laba, 'FilledImage','BoundingBox');
        boundingbox=cat(1,sa.BoundingBox);
        imaFill=bwlabel(sa.FilledImage);
        [ c2 r2 raio2 ] = HandCenter( imaFill );
        r=double(c2+uint16(boundingbox(1)));
        c=double(r2+uint16(boundingbox(2)));
        %ENCONTRAR EQUIVALENCIA ENTRE OS kPTOS - tracking dos ptos de máximos
encontrados
        if tamK>0
            for z=1:tamK
                for q=1:size(x_ant,1)
                    distk(q)=norm([Kptos(z,1) Kptos(z,2)]-[x_ant(q) y_ant(q)]);
                end
                mindistK=min(distk(:));
                indh=find(distk(:)==mindistK);
                matriz(indh,1)=matriz_ant(indh,1);
                x_now=Kptos(z,1);
                y_now=Kptos(z,2);
                v=[x_now y_now]-[r c];
                dist=norm(v)/raio2;
                %PREENCHE A MATRIZ DO OBJETO DESTE FRAME
                matriz(indh,2)=x_now;
                matriz(indh,3)=y_now;
                matriz(indh,4) = r;
                matriz(indh,5) = c;
                matriz(indh,6) = raio2;
                matriz(indh,7)=dist;% DISTANCIA DO PTO AO CENTRO NORMALIZADA
                distk=zeros();
            end
        end
```

```
(...)
    %   PARTE TRATAMENTO DE OCLUSÃO  - para ptos de máximos nao encontrados

        for h=1:size(x_ant,1)
            if matriz(h,1)==0
                for g=1:size(xb,1)
                    dista(g)=norm([x_ant(h) y_ant(h)]-[xb(g) yb(g)]);
                end
                matriz(h,1)=h;
                mindist_now=min(dista(:));
                indg=find(dista(:)==mindist_now);

                sizeind=size(indg,1);

                if sizeind>1
                    indg=indg(uint16(sizeind/2));
                end
                x_now=xb(indg);
                y_now=yb(indg);
                v=[x_now y_now]-[r c];
                dist=norm(v)/raio2;

    %           PREENCHE LINHAS DA MATRIZ QUE AINDA NAO TINHAM SIDO
    %           IDENTIFICADAS
                matriz(h,2)=x_now;
                matriz(h,3)=y_now;
                matriz(h,4) = r;
                matriz(h,5) = c;
                matriz(h,6) = raio2;
                matriz(h,7)=dist;
                dista=zeros();
            end
        end
            Objects{w}=matriz;
            figure(1);
            imshow(im);
            hold on
            for t=1:size(Objects{w-1},1)

text(Objects{w}(t,3),Objects{w}(t,2),num2str(Objects{w}(t,1)),'BackgroundColor',[.
7 .9 .7]);
            end
            hold off
            w=w+1;
    end
    mm=mm+1;
end
```