UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

SUELLEN DE ARAUJO CADUDA DA SILVA MOTTA

RECONHECIMENTO DE GESTOS EM LIBRAS ATRAVÉS DO PROCESSAMENTO DE IMAGENS DE VÍDEO UTILIZANDO MODELOS OCULTOS DE MARKOV

São Luís
2012

SUELLEN DE ARAUJO CADUDA DA SILVA MOTTA

RECONHECIMENTO DE GESTOS EM LIBRAS ATRAVÉS DO PROCESSAMENTO DE IMAGENS DE VÍDEO UTILIZANDO MODELOS OCULTOS DE MARKOV

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Aristófanes Corrêa Silva

São Luís
2012

Motta, Suellen de Araujo Caduda da Silva.

Reconhecimento de gestos em Libras através do processamento de imagens de vídeo utilizando Modelos Ocultos de Markov / Suellen de Araujo Caduda da Silva Motta. – São Luís, 2012.

52 f.

Impresso por computador (Fotocópia).

Orientador: Aristófanes Corrêa Silva.

Monografia (Graduação) – Universidade Federal do Maranhão, Curso de Ciência da Computação, 2012.

1. Computação – Modelagem de sistema – Reconhecimento – Libras 2. Gestos – Reconhecimento 3. Modelos Ocultos – Markov I. Título

CDU 004.414.23: 376

SUELLEN DE ARAUJO CADUDA DA SILVA MOTTA

RECONHECIMENTO DE GESTOS EM LIBRAS ATRAVÉS DO PROCESSAMENTO DE IMAGENS DE VÍDEO UTILIZANDO MODELOS OCULTOS DE MARKOV

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, para obtenção do grau de Bacharel em Ciência da Computação.

Aprovada em: 01 / 08 / 2012

BANCA EXAMINADORA

Prof. Aristófanes Corrêa Silva (Orientador)
Doutor em Informática
Universidade Federal do Maranhão

Prof. Anselmo Cardoso de Paiva
Doutor em Informática
Universidade Federal do Maranhão

Prof. Carlos Eduardo Portela Serra de Castro
Mestre em Informática
Universidade Federal do Maranhão

AGRADECIMENTOS

Agradeço primeiramente à minha família, que ofereceu todo o suporte necessário para o meu crescimento pessoal e profissional.

Ao namorado Igor, obrigada pelo carinho e dedicação ao longo desses anos, e pela compreensão, especialmente nas últimas semanas.

Aos professores do curso de Ciência da Computação, em especial ao meu orientador Aristófanes Corrêa Silva pela paciência, dedicação e conhecimentos passados.

Aos amigos do curso agradeço pelos quatro anos e meio de convivência. Foi bastante gratificante aprender, me divertir e "sofrer" com vocês todos os dias. Aos amigos Pedro, Pablo, Samuel e Lucena, um agradecimento especial pela contribuição nesta monografia.

Por fim, agradeço a todos que direta ou indiretamente contribuíram para a realização deste trabalho.

“A razão é o passo, o aumento da ciência o caminho, e o benefício da humanidade é o fim”.

(Thomas Hobbes)

RESUMO

Segundo o censo do IBGE de 2010, o Brasil possui mais de 2,1 milhões de pessoas com grande dificuldade auditiva, das quais 344,2 mil são completamente surdas. Muitas delas comunicam-se entre si através da Língua Brasileira de Sinais (Libras), que desde 2002 é reconhecida como a Língua Oficial da Pessoa Surda no país. Incluir essas pessoas na sociedade deve ser um objetivo buscado por todos os brasileiros. Neste sentido, qualquer ferramenta computacional que auxilie na comunicação dos surdos com os falantes da língua portuguesa, ou na interação mais natural dessas pessoas com sistemas computacionais, é bem-vinda. O campo do reconhecimento de gestos através de técnicas baseadas em visão computacional e processamento de imagens é objeto de pesquisa há muitos anos. Inúmeros trabalhos bem-sucedidos modelam movimentos humanos utilizando os Modelos Ocultos de Markov, técnica que já provou ser de bastante eficiência no reconhecimento da fala, e recentemente vem provando também ser uma poderosa ferramenta para o reconhecimento de gestos. Poucos trabalhos, no entanto, aplicam essas técnicas no reconhecimento das línguas de sinais, especialmente a Libras. Assim, o principal objetivo deste trabalho é a elaboração de uma metodologia eficiente para reconhecimento de algumas palavras em Libras, o que inclui o desenvolvimento de dois módulos: um para o rastreamento das mãos em vídeos, e o outro para a classificação dos gestos, utilizando os Modelos Ocultos de Markov. A taxa de acerto de 92,5% obtida nos testes realizados sobre as bases de treino e teste comprovam a eficácia da metodologia proposta e indicam a possibilidade de extensão do trabalho, no intuito de abranger maior número de gestos e possivelmente sentenças.

Palavras-chave: Libras. Reconhecimento de gestos. Modelos Ocultos de Markov.

ABSTRACT

According to the IBGE census of 2010, Brazil has more than 2.1 million people with hard hearing. From this group, 344.2 thousand are completely deaf. Many of them communicate with each other using the Brazilian Sign Language (Língua Brasileira de Sinais – Libras), which is recognized as the Official Deaf People Language in the country since 2002. To include these people into the society must be a goal sought by all the Brazilians. In this sense, any computational tool which is able to assist in communicating between deaf people and Portuguese speakers, or in improving more natural interaction of these people with computer systems, is welcome. Field of gesture recognition using computer vision and image processing based techniques has been subject of research for many years. Several studies successfully model human motions using the Hidden Markov Models, a technique which has proven to be very efficient in speech recognition and recently has also been proving to be a powerful tool for gesture recognizing. Few studies, however, apply these techniques in the recognition of sign languages, especially the Libras. Thus, this project mainly objectives the development of an efficient methodology for recognition of some words in Libras, which includes the development of two modules: one for tracking hands on videos, and other for classification of gestures, using the Hidden Markov Models. The accuracy of 92.5% obtained in tests conducted on training and testing bases prove the efficiency of the methodology and indicate the possibility of an extension work, in order to cover larger number of gestures and possibly sentences.

Keywords: Libras. Gesture recognition. Hidden Markov Models.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A pesquisa no campo de reconhecimento de gestos possui várias motivações. Todas elas estão relacionadas ao objetivo de melhorar a interface entre homem e máquina (TANGUARY Jr., 1995). Se um computador é capaz de reconhecer gestos humanos, uma infinidade de aplicações úteis à sociedade pode ser desenvolvida, por exemplo, na área de ambientes virtuais, sistemas de segurança, ferramentas de auxílio ao médico, e tradução de línguas de sinais.

"As línguas de sinais são linguagens naturais utilizadas por comunidades de pessoas [...] surdas e/ou mudas, por todo o mundo, cujas formas consistem em sequências de movimentos e configurações executados pelas partes do corpo" (DIAS, SOUZA, & PISTORI, 2006). A comunicação através dessas línguas envolve movimentos complexos que utilizam em geral mãos, braços, ombros e expressões faciais.

A Língua Brasileira de Sinais, ou simplesmente Libras, foi reconhecida em 2002 pela Nação Brasileira como a Língua Oficial da Pessoa Surda através das Leis nº 10.436, de 24-4-2002 e nº 10.098, de 19-12-2002. Trata-se de um sistema linguístico legítimo, de modalidade gesto-visual e com estrutura gramatical independente da língua portuguesa (AZEREDO, 2006).

No Brasil, existem mais de 2,1 milhões de pessoas com grande dificuldade auditiva, das quais 344,2 mil são completamente surdas (IBGE, 2010). Há uma escassez de dados estatísticos sobre a quantidade de pessoas que usam Libras. No entanto, existe um enorme incentivo na divulgação da língua entre a comunidade surda e mesmo entre os falantes do português, afinal, quanto mais pessoas utilizarem Libras, maior a interação dos surdos-mudos com a sociedade.

A análise dos movimentos exige primeiramente o monitoramento das posições e configurações das partes do corpo ao longo do tempo. Uma abordagem existente para executar tal tarefa é a utilização de instrumentos de apoio, tais como luvas especiais ou sensores que permitam alta precisão no rastreamento. O uso desses equipamentos para interagir com o sistema caracteriza uma técnica do tipo *intrusiva* e causa certa rejeição por parte dos usuários, devido ao desconforto que ela proporciona.

O ideal é que os gestos sejam realizados como no dia a dia, da maneira mais natural possível. Técnicas *não intrusivas*, baseadas somente em visão computacional, são mais aceitas por permitirem ao usuário maior liberdade de movimento. Por outro lado, esta

abordagem geralmente exige custos computacionais mais altos, devido à maior dificuldade no processamento das imagens em vídeo para detecção das partes do corpo.

Este trabalho se encaixa na segunda abordagem, porém com algumas particularidades. Primeiramente, por uma questão de simplificação, somente mãos e braços[1] serão considerados na execução dos gestos, deixando de fora os movimentos com os ombros e as expressões faciais. Os trabalhos de (SIOLA, 2010), (SOUZA, DIAS, & PISTORI, 2007) e (STARNER, WEAVER, & PENTLAND, 1998) mostram que, com apenas estes elementos, já é possível reconhecer um amplo conjunto de gestos em Libras. Além disso, a metodologia exige que mãos e face sejam os únicos objetos da cor da pele presentes na imagem.

Mais do que o monitoramento das partes do corpo, são necessários outros procedimentos para a classificação dos gestos. É importante a escolha de um modelo computacional apropriado que bem represente a estrutura do problema e permita a melhor distinção possível entre os gestos executados em Libras.

Os Modelos Ocultos de Markov são estruturas utilizadas na modelagem de problemas com variações temporais. Há vários anos, os HMMs, como são conhecidos (*Hidden Markov Models*), se mostraram úteis no reconhecimento da fala (JUANG & RABINER, 1991) e posteriormente da escrita (SHU, 1997), e recentemente vem provando também ser uma poderosa ferramenta para o reconhecimento de gestos. (TANGUARY Jr., 1995) foi um dos primeiros a utilizá-lo para essa finalidade; (STARNER, WEAVER, & PENTLAND, 1998) aplicaram os HMMs no reconhecimento dos gestos da ASL (*American Sign Language* – Língua Americana de Sinais) sob duas abordagens: uma utilizando uma câmera de mesa e outra uma câmera acoplada a um boné para permitir visões de cima; (SOUZA, DIAS, & PISTORI, 2007) reconheceram gestos em Libras.

Dada a ampla variedade de trabalhos em reconhecimento de gestos que utilizam os HMMs, esta foi a modelagem selecionada para a classificação dos gestos em Libras.

1.1 Objetivos

Este trabalho tem como objetivo geral o desenvolvimento de uma metodologia para reconhecimento de gestos em Libras pré-determinados e separadamente executados, filmados com uma câmera comum, de forma não intrusiva.

Os objetivos específicos são listados:

[1] Os braços, propriamente, não serão analisados. Mas a posição das mãos somente já é capaz de fornecer certa informação sobre a sua configuração em dado momento.

a) Detectar as mãos e acompanhar seus movimentos em imagens de vídeo;

b) Criar uma pequena base de vídeos de gestos de Libras previamente selecionados para treinamento e teste;

c) Elaborar e testar modelos (HMMs) eficientes para o reconhecimento automático dos gestos filmados.

1.2 Organização do Trabalho

Este capítulo apresentou a motivação e os objetivos de desenvolver um sistema de reconhecimento de gestos em Libras e listou os principais trabalhos relacionados. Uma breve descrição sobre a Língua Brasileira de Sinais é apresentada no Capítulo 2.

O Capítulo 3 traz a base teórica sobre as técnicas aplicadas aqui. Para o entendimento da metodologia proposta, o leitor deve ter um conhecimento básico sobre o Processamento de Imagens. A Seção 3.1.1 traz alguns conceitos básicos. Outros assuntos mais amplos, como a morfologia matemática e os modelos de cor são abordados na Seção 3.1.2 e na Seção 3.1.3, respectivamente. Um entendimento sobre os modelos de cor é necessário para se compreender o processo de segmentação da cor da pele, cuja teoria é dada em 3.2.

Para a detecção da face, a técnica utilizada foi o algoritmo proposto por (VIOLA & JONES, 2004), cuja explicação é apresentada na Seção 3.3.

Os Modelos Ocultos de Markov são brevemente discutidos na Seção 3.4.

O Capítulo 4 apresenta o desenvolvimento da metodologia, com a aplicação das técnicas descritas no Capítulo 2. Nele está descrito de forma mais detalhada o procedimento realizado para detecção e rastreamento das mãos – Seção 4.1 – e o reconhecimento dos gestos em Libras – Seção 4.2.

Os resultados obtidos nas duas etapas, rastreamento e reconhecimento, são detalhados no Capítulo 5.

O Capítulo 6 traz uma conclusão, resumindo o que foi proposto, propondo melhorias e sugerindo a continuação em trabalhos futuros.

2 A LÍNGUA BRASILEIRA DE SINAIS

As línguas de sinais, de uma forma geral, são compostas por gestos realizados com o corpo. Esses gestos podem ser analisados segundo vários aspectos. O primeiro deles é a configuração das mãos: a forma em que o intérprete posiciona os dedos e orienta a mão. Existem 46 configurações diferentes. (DIAS, SOUZA, & PISTORI, 2006) As mais conhecidas são as que representam as letras do alfabeto latino (Figura 1).

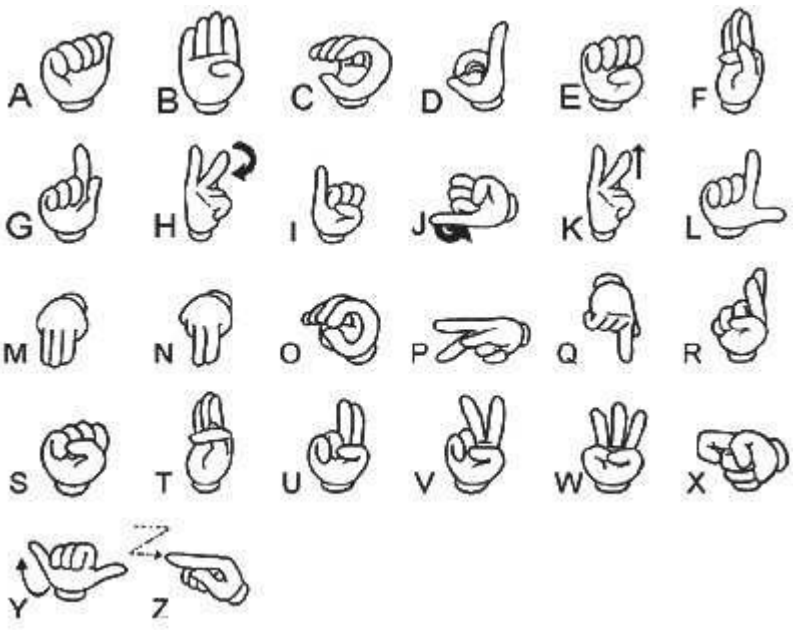


Figura 1 – Alfabeto em Libras. Fonte: (Site: Curso de Libras Gratuito, 2009)

Quando pensam em Libras, muitas pessoas associam a língua a estes símbolos, e imaginam que as palavras são formadas por execuções sequenciais destes. De fato, esta *soletração* pode ser feita, caso o intérprete ache necessário. Entretanto, essas e outras configurações das mãos são mais comumente empregadas durante as execuções dos *sinais* que representam as palavras, e não para compor a sua estrutura letra a letra, o que seria bem mais demorado e impediria que as línguas de sinais fossem equivalentes ou muitas vezes até melhores que a língua falada, no quesito velocidade.

Os sinais são a estrutura léxica dos gestos. Um sinal representa uma palavra, ou expressão, e é composto por uma sequência de posturas. Uma postura é uma configuração estática, determinada principalmente pela configuração das mãos, pelos seus *pontos de articulação*, e pela expressão facial. O ponto de articulação se refere ao ponto no espaço em que está a mão em determinado momento, muitas vezes tocando outra parte do corpo. (Site: Curso de Libras Gratuito, 2009)

Outro aspecto, o *movimento,* se refere às sequências de posturas. Alguns sinais são estáticos, ou seja, possuem uma única postura. Outros são compostos por uma sequência delas, intercalados por posições de transição entre uma e outra. A Figura 2 exibe as posturas principais do gesto MAU.

Figura 2 – Execução do gesto MAU. Fonte: (DIAS, SOUZA, & PISTORI, 2006)

Expressões faciais agregam ainda mais significado, e principalmente, mais naturalidade, aos gestos em Libras. Demonstrando suas emoções durante os movimentos, o intérprete dita a entonação de cada frase executada. Observe, na Figura 2, a força da expressão facial no significado da palavra dita. O movimento de inflar as bochechas ou o de pender a cabeça também faz parte de alguns gestos.

A *direção* dos movimentos também caracteriza os sinais. Em um diálogo, a ideia desenvolvida pode depender da direção em que as mãos se movimentam. Os verbos IR e VIR, por exemplo, opõem-se, no aspecto da direcionalidade. (Site: O que é Libras, 2000)

Atribui-se o status de língua à Libras e a todas as outras línguas de sinais, por elas também serem compostas pelos níveis linguísticos: fonológico, morfológico, sintático e semântico. O que as difere é somente o espaço realizado para a comunicação, que no caso dos sinais, é o espaço visual. Libras possui suas próprias estruturas gramaticais, independente da língua portuguesa, e também é diferenciada, de região para região. Há dialetos e sotaques. E mais, o seu poder de expressão é tão poderoso quanto o das línguas faladas. “Seus usuários podem discutir filosofia, política, e até mesmo produzir poemas e peças teatrais” (Site: O que é Libras, 2000).

3 FUNDAMENTAÇÃO TEÓRICA

Para melhor entendimento da metodologia apresentada, uma introdução sobre as técnicas utilizadas é apresentada neste capítulo.

A metodologia envolve principalmente conhecimentos na área de processamento de imagens – conceitos básicos de imagens; modelos de cores; segmentação da pele; detecção da face; rastreamento de regiões – e de aprendizado de máquina – Modelos Ocultos de Markov – para o reconhecimento dos gestos em Libras.

3.1 Fundamentos do Processamento de Imagens Digitais

Esta seção apresenta conceitos sobre as técnicas dentro da área de Processamento de Imagens que serão utilizadas na fase de rastreamento das mãos.

3.1.1 Conceitos Básicos

Uma imagem digital é definida como uma função bidimensional *f(x, y)*, em que *x* e *y* são coordenadas finitas e discretas de um ponto em um plano cartesiano e o valor de *f* é denominado intensidade desse ponto, e também é um valor finito e discreto (GONZALEZ & WOODS, 2010).

Uma matriz M_{nxm} é geralmente utilizada para representar uma imagem. Neste caso, as *n* linhas representam os *n* valores possíveis de *y* (ou a altura da imagem) e as *m* colunas os *m* valores possíveis de *x* (ou a largura da imagem). O valor armazenado em m_{ij} é o valor de *f(j, i)*, ou a intensidade daquilo que denominamos *pixel* (elemento pictórico).

Um *pixel* é o menor elemento de uma imagem. Possui, como já explicado, posição e valor de intensidade. Este valor de intensidade representa o brilho, e pode variar conforme o número de *bits* definido para representá-lo – o que se denomina profundidade. Em uma imagem de profundidade oito, comumente utilizada, os *pixels* assumem $2^8 = 256$ valores possíveis.

Um único valor de intensidade – ou um canal – é suficiente para representar a cor de um *pixel*, quando se trata de uma imagem em escala de cinza. Neste tipo de imagem, considerando uma profundidade de oito *bits*, a cor que cada *pixel* pode possuir, inicia na cor preta – intensidade 0 (zero) – e varia gradativamente até o branco, representado pelo valor 255. Entretanto, para imagens coloridas, mais canais são necessários, e neste caso, um modelo para representar as cores é exigido. O modelo mais comum é o RGB (*red, green, blue* –

vermelho, verde, azul), em que as cores de cada *pixel* são representadas por três canais, sendo que cada canal representa a intensidade das cores primárias vermelho, verde e azul. Este modelo e outros existentes para a representação das cores em imagens são descritos em mais detalhes na Seção 3.1.3.

Por último, se faz útil apresentar nesta seção o conceito de segmentação e de imagens binárias. Segmentar uma imagem significa separar um objeto de interesse do restante da imagem. O resultado desta atividade é normalmente uma imagem binária – imagem em duas cores, em que uma representa o fundo e a outra o objeto de interesse. Existem inúmeros tipos de segmentação: alguns baseados na forma, outros na textura, outros em limiares de intensidade, mas apresentar todos eles está além do escopo deste trabalho. Para maior aprofundamento neste campo, e também em outros na área do processamento digital de imagens, sugere-se a leitura de (GONZALEZ & WOODS, 2010). Neste trabalho, a segmentação de imagens é do tipo segmentação por limiar, e é utilizada para separar as áreas de pele nas imagens, conforme descrito na Seção 3.2.

3.1.2 Morfologia Matemática

A teoria dos conjuntos se torna uma ferramenta muito poderosa no processamento de imagens digitais, especialmente quando o campo em questão é o da morfologia matemática. “A palavra morfologia geralmente denota um ramo da biologia que lida com a forma e a estrutura dos animais e das plantas” (GONZALEZ & WOODS, 2010). No Processamento de Imagens, a morfologia matemática lidará com a forma de objetos de interesse, os agrupamentos de *pixels*, com o objetivo de descrever seu comportamento, através da análise de fronteiras, esqueletos, fecho convexo, etc. A morfologia também é utilizada, e é o caso deste trabalho, no pré ou pós-processamento de imagens, para a remoção de ruídos.

A linguagem da morfologia matemática é a teoria dos conjuntos, os quais são os objetos da imagem. Em uma imagem binária, em que o objeto está em branco, por exemplo, o conjunto é formado por todos os *pixels* pertencentes a esse objeto, e cada *pixel* é um elemento representado pela sua coordenada *(x,y).*

A morfologia matemática é um campo bastante amplo, e aqui são descritas somente as operações mais utilizadas: erosão e dilatação. O que as duas operações tem em comum é o chamado “elemento estruturante”. Um elemento estruturante é um conjunto, ou uma subimagem, que será convoluída sobre a imagem principal. Ele pode assumir várias

formas, dependendo do objetivo da operação. Alguns exemplos de elementos estruturantes são mostrados na Figura 3.

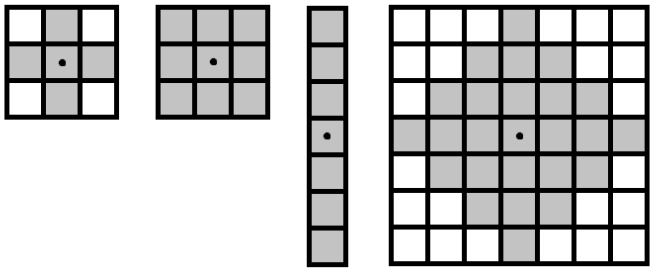
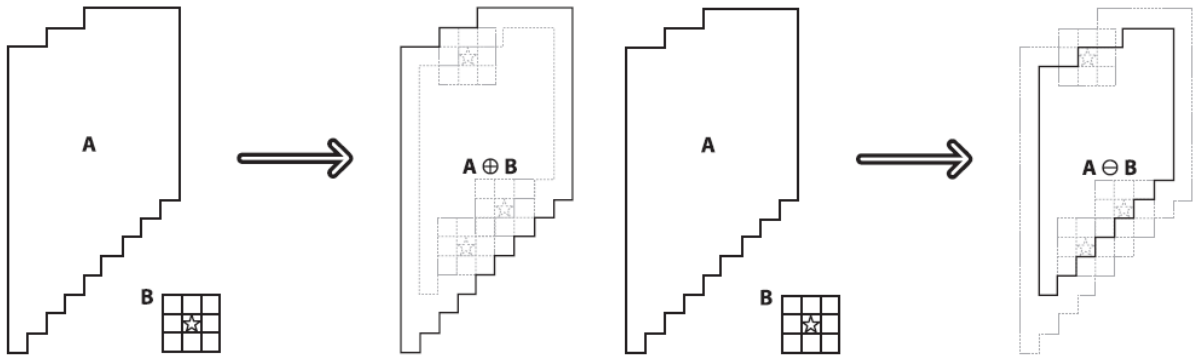


Figura 3 – Elementos estruturantes.

A chamada convolução é a passagem de uma máscara, um conjunto de *pixels*, sobre uma imagem. É como se se posicionasse ordenadamente a máscara por cima da imagem de todas as formas possíveis, e em cada posicionamento, operações entre os *pixels* sobrepostos são realizadas. No caso da erosão e da dilatação, essas operações são respectivamente as operações lógicas *OU* e *E*, e a máscara é o elemento estruturante. A Figura 4 ilustra o funcionamento e o resultado das operações. O conjunto *A* é o objeto na imagem original e o conjunto *B* é um elemento estruturante quadrado de tamanho 3x3.



(a) (b)

Figura 4 – Operações de (a) dilatação e de (b) erosão. Fonte: (BRADSKI & KAEHLER, 2008, pp. 116-117).

A operação de erosão é formalmente definida por:

$$A \ominus B = \{z|(B)_z \subseteq A\} \tag{1}$$

E a dilatação é formalmente definida por:

$$A \oplus B = \{z|\hat{B}_z \cap A \neq \emptyset\} \tag{2}$$

O efeito causado pela erosão é a diminuição ou o afinamento dos objetos. Isso é extremamente útil para se remover determinados componentes da imagem. O tamanho e a forma do elemento estruturante devem ser bem escolhidos, pois são determinantes na geração da nova forma do objeto.

A dilatação, por sua vez, proporciona o aumento dos objetos e é especialmente utilizada para preencher "buracos" na imagem. E, da mesma forma que a erosão, seu resultado varia conforme o tamanho e a forma do elemento estruturante.

Efeitos melhores podem ser obtidos realizando essas operações repetidas vezes, ou ainda fazendo combinações entre elas.

3.1.3 Modelos de Cores

Representar e reproduzir as cores vistas no mundo real não é uma tarefa fácil. Os modelos de cores surgiram para especificar as cores de uma forma padronizada e comumente aceita. Mais especificamente, um modelo de cor é a definição de um sistema de coordenadas e, dentro dele, de um subespaço, no qual cada ponto representa uma única cor (GONZALEZ & WOODS, 2010).

Cada modelo de cor é criado visando representar as cores para uma funcionalidade específica. Muitos são voltados para *hardware*. O modelo RGB (*red, green, blue* – vermelho, verde, azul), por exemplo, é largamente utilizado para representar a cor em monitores coloridos e em câmeras de vídeos e o modelo CMY (*cyan, magenta, yellow* – ciano, magenta, amarelo) para a impressão colorida.

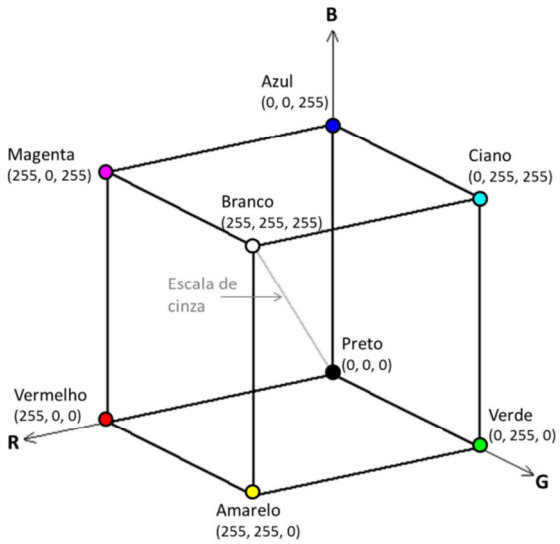
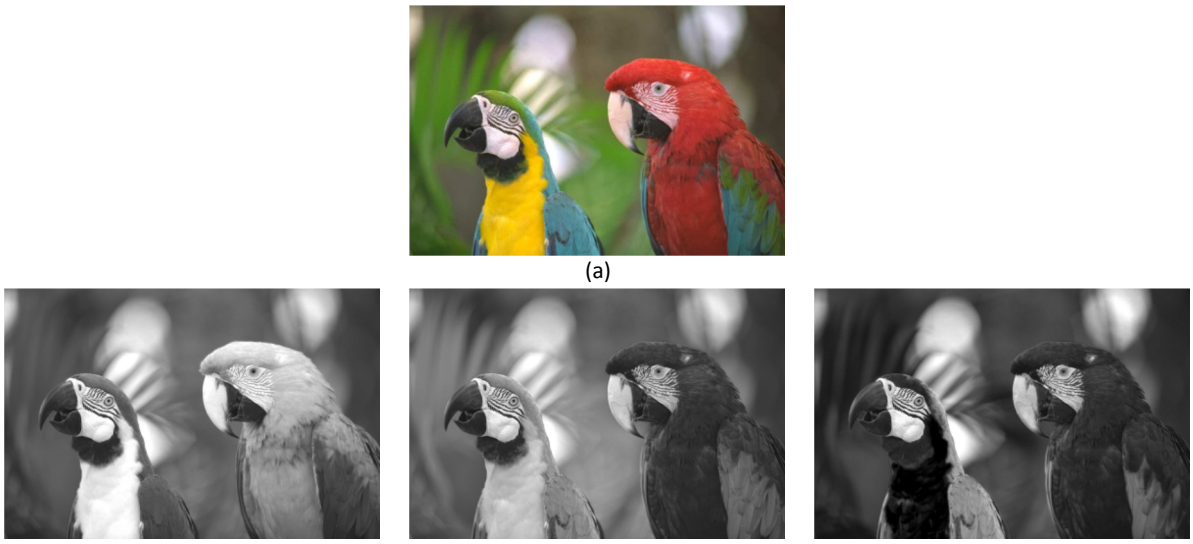


Figura 5 – Representação do modelo RGB no formato de cubo.

O modelo RGB especifica uma cor como um elemento formado pelas cores de luz primárias vermelha, verde e azul e pode ser representado por um cubo em um plano cartesiano. A Figura 5 exibe essa representação. O preto está no vértice sobre a origem do sistema e o branco no vértice mais distante. Todos os pontos pertencentes ao segmento de reta que une esses vértices equivalem a tons de cinza. Dos seis vértices restantes, os três sobre os eixos representam as cores primárias e os demais, as cores secundárias.

Em uma imagem colorida sob o formato RGB, cada *pixel* é representado por três canais, um para a cor vermelha, outro para a verde e outro para a azul. Utilizando-se uma profundidade 8, cada canal então possui $2^8 = 256$ valores possíveis. A Figura 6 exibe uma imagem representada no sistema RGB, e os seus componentes individuais.



(a) (b) (c) (d)

Figura 6 – Representação das cores no modelo RGB.
(a) Todos os componentes. (b) Componente R. (c) Componente G. (d) Componente B.

O espaço RGB pode ser normalizado para torná-lo invariante às mudanças de orientação das fontes de luz em superfícies opacas, quando ignorada a luz ambiente. O modelo RGB normalizado é definido pelas componentes *r, g* e *b*, dadas pelas equações abaixo. A normalização diminui a dependência dos componentes *r* e *g* ao brilho do espaço de cores RGB. (RIBEIRO, 2006)

$$r = \frac{R}{R + G + B} \tag{3}$$

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

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

O modelo HSI – ou HSV ou HSL – (*hue, saturation, intensity/value/lightness* – matiz, saturação, intensidade/valor/luminosidade) se aproxima mais da forma como os seres humanos descrevem e interpretam as cores. "Não pensamos em imagens coloridas como compostas de três imagens primárias que se combinam para formar uma única imagem" (GONZALEZ & WOODS, 2010). Matiz é um elemento que descreve uma cor pura, a cor dominante de uma área. Saturação é uma medida do grau de diluição de uma cor pura por luz branca. E a intensidade é um componente que representa a quantidade de luz percebida.. (GONZALEZ & WOODS, 2010). O modelo HSI é representado por um cone:

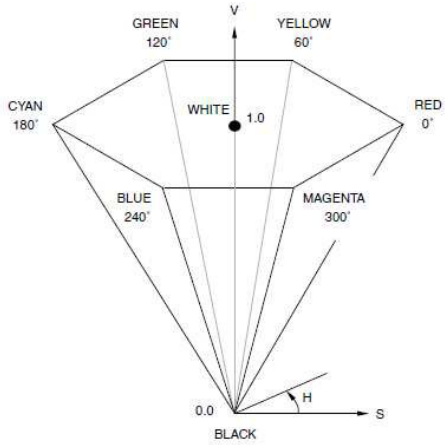
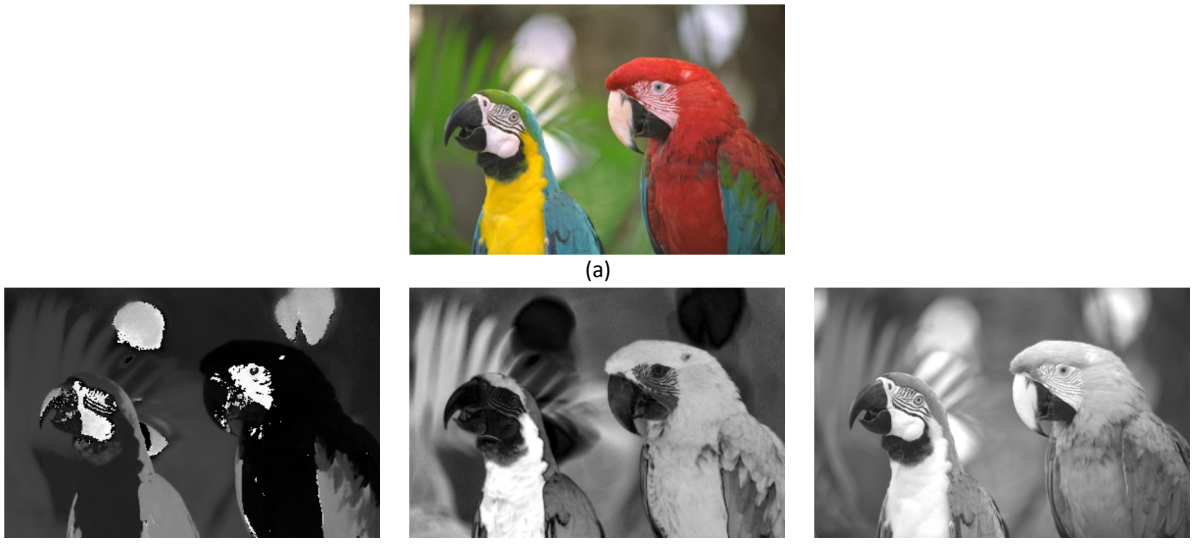


Figura 7 – Representação do modelo HSI em forma de cone. Fonte: (JACK, 2007, p. 39)

A Figura 8 mostra a mesma imagem em RGB, mas agora as imagens abaixo dela exibem os componentes matiz (H), saturação (S) e intensidade (I).



(a)

(b) (c) (d)

Figura 8 – Representação das cores no modelo HSI.
(a) Imagem RGB. (b) Componente H. (c) Componente S. (d) Componente I.

O modelo YCbCr (muitas vezes também designado por YUV) se baseia num modo de transmissão de vídeo criado para permitir a compatibilidade da televisão a cores com antigos receptores preto e branco. Por isso, ele separa os sinais do modelo em RGB em um sinal de luminosidade (Y), único utilizado na transmissão sem cores e dois sinais de cromaticidade azul e vermelho (Cb e Cr), que contêm a informação sobre a cor.

A representação gráfica do modelo YCbCr é exibida na Figura 9 e a mesma imagem agora decomposta pelos elementos Y, Cb e Cr é exibida na Figura 10. Na figura, R, G, e B representam *Red*, *Green* e *Blue*, e Y, C e M representam *Yellow*, *Cyan* e *Magenta*.

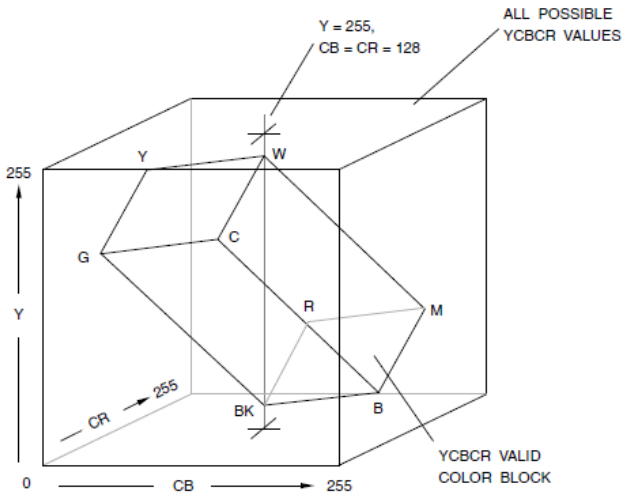
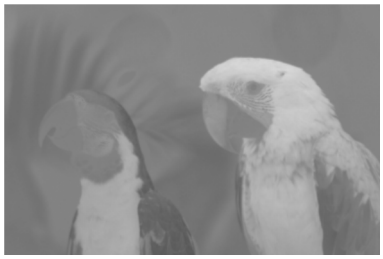


Figura 9 – Representação gráfica do modelo YCbCr. Fonte: (JACK, 2007, p. 43)

(a)

(b)

(c)



(d)

Figura 10 – Representação das cores no modelo YCbCr.
(a) Imagem RGB. (b) Componente Y. (c) Componente Cb. (d) Componente Cr.

O modelo RGB é geralmente o modelo padrão utilizado na aquisição de imagens por diversos dispositivos. Por essa razão, nesta seção são fornecidas equações de conversão deste modelo para os dois modelos aqui apresentados.

A conversão dos valores dos *pixels* no formato RGB para o formato HSI é direta e dada pelas equações a seguir.

O componente *matiz* é dado por:

$$H = \begin{Bmatrix} \theta & se\ B \leq G \\ 360 - \theta & se\ B > G \end{Bmatrix} \tag{6}$$

Sendo θ dado por:

$$\theta = \cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{1/2}}\right\} \tag{7}$$

O componente *saturação* é dado por:

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

E o componente *intensidade* é dado pela média entre os valores R, G e B.

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

As equações da operação inversa dependerão do valor de H. Observe a Tabela 1:

Tabela 1 - Equações de transformação do espaço HSI para o espaço RGB.

	$0º \leq H < 120º$	$120º \leq H < 240º$	$240º \leq H < 360º$
Transformações	$H = H$	$H = H - 120°$	$H = H - 240°$
	$B = I(1-S)$	$R = I(1-S)$	$G = I(1-S)$
	$R = I\left[1 + \frac{S\cos H}{\cos(60° - H)}\right]$	$G = I\left[1 + \frac{S\cos H}{\cos(60° - H)}\right]$	$B = I\left[1 + \frac{S\cos H}{\cos(60° - H)}\right]$
	$G = 3I - (R+B)$	$B = 3I - (R+G)$	$R = 3I - (G+B)$

A conversão do modelo RGB para o modelo YCbCr também é direta, e obtida através das equações (RIBEIRO, 2006):

$$Y = 0.299R + 0.587G + 0.114B \tag{10}$$

$$Cb = (B - Y)0.564 + 128 \tag{11}$$

$$Cr = (R - Y)0.713 + 128 \tag{12}$$

3.2 Segmentação de Regiões da Cor da Pele

A segmentação do objeto de interesse é uma parte fundamental do processamento de imagens. Neste trabalho, os objetos são as mãos e a face. Intuitivamente, a cor é a melhor alternativa para detectá-los, por ser uma característica em comum e bastante invariável às suas formas geométricas. Assim, para encontrar os objetos, o primeiro passo é segmentar toda e qualquer região cor de pele na imagem. Entretanto, detectar essas regiões não é uma tarefa tão simples.

Técnicas robustas não satisfazem o requisito de rapidez necessário para um sistema em tempo real, e técnicas mais triviais, baseadas em limiares conhecidos nos mais variados sistemas de cores tem a desvantagem de ser bastante suscetíveis às mudanças na iluminação, mas, no entanto, oferecem bons resultados.

(VEZHNEVETS, SAZONOV, & ANDREEVA, 2003) fazem um levantamento de diversos trabalhos, os quais apresentam testes e comparações entre modelos de cor e a influência de sua escolha para a separabilidade dos tons de pele. (RIBEIRO, 2006) também utiliza diferentes métodos para esse propósito. Destes trabalhos são extraídas três das melhores formas de segmentação da cor da pele por limiar.

No modelo RGB, um *pixel* é considerado pertencente à pele, se satisfizer à seguinte equação (RIBEIRO, 2006):

$$\begin{gathered}(R > 95) e\ (G > 40) e\ (B > 20)\ e \\ \big((max\{R,G,B\} - min\{R,G,B\}) > 15\big)\ e\ (|R - G| > 15)\ e \\ (R > G)\ e\ (R > B)\end{gathered} \tag{13}$$

O modelo RGB normalizado e a componente I do sistema HSI também podem ser utilizados para separar os *pixels* da pele, que são aqueles que satisfazem (RIBEIRO, 2006):

$$(0.35 \leq r \leq 0.465)\ e\ (0.28 \leq g \leq 0.363)\ e\ (0.25 \leq V \leq 1) \tag{14}$$

No modelo YCbCr, são considerados *pixels* cor da pele aqueles cujas componentes tornem verdadeiras as equações (VEZHNEVETS, SAZONOV, & ANDREEVA, 2003):

$$(77 \leq Cb \leq 127)\ e\ (133 \leq Cr \leq 173) \tag{15}$$

3.3 Algoritmo para Detecção da Face

Em (VIOLA & JONES, 2004), os autores propõem um método para detecção de face[2], que se compara aos melhores métodos de detecção então publicados e inova pela sua rapidez de processamento. Assim, este se tornou um método largamente utilizado no campo de visão computacional em todo o mundo.

Os autores dividem o trabalho em três partes principais. A primeira contribuição é uma nova representação de imagem, a qual denominam "*integral image*", ou imagem integral. A segunda é um classificador simples e eficiente construído utilizando-se o algoritmo de aprendizado AdaBoost para selecionar as características mais úteis de um conjunto vasto de características em potencial. Por fim, a terceira contribuição é um método de combinação de classificadores em forma de cascata, que proporciona, ao permitir o rápido descarte de regiões de fundo, maior rapidez no processamento das prováveis regiões de face (VIOLA & JONES, 2004). Aprofundemo-nos um pouco mais em cada uma dessas contribuições.

A imagem integral é computada da seguinte forma: cada posição (x, y) contém a soma de *pixels* acima e à esquerda. Formalmente, sendo *i* a imagem original e *ii* a imagem integral:

$$ii(x, y) = \sum_{x' \leq x,\ y' \leq y} i(x', y') \tag{16}$$

Utilizando a imagem integral é bastante simples saber a soma dos *pixels* em determinada região retangular. Por exemplo, considere a imagem integral exibida na Figura 11 e o objetivo de descobrir a soma de *pixels* da região *D*. O valor do *pixel* na posição 4 equivale à soma das somas dos *pixels* das regiões *A, B, C* e *D*. O *pixel* na posição 1 equivale a soma dos *pixels* da região *A*, o da posição 2 às regiões *A* e *B* e o da posição 3 às regiões *A* e *C*. Assim, a soma de *pixels* na região *D* pode ser obtida com simples consultas à posições e com

[2] Na verdade, este método pode ser utilizado confiavelmente para outros objetos além da face, desde que estes sejam devidamente texturizados e rígidos

a operação: (4 + 1 – (2 + 3)) – os números representam as posições na Figura 11 (VIOLA & JONES, 2004).

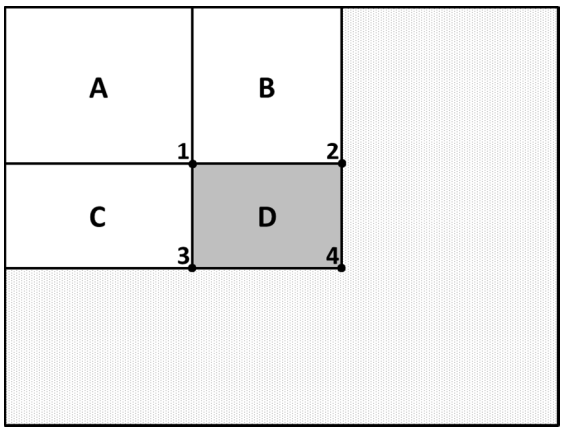


Figura 11 – Exemplo de imagem integral.

A imagem integral é utilizada para computar rapidamente o valor das regiões retangulares, as "Haar *features*" ou perfis Haar. O valor de um *features* sobre uma janela *x* é dado pela equação abaixo, onde *p* representa um *pixel* pertencente à região *x*:

$$f(x) = \sum^{x} p_{preto} - \sum^{x} p_{branco} \tag{17}$$

A biblioteca de visão computacional OpenCV (Open Source Computer Vision) (BRADSKI & KAEHLER, 2008), utilizada no desenvolvimento deste trabalho, implementa uma versão do algoritmo de Viola e Jones estendida por (LIENHART & MAYDT, 2002). Entre outras, a diferença principal está nas *features* diagonais acrescentadas por Lienhart e Maydt.

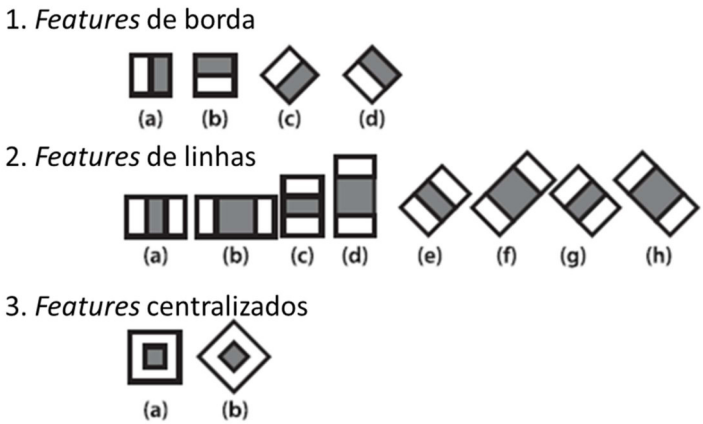


Figura 12 – Haar *features* utilizados na implementação do algoritmo de Viola e Jones na biblioteca OpenCV. (Imagem adaptada / traduzida) Fonte: (BRADSKI & KAEHLER, 2008, p. 509)

Cada grupo classificador está organizado em nós de uma "*rejection cascade*" – ou, livremente, rejeição em cascata, como mostrado na Figura 13.

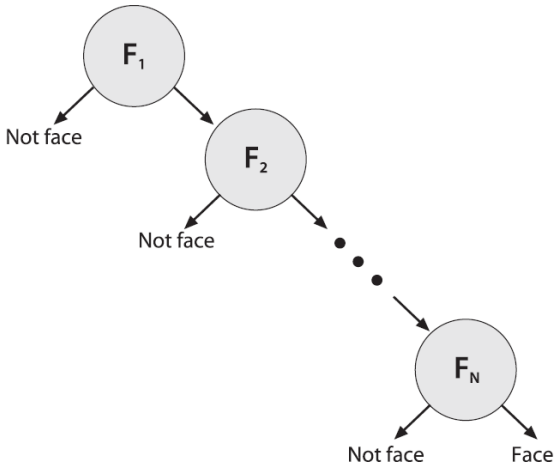


Figura 13 – Classificador em cascata. Fonte: (BRADSKI & KAEHLER, 2008, p. 510).

Cada nó da cascata é por um conjunto de classificadores fortes, os quais são compostos por diversos classificadores fracos. Um classificador fraco *h* é definido pela função abaixo, em que *x* é uma subjanela, *f* é um *feature*, e *t* é um limiar:

$$h(x,p,f,t) = \begin{cases} +1, & se\ f(x) < t \\ -1, & se\ f(x) \geq t \end{cases} \tag{18}$$

Um classificador forte é definido pela função:

$$C(x) = \begin{cases} +1, & se \sum_{t=1}^{T} \alpha_t h_t(x) \leq \frac{1}{2} \sum_{t=1}^{T} h_t(x) \\ -1, & caso\ contrário \end{cases} \tag{19}$$

Em que α_t é uma constante calculada durante o treinamento, $h_t(x)$ é o valor do *t*-ésimo classificador fraco e *T* é o número de classificadores fracos.

O treinamento dos classificadores fortes é realizado pelo algoritmo AdaBoost, cuja descrição é extensa e está fora do escopo deste trabalho.

Os nós estão ordenados de forma que o primeiro nó é aquele que melhor detecta regiões contendo o objeto de interesse. Um objeto é automaticamente rejeitado da classificação, quando classificado como não pertencente à classe "face" em qualquer nódulo, o que agiliza o processo. Na prática, de 70 a 80% das não-faces são rejeitadas nos primeiros dois nós da cascata. (BRADSKI & KAEHLER, 2008)

3.4 Modelos Ocultos de Markov

Uma cadeia é dita de Markov quando se trata de um fenômeno estocástico que obedece a propriedade de Markov. Esta propriedade define que, dada uma série de eventos aleatórios ocorridos ao longo do tempo, a probabilidade de ocorrência de um estado depende somente dos *j* estados mais recentes. Se o valor de *j* é igual a 1, o estado atual depende somente do seu estado anterior, e o processo de Markov é denominado de primeira ordem.

Os Modelos Ocultos de Markov, ou simplesmente HMMs (*Hidden Markov Models*), são similares às cadeias de Markov. Entretanto, a sequência de estados percorrida é oculta ao observador, o qual só enxerga os elementos das observações, pertencentes ao conjunto de símbolos.

Os símbolos e as terminologias referentes aos HMMs adotadas neste trabalho são descritos a seguir, e estão, em sua maior parte, de acordo com as adotadas em (SOUZA, DIAS, & PISTORI, 2007).

Um HMM é representado pelo símbolo λ, e descrito por $\lambda = \{A, B, \pi\}$. A_{NxN} é a matriz de probabilidades de transição entre estados. O conjunto de estados, de tamanho *N*, é representado por $S = \{s_1, s_2, \dots, s_N\}$. O elemento a_{ij} representa a probabilidade de transição do estado origem *i* para o estado destino *j*. B_{NxM} é uma matriz cujos elementos b_{ij} representam a probabilidade do estado *i* gerar o símbolo *j*. O conjunto de símbolos é representado por $S = \{s_1, s_2, \dots, s_N\}$. Por último, o vetor $Q = \{\pi_1, \pi_2, \dots, \pi_N\}$ contém as probabilidades de cada estado ser o primeiro da sequência dos estados geradores.

Em resumo:

HMM	$\lambda = \{A, B, \pi\}$
Matriz de transição entre estados	A_{NxN}
Matriz de probabilidades estado-símbolo	B_{NxM}
Vetor de probabilidades iniciais	$Q = \{\pi_1, \pi_2, \dots, \pi_N\}$
Probabilidade de a sequência O ter sido gerada pelo modelo λ	$P(O\|\lambda)$
Conjunto de N estados	$S = \{s_1, s_2, \dots, s_N\}$
Conjunto de T estados percorridos para determinada observação	$Q = \{q_1, q_2, \dots, q_T\}$
Conjunto de M símbolos	$V = \{v_1, v_2, \dots, v_M\}$
Sequência de T observações	$O = \{o_1, o_2, \dots, o_T\}$

Como um exemplo simples, considere o problema do lançamento de duas moedas, das quais uma é justa e a outra é viciada. Um lançador de moedas lança constantemente uma moeda por vez, que pode ser qualquer uma delas.

Modelando-se o problema como um HMM, o momento em que o lançador estivesse lançando a moeda justa seria um estado, e o momento em que estivesse lançando a moeda viciada, um outro estado. O lançador quase sempre lança a mesma moeda lançada anteriormente, variando somente em 10% das vezes. Estas são as probabilidades de transição de estados.

A moeda justa tem chances iguais de produzir cara ou coroa. A moeda viciada, por sua vez, quando lançada, produz 75% das vezes cara, e 25% das vezes, coroa. Estas são as probabilidades de geração de símbolos.

Por fim, o lançador pode iniciar o processo de lançamento de moedas por qualquer uma delas, o que torna as probabilidades iniciais de cada estado iguais a 50% para cada um.

A Figura 14 exibe o modelo criado, em sua forma gráfica:

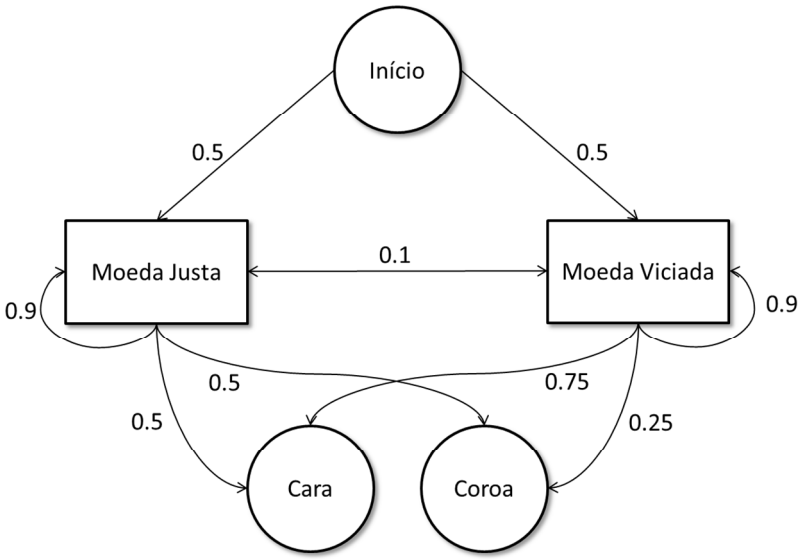


Figura 14 – Exemplo de HMM.

Existem três problemas-chave dentro dos HMMs, cujas soluções são fundamentais para o funcionamento eficaz das aplicações que os utilizam. São eles: o problema da avaliação, o problema da decodificação e o problema do aprendizado.

3.4.1 O Problema da Avaliação

Descobrir a probabilidade de uma dada sequência O de observações ter sido gerada por um modelo λ consiste no problema da avaliação. Uma maneira simples de realizar este cálculo, porém computacionalmente cara, é verificar todas as sequências de estados de tamanho T

possíveis e então calcular suas probabilidades. Alternativamente, dois métodos que usam a programação dinâmica são comumente empregados nesta operação: os métodos *Forward* e *Backward*. Somente um deles é necessário.

Dadas as probabilidades parciais de observações da sequência $o_1 o_2 \dots o_t$ no estado *i* no instante *t* representadas na Equação (20), o algoritmo *Forward* é descrito nos três passos a seguir.

$$\alpha_t(i) = P(o_1 o_2 \dots o_t, q_t = s_t|\lambda) \tag{20}$$

1. Inicialização:

$$\alpha_1 = \pi_i b_i(o_1), 1 \leq i \leq N \tag{21}$$

2. Indução:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} a_t(i) a_{ij}\right] b_j(o_{t+1}), 1 \leq t \leq T-1, 1 \leq j \leq N \tag{22}$$

3. Finalização:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{23}$$

A indução determina que para cada estado s_j, $\alpha_j(t)$ armazena a probabilidade de atingir aquele estado, dadas as observações até o instante *t*. (SIOLA, 2010) Assim, esse valor pode ser consultado para as referências futuras, evitando um novo processamento.

3.4.2 O Problema da Decodificação

O problema da decodificação refere-se à busca da sequência de estados percorrida em um modelo, a qual produz a maior probabilidade final para uma dada sequência de observações. O algoritmo de Viterbi, que também utiliza a programação dinâmica, é geralmente utilizado na resolução deste problema, e segue os seguintes passos:

1. Inicialização:

$$\delta_1(i) = \pi_i b_i(o_1), 1 \leq i \leq N, \psi(i) = 0 \tag{24}$$

2. Recursão:

$$\delta_t(j) = \max_{1 \leq i \leq N} \left[\delta_{t-1}(i) a_{ij}\right] b_j(o_t), 2 \leq t \leq T, 1 \leq j \leq N \tag{25}$$

$$\psi_t(j) = \arg\max_{i \leq i \leq N} \left[\delta_{t-1}(i) a_{ij}\right], 2 \leq t \leq T, 1 \leq j \leq N \tag{26}$$

3. Finalização:

$$P* = \max_{1 \leq i \leq N} [\delta_T(i)] \tag{27}$$

$$q_T^* = \arg\max_{1 \leq i \leq N} [\delta_t(i)] \tag{28}$$

4. *Backtracking* da sequência de estados ótima:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), t = T-1, T-2, \ldots, 1 \tag{29}$$

Através da recursão e do *backtracking*, pode-se determinar os pontos em que o algoritmo retrocede e procura por uma sequência melhor de estados do que a atual. (SIOLA, 2010)

3.4.3 O Problema do Aprendizado

O último problema, o problema do aprendizado, consiste no ajuste dos parâmetros (A,B,π) do HMM, a partir de uma ou mais sequências de observações, que são as sequências de treinamento. Este é um processo executado pelo algoritmo de Baum-Welch, e consiste em um ajuste iterativo e automático dos parâmetros de um HMM de forma que, posteriormente, ele seja capaz de reconhecer outras sequências de observações semelhantes.

Considere a probabilidade de estar no estado s_i no momento t e passar para o estado s_j no momento $t+1$, representada pela Equação (30) e calculada através da Equação (31).

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j | O, \lambda) \tag{30}$$

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} = \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \tag{31}$$

Na equação acima, $\alpha_t(i)$ é a probabilidade de estar no estado si no tempo *t* desde o início da observação e $\beta_t(i)$ é a probabilidade da geração da sequência no modelo no instante *t*.

A probabilidade $\gamma_t(i)$ de estar no estado si no instante de tempo *t* é representada pela Equação (32). Esta probabilidade pode ser definida em função de α e β – Equação (33) – e em função de $\xi_t(i,j)$ – Equação (34).

$$\gamma_t(i) = P(q_t = s_i | O, \lambda) \tag{32}$$

$$\gamma_t(i) = \frac{\alpha_t(i) \beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i) \beta_t(i)} \tag{33}$$

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{34}$$

Tomando a última representação, a probabilidade de partir do estado s_i para o estado s_j na sequência de observações é dada pelo somatório das probabilidades de estar em cada instante da observação no estado s_i e transitar para o estado s_j (SIOLA, 2010):

$$\sum_{t=1}^{T-1} \xi_t(i,j) \tag{35}$$

E a probabilidade de estar no estado s_i e ir para um estado é dada por:

$$\sum_{t=1}^{T-1} \gamma_t(i) \tag{36}$$

Os parâmetros do HMM λ são então calculados através das variáveis π_i, a_{ij} e $b_i(k)$.

$$\pi_i = probabilidade\ de\ estar\ no\ estado\ s_i\ no\ instante\ 1 = \gamma_1(i) \tag{37}$$

$$a_{ij} = \frac{número\ de\ transições\ do\ estado\ s_i\ para\ o\ estado\ s_j}{número\ de\ transições\ do\ estado\ s_i\ para\ qualquer\ estado} \tag{38}$$

$$b_i(k) = \frac{número\ de\ vezes\ que\ foi\ observado\ o\ símbolo\ o_k\ no\ estado\ s_i}{número\ de\ vezes\ no\ estado\ s_i} \quad (39)$$

3.4.4 Aplicação dos HMMs no Reconhecimento de Gestos

A evidência de que os Modelos Ocultos de Markov podem ser utilizados no reconhecimento de gestos, em especial as línguas de sinais, está na semelhança destes com o comportamento da fala, cujo principal método de modelagem nos últimos anos vem sendo os próprios HMMs. (STARNER, WEAVER, & PENTLAND, 1998) Assim como na fala, os movimentos dos gestos podem ser vistos como um fluxo de sinais emitidos ao longo do tempo, sugerindo uma transição entre estados. Em ambos, unidades menores são combinadas para formar unidades mais complexas. No campo da fala, fonemas são combinados para formar palavras. Nos gestos, uma sequência de posturas forma uma palavra.

Os HMMs têm propriedades que os tornam bastante atraentes para o reconhecimento da Libras. Para o treinamento, bastam como entrada conjuntos de observações que, a partir do algoritmo de aprendizado, os parâmetros probabilísticos dos modelos são ajustados.

Os estados serão posturas específicas de cada gesto executado. As observações serão medidas que caracterizam os objetos rastreados, no caso as mãos. E cada gesto será modelado por um HMM.

No contexto do reconhecimento dos gestos, o problema da avaliação será resolvido quando, dada uma sequência de observações – grupos de características – o HMM com a maior probabilidade de tê-la gerado for descoberto e então classificado como o gesto executado.

4 METODOLOGIA

A metodologia desenvolvida pode ser dividida em duas grandes etapas, como ilustra a Figura 15: o rastreamento das mãos e o reconhecimento de gestos. O rastreamento envolve principalmente aquisição das imagens, segmentação, detecção de face, e o monitoramento das regiões das mãos. O reconhecimento de gestos, realizado dentro da biblioteca GART (*Gesture and Activity Recognition Toolkit*) envolve o treinamento e a classificação das observações através dos Modelos Ocultos de Markov, utilizando as características resultantes da primeira etapa.

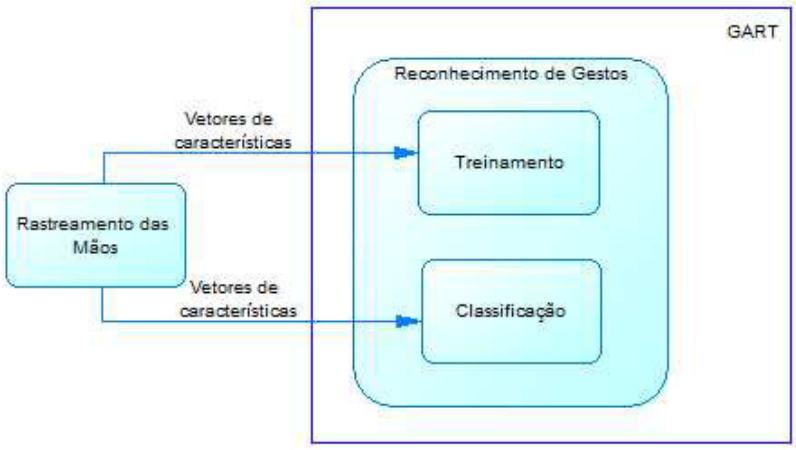


Figura 15 – Fluxo geral da metodologia.

4.1 Rastreamento das Mãos

O diagrama de atividades exibido na Figura 16 explica graficamente a fase de rastreamento. Considere-o como o processamento de um único vídeo.

A imagem adquirida no primeiro passo é extraída de cada quadro dos vídeos adquiridos conforme descrito na seção seguinte. A abertura dos vídeos e a manipulação das imagens são realizados com auxílio da biblioteca de visão computacional OpenCV.

Cada quadro passa pelo processo de segmentação da cor da pele e depois por um pós-processamento para a remoção de ruídos. Estes passos serão descritos na Seção 4.1.2.

O monitoramento das mãos é um passo mais complexo, que envolve a detecção dos agrupamentos de *pixels*, representação dessas regiões em forma de elipses, atribuição de *pixels* a elipses conforme algumas regras e previsão de posicionamento das mãos. E ainda está integrado à etapa de detecção de face para a distinção entre as regiões. Toda esta etapa da metodologia está descrita na Seção 4.1.3.

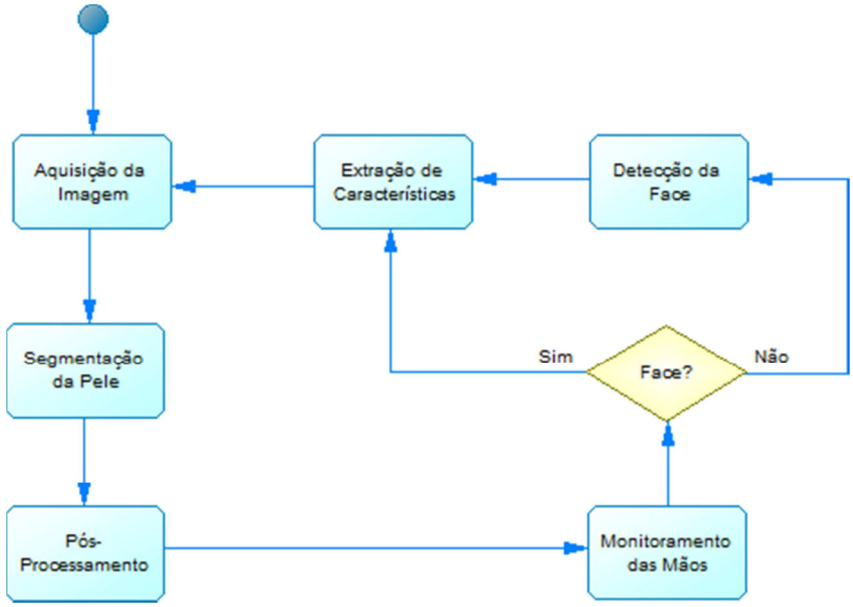


Figura 16 – Diagrama de atividades descrevendo a metodologia na fase de rastreamento.

O resultado final do rastreamento serão vários vetores de medidas que caracterizam as mãos. A extração dessas medidas é a última etapa realizada em cada quadro e está detalhada na Seção 4.1.4.

4.1.1 Aquisição dos Vídeos

A aquisição das imagens para treino e teste foi realizada utilizando-se uma câmera simples integrada a um *notebook*. As filmagens foram realizadas em uma dimensão 640x480 *pixels*, a uma taxa de 15 frames por segundo. Essa resolução foi escolhida por ser relativamente pequena e ainda permitir uma boa visualização dos gestos. No entanto, melhor desempenho em tempo real pode ser obtido, sem perda de precisão, utilizando-se uma resolução menor.

Trinta e um gestos em Libras foram escolhidos para serem reconhecidos. A seleção levou em consideração o significado, a dificuldade e a diferença entre os gestos. Uma primeira seleção resultou subjetivamente em palavras interessantes, com grande significado, ou que pudessem em um futuro trabalho ser agrupadas em sentenças. Por isso, a escolha de palavras de classes gramaticais diferentes: pronomes, verbos e, em sua maioria, substantivos. Deste subgrupo foram selecionados gestos que explorassem as várias posições possíveis da língua: gestos rápidos ou demorados, discretos ou extravagantes, de acima da cabeça até abaixo do quadril, etc. Por fim, foi considerada na seleção também a diferença entre cada gesto. Optou-se por não manter na base gestos com grande semelhança entre si que pudessem

causar dificuldade na sua distinção pelo sistema. O conjunto da gestos selecionados foram os que representam as palavras:

Tabela 2 - Gestos selecionados.

Abacaxi	Calça	Gravata	Pirâmide
Acender	Cinto de Segurança	Hoje	Semáforo
Amigo	Dia	Mau	Tempo
Animal	Eu	Mochila	Transportar
Automóvel	Família	Mouse	Videogame
Balde	Fechar	Música	Você
Boliche	Fila	Obrigado	Xampu
Bom	Gostar	Ouvir	Zebra

Para compor a base de vídeos, cada gesto foi executado cinco vezes por um único intérprete, o que torna o sistema possivelmente dependente de usuário. A filmagem foi realizada com a câmera de frente para o intérprete, com visão completa de todo o seu alcance das mãos. Duas exigências foram atendidas nesta etapa: o intérprete deve usar roupas que deixem à mostra somente as mãos e a cabeça, e o fundo das filmagens não pode conter quaisquer objetos da cor da pele que ocupem grande área da imagem. Por isso os gestos foram executados em fundo preto e, para evitar ruídos causados pela iluminação a luz foi levemente controlada e incide de frente. A Figura 17 ilustra a execução dos vídeos.

Figura 17 – Características das filmagens realizadas.

4.1.2 Segmentação e Agrupamento das Regiões Cor da Pele

Tanto na fase de treinamento como na fase de teste da metodologia, todo quadro adquirido é primeiramente submetido a uma segmentação por limiar, para detecção das áreas cor de pele na imagem. A escolha desse tipo de segmentação foi realizada devido à sua simplicidade de implementação e à sua rapidez de processamento. Além, é claro, de gerar resultados satisfatórios para o propósito deste trabalho.

Uma dificuldade inerente à separação da pele por cor é a sua vulnerabilidade à iluminação do ambiente. Mesmo os modelos que separam a luminosidade dos outros componentes das cores não estão completamente imunes a ambientes com pouca luz ou extremamente iluminados. Uma falha na segmentação pode causar a não detecção das mãos ou a alteração da sua forma e assim comprometer todo o desempenho do sistema nas fases seguintes. Por este motivo, a aquisição das imagens descrita na Seção 4.1.1 foi realizada em ambiente com a luz uniformemente distribuída, de forma a contornar este problema.

Uma região conectada, ou geralmente denominada *Blob (Binary Linked Object)*, é um conjunto de *pixels* que compartilham a mesma *label* – rótulo, identificação. (NIELSEN, CANALÍS, & TEJERA, 2004). Uma análise sobre cada imagem é feita agrupando os *pixels* da cor da pele em regiões distintas e unicamente identificadas por um rótulo. Esta tarefa é auxiliada pela biblioteca cvBlobsLib (OpenCVWiki: cvBlobsLib), construída sobre o OpenCV. Suas funções permitem a rápida detecção dos *blob*s em uma imagem passada como parâmetro, os quais são agrupados em um vetor e possuem propriedades automaticamente calculadas que são de grande utilidade para as etapas seguintes, entre as quais área, perímetro, centro de massa, etc.

Essas regiões conectadas posteriormente serão rastreadas e associadas às mãos ou ao rosto, como explicado na seção seguinte.

4.1.3 Monitoramento das Mãos

Para entender a metodologia de rastreamento das regiões cor de pele utilizada neste trabalho e proposta por (ARGYROS & LOURAKIS, 2004), dois conceitos precisam ficar claros: o de *blob* e o de objeto.

Blob é cada agrupamento de *pixels* da cor da pele. Na Figura 18, por exemplo, existem três *blobs* (b_1, b_2 e b_3). Os *blobs* são detectados conforme especificado na Seção 4.1.2 (utilizando-se a biblioteca cvBlobsLib) a partir das imagens binárias resultantes da etapa de segmentação. São descartados os *blobs* com área menor que 0,1% de toda a área de

visualização, que na execução desse trabalho, foi sempre equivalente a 640x480. Este procedimento permite a eliminação de ruídos, e o objetivo é que permaneçam visualizadas somente mãos e face para a execução desta etapa.

Formas complexas, como as mãos e a cabeça podem ser modeladas por elipses. Uma elipse representa bem essas regiões, definindo propriedades como posição, tamanho aproximado e orientação. As elipses serão os objetos, e são definidas pela quíntupla:

$$e = (C_x, C_y, L, W, \theta) \tag{40}$$

Sendo que (C_x, C_y) é centro da elipse, L é maior eixo, W é o menor eixo, e θ é a orientação da elipse: o menor ângulo entre o maior eixo e a horizontal.

No exemplo mostrado na Figura 18, existem quatro elipses (e_1, e_2, e_3 e e_4). As propriedades geométricas (Equação (40)) são calculadas a partir dos *pixels* da região atribuída à elipse. Este cálculo é feito quando uma nova região (*blob*) aparece na imagem, e também a todo instante, para adaptar o formato da elipse à região a que lhe cabe.

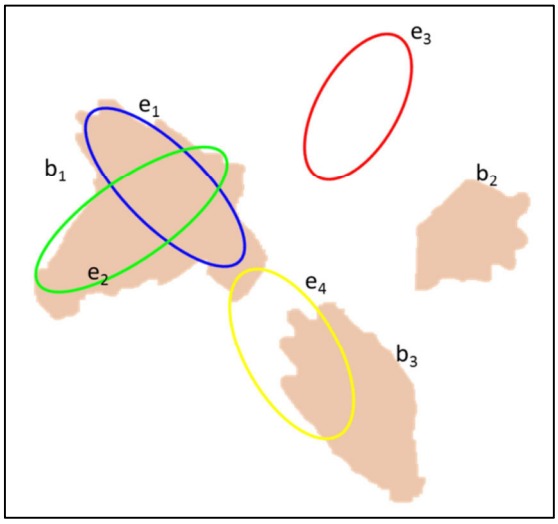


Figura 18 – *Blobs* e objetos.

A equação da distância entre um ponto qualquer e uma elipse, é necessária para o procedimento do rastreamento que será descrito. Fazendo uso deste cálculo, é possível dizer se um *pixel p* de coordenadas *(x,y)* de um *blob* b_i está dentro $(d < 1)$, sobre $(d = 1)$, ou fora $(d > 1)$ de um objeto/elipse e_j, e qual a sua distância até ele. Sendo assim, a Equação (41) será utilizada com bastante frequência no procedimento que segue para acompanhamento das regiões de pele.

$$d(p, e) = \sqrt{\vec{v} \cdot \vec{v}} \tag{41}$$

Sendo que:

$$\vec{v} = \begin{bmatrix} \cos\theta & -\operatorname{sen}\theta \\ \operatorname{sen}\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \frac{x - C_x}{L} \\ \frac{y - C_y}{W} \end{bmatrix} \tag{42}$$

Continuamente, a cada quadro de um vídeo, *blobs* serão associados a objetos, os quais serão reposicionados e redimensionados conforme os *pixels* que lhe foram atribuídos. Essa relação *blob*-objeto nem sempre é de um para um. Um *blob* pode pertencer a mais de um objeto. Porém, um objeto apenas pode conter *pixels* de um único *blob*. Toda essa atividade de atribuição de *blobs* a objetos é regida por algumas regras:

1. Se um *blob* não possui interseção com nenhum objeto, um novo objeto é criado para ele.
2. Quando um objeto possui *pixels* de mais de um *blob*, ele é associado àquele *blob* que esteja "sozinho", ou seja, não contém *pixels* pertencentes a nenhum outro objeto.
3. Para os demais *blobs*, se um *pixel* está dentro de um objeto (isso é verificado através da Equação (31)), ele pertence a este objeto. Do contrário ele pertence ao objeto mais próximo.
4. Se um objeto não contiver *pixels* de nenhum *blob*, ou se após as operações 2 e 3, nenhum *pixel* lhe for atribuído, ele será descartado.

Na prática, a primeira regra se refere ao caso em que novas regiões surgem no campo de visão. Se for a primeira vez que determinado *blob* é detectado, seu movimento obviamente não está ainda sendo controlado, e agora passará a estar.

A segunda regra faz menção a situações em que duas regiões estavam conectadas em um primeiro momento, formando um só *blob*, e então se separam. Neste caso, ocorrerão casos como o demonstrado pelo objeto e_3 na Figura 18, em que *pixels* de diferentes *blob*s estão no interior de uma única elipse. No caso da figura, o objeto e_3 será finalmente atribuído unicamente ao *blob* b_3, pois este não possui interseção com nenhum outro objeto.

A terceira regra trata do caso em que duas regiões distintas se ocluem como no exemplo do *blob* b_1 da Figura 18. Isso ocorre, por exemplo, quando as mãos se cruzam, ou uma delas passa na frente da face. Neste caso, haverá *pixels* pertencentes a mais de um objeto. E também *pixels* pertencentes a nenhum deles, e neste caso, a ideia é atribuí-los ao objeto mais próximo, baseando-se na distância calculada com a Equação (41).

Por fim, a quarta regra engloba o caso das regiões que desaparecem do campo de visão. Se uma mão desaparece do vídeo, o objeto que a representava deve ser excluído. Entretanto, visando contornar possíveis casos de oclusão por regiões não cor da pele ou possíveis falhas de segmentação, um objeto sem *pixels* não é instantaneamente excluído naquela iteração. A ele é permitido uma "sobrevivência" de 15 quadros.

As regiões da cor da pele sofrem deformações ao longo dos quadros, e os objetos são adaptados às suas novas formas a cada instante. Mas os *blobs* também sofrem translações ao longo do tempo, e as elipses precisam acompanhar os seus movimentos, para que se possa manter a associação entre eles o máximo possível. A posição de um objeto é definida pelo seu centroide. A cada quadro tenta-se prever a posição de cada centroide no quadro seguinte, baseando-se no quadro imediatamente anterior. Sendo C_{x_t} e C_{y_t} as coordenadas de um centroide em determinado instante *t*, a previsão é feita da seguinte forma:

$$\begin{aligned} C_{x_{t+1}} &= C_{x_t} + \Delta C_x \\ C_{y_{t+1}} &= C_{y_t} + \Delta C_y \end{aligned} \tag{43}$$

Sendo que:

$$\begin{aligned} \Delta C_x &= C_{x_t} - C_{x_{t-1}} \\ \Delta C_y &= C_{y_t} - C_{y_{t-1}} \end{aligned} \tag{44}$$

Esta forma de previsão é extremamente simples, mas, no entanto, gera um resultado bastante satisfatório.

Para distinguir as mãos e a face, mantém-se uma referência para três objetos especiais: FACE, MÃO_ESQUERDA e MÃO_DIREITA. No primeiro quadro, a face é detectada utilizando-se o algoritmo descrito na Seção 3.3 implementado pela biblioteca OpenCV. Assume-se primeiramente como mão esquerda o objeto não-face mais à esquerda, e como mão direita, o objeto não-face mais à direita. Se a referência para a face é perdida, o algoritmo de detecção é novamente aplicado. E se alguma das mãos é perdida, o próximo objeto detectado é classificado como a mão ausente.

Para a implementação, considere que um vetor de objetos ativos é sempre mantido durante as iterações. Novos objetos detectados são a ele adicionados e objetos sem *blobs* após uma iteração (ou 15, o tempo de sobrevivência) são dele removidos.

Considere ainda uma matriz denominada matriz de frequências, criada para satisfazer a regra 2. A matriz de frequências *M* é uma matriz cujos elementos m_{ij} contêm a

quantidade de *pixels* do *blob i* que pertencem ao objeto *j*, ou seja, $d(p_i,j) \leq 1$, sendo p_i um *pixel* pertencente ao *blob i*. Interagir com essa matriz é mais fácil e rápido para atribuir *pixels* a objetos compartilhados. Analisando uma coluna, por exemplo, é simples saber a quais *blob*s pertencem os *pixels* de um determinado objeto. E analisando uma linha, quais objetos contém *pixels* de um determinado *blob*.

Ao fim do processamento de cada quadro, medidas são extraídas de cada mão conhecida. Se em um quadro, uma das mãos não foi detectada, suas medidas serão zeradas. As medidas representam as características que serão descritas na seção seguinte.

4.1.4 Extração de Características

Para analisar um objeto segmentado em uma imagem ou vídeo, primeiramente é necessário extrair dele características que o representem significativamente, dado o objetivo da aplicação. As características são dados numéricos que podem representar, por exemplo, a forma ou a textura de uma região.

Nos vídeos gravados com execuções de gestos em Libras, é interessante descobrir a posição e a configuração das mãos ao longo do tempo, pois elas é que caracterizam as posturas que compõem os gestos.

(DIAS, SOUZA, & PISTORI, 2006) definem um conjunto de características que representam as posturas. São elas: PEV (Posição Espacial Vertical); PEH (Posição Espacial Horizontal); CON (Configuração da Mão); ORI (Orientação da Palma da Mão); DIP (Direção da Palma da Mão); e SIB (Situação das Bochechas).

As duas primeiras características – PEV e PEH – representam a localização das mãos. PEV determina a altura das mãos em relação ao corpo do usuário, e PEH determina a posição das mãos em relação a um eixo imaginário que corta o centro do corpo do intérprete. Segundo (DIAS, SOUZA, & PISTORI, 2006), para a primeira, existem nove variações possíveis, e para a segunda, oito variações.

A terceira característica – CON – é determinada pela posição dos dedos. São, por exemplo, as configurações das letras do alfabeto. Existem 22 configurações possíveis, segundo (DIAS, SOUZA, & PISTORI, 2006).

ORI e DIP definem a rotação da palma da mão. A primeira é classificada em vertical ou horizontal em relação a um eixo imaginário que segue do pulso até a ponta do dedo médio, e a segunda, define sete variações de orientação em relação ao corpo do usuário.

Uma adaptação destas características foi implementada neste trabalho. Primeiramente, definiu-se que a última característica – SIB –, a qual define se as bochechas estão infladas ou não, seria descartada. Nenhum dos gestos selecionados depende da configuração das bochechas, e somente a análise das mãos já é suficiente para classificar inúmeros movimentos em Libras.

O posicionamento das mãos na vertical e na horizontal é representado por uma espécie de coordenada polar. Considerando-se o centro da face como origem, o eixo horizontal que passa por ele, e o segmento de reta formado pelo ponto da origem e o centro da mão, toma-se por características: a distância *d* em *pixels* dada pelo comprimento desse segmento; e o ângulo α formado entre ele e o eixo horizontal, no sentido horário. A Figura 19 ilustra essas medidas.

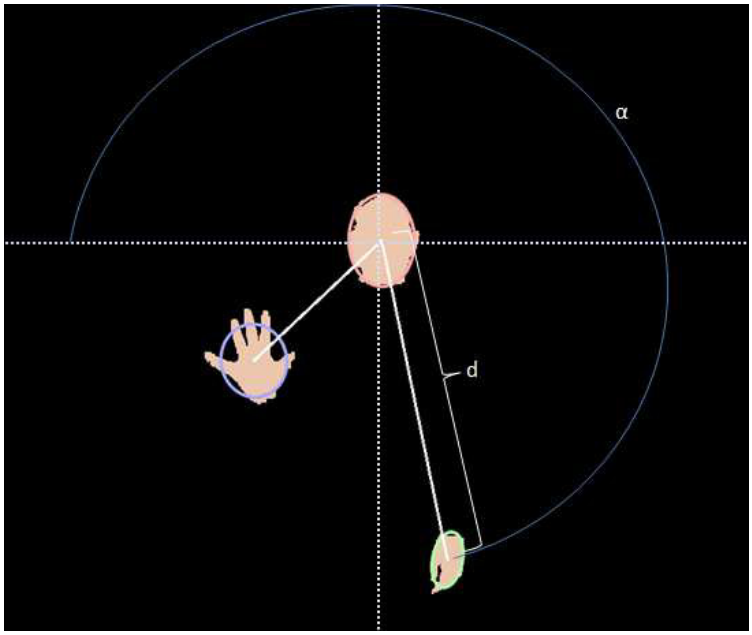


Figura 19 – Medidas da distância e do ângulo da mão em relação à cabeça.

A configuração da mão é a medida mais complexa a ser extraída. O posicionamento dos dedos, mesmo quando completamente visível na imagem, é mais difícil de ser analisado do que as outras informações. No entanto, visto que a soletração não é considerada, esta característica pode ser simplificada. Para medir esta configuração serão utilizadas a área *A* ocupada pela região da mão, dada pela quantidade de *pixels* que ela contém; e a medida de excentricidade *e*, cujo valor aproxima-se de 0 (zero) quando a forma da elipse se aproxima de uma reta, e aproxima-se de 1 (um) quando a forma da elipse se aproxima de um círculo; e, cujo cálculo é feito da seguinte forma, sendo *L* o maior eixo da elipse e *W* o menor eixo:

$$e = \frac{L}{W} \tag{45}$$

A direção da palma da mão (DIP) é dada pelo próprio ângulo θ de orientação das elipses-objeto, e assume-se que esta medida é suficiente para representar a rotação das mãos, descartando-se assim a medida ORI. A Figura 20 exibe uma região de mão com o ângulo θ indicado.

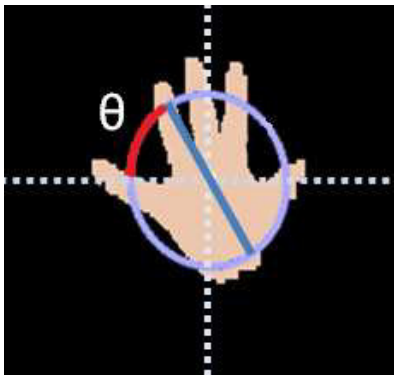


Figura 20 – Ângulo ϑ indicado na mão.

Portanto, um vetor de dez características é extraído de cada quadro q de cada vídeo. O vetor é composto pelas cinco medidas descritas acima, calculadas tanto para a mão esquerda *E*, quanto para a mão direita *D*.

$$\overrightarrow{v_q} = \{d_E, \alpha_E, A_E, e_E, \theta_E, d_D, \alpha_D, A_D, e_D, \theta_D\} \tag{46}$$

Se um vídeo possui sessenta quadros, então sessenta vetores como o vetor acima serão extraídos e armazenados em arquivo XML, cujo formato é descrito na seção seguinte.

4.2 Reconhecimento dos Gestos em Libras

Os Modelos Ocultos de Markov são utilizados para treinar e reconhecer os gestos em Libras. Cada gesto é modelado por um HMM. Os estados são as posturas mais marcantes de cada gesto e as observações são as características extraídas dos vídeos, descritas na seção anterior. As probabilidades são calculadas através do treinamento de parte da base de vídeos, com o algoritmo de Balm-Welch, e os gestos são reconhecidos com a ajuda do algoritmo de Viterbi, o qual, como descrito na Seção 3.4, maximiza a probabilidade de determinado HMM gerar uma sequência de observações. O HMM com a maior probabilidade então é definido como o gesto executado.

Para esta etapa da metodologia, implementou-se um sistema em JAVA com o auxílio de duas ferramentas (*toolkits*): GART – *Gesture and Activity Recognition Toolkit* –, e HTK. A HTK implementa os algoritmos utilizados para a resolução dos três problemas dos HMMs, descritos na Seção 3.4. Construída como uma camada sobre o HTK, a ferramenta GART oferece uma abstração em alto nível para lidar com estes algoritmos. Ela permite que o usuário se preocupe somente com as características extraídas dos vídeos, e o ajuste de alguns poucos parâmetros, para se realizar um treinamento e uma classificação eficientes.

Uma pequena parte dos vídeos foi reservada para teste. O restante foi utilizado para treino. Tanto para treinamento, quanto para teste, a entrada do sistema em JAVA são os arquivos XML gerados ainda na etapa de rastreamento. Estes arquivos estão no formato reconhecido pela biblioteca GART, exemplificado na Figura 21.

A Figura 21 mostra o arquivo XML resultante do processamento de uma execução da palavra "Boliche". Cada elemento `<sample>` agrupa uma execução de um gesto. No exemplo, só há uma execução. O seu primeiro elemento filho, `<tag>`, informa as posições de início e fim do conjunto de vetores de características e o rótulo que elas representam. O conjunto de vetores propriamente é agrupado sob o elemento `<fvectors>`. Cada elemento `<v>` é um vetor de dez características extraídas de um quadro do vídeo, e no exemplo mostrado, existem 61 desses elementos (*start* = 0 e *end* = 60), portanto o vídeo processado possuía 61 quadros. Na figura, as reticências foram utilizadas para melhor visualização.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<library name="Boliche_Library">
    <samples>
        <sample id="1">
            <tag start="0" end="60" label="Boliche" />
            <fvectors>
                <v>-0.159904464317668 -0.284065637661997 ... -0.678651352207210</v>
                <v>-0.151785059987558 -0.280563465128201 ... -0.672580277365668</v>
                ...
            </fvectors>
        </sample>
    </samples>
</library>
```

Figura 21 – Exemplo de arquivo XML resultado da fase de rastreamento.

Adicionando-se mais amostras (*samples*) do gesto "Boliche", este arquivo exemplo se torna uma entrada para o treinamento. A base completa de treinamento é formada por um arquivo de cada gesto, cada qual contendo várias amostras. Utilizando esta base a biblioteca GART estima os melhores parâmetros de cada HMM representante de um gesto. O

usuário pode decidir alterar alguns parâmetros do treinamento, como por exemplo, o número de estados dos HMMs, em busca de melhores resultados, ou simplesmente utilizar os valores padrão.

O resultado do treinamento são arquivos de *log*, configuração, detalhamento da base de treino, lista de símbolos, entre outros. Estes arquivos serão carregados no momento de uma classificação. Os arquivos de uma entrada de testes possuem o mesmo formato descrito acima, porém contendo qualquer número de amostras, e possivelmente de gestos diferentes. Todos eles serão lidos e os gestos classificados correspondentes serão exibidos na tela de saída.

5 RESULTADOS

Neste capítulo serão apresentados os testes e resultados das principais etapas descritas na metodologia detalhada no Capítulo 4.

O método de segmentação da pele foi selecionado com base em testes realizados em algumas imagens. A Figura 22 apresenta um desses testes. A imagem original, adquirida pela câmera no modelo RGB e mostrada na Figura 22(a) foi segmentada de três formas diferentes. Na Figura 22(b), segmentou-se no próprio modelo RGB utilizando-se o limiar da Equação (13). A imagem da Figura 22(c) é resultado de uma segmentação utilizando-se o limiar da Equação (15), após a transformação da imagem para o modelo YCbCr. Por fim, a segmentação utilizando o espaço RGB normalizado e a componente H do modelo HSI – limiar da Equação (14) – resultou na Figura 22(d).

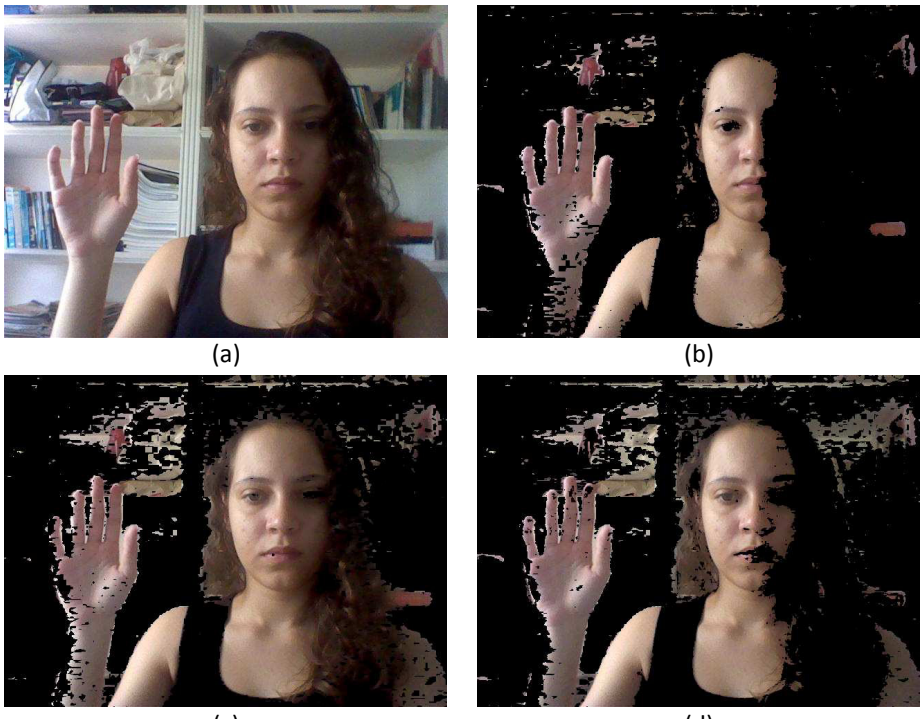


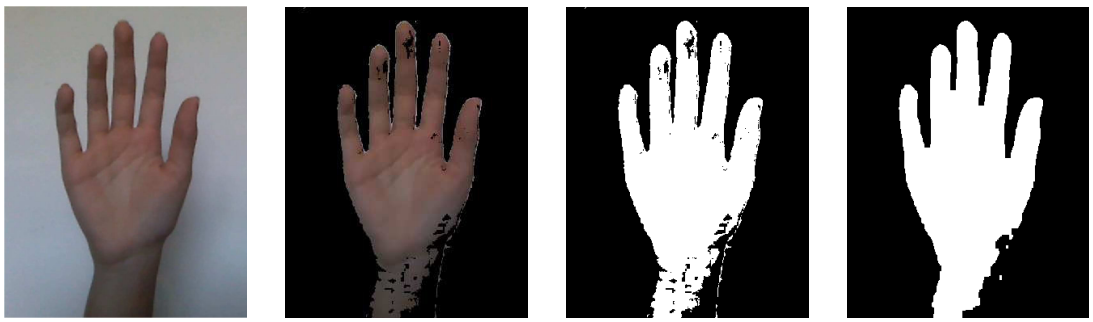
Figura 22 – Testes de segmentação da pele. (a) Imagem original. (b) Segmentação com limiar em RGB. (c) Segmentação com limiar em YCbCr. (d) Segmentação com limiar em RGB normalizado e componente H do HSI.

A melhor forma de segmentação da pele variou conforme o ambiente de filmagem. Observou-se que a luminosidade do local é um fator bastante interferente neste processo. Locais muito iluminados ou muito escuros não permitem a correta separação das regiões cor de pele nas imagens. A direção da luz também influencia. Observe nas imagens da

Figura 22, em que a fonte de luz está à esquerda, como a segmentação na região à direita das imagens foi prejudicada, em todos os casos.

Para os vídeos de treino e teste, gravados em local com a luz de frente, levemente controlada e em fundo preto, o método de segmentação utilizando as componentes normalizadas RGB e a componente H do modelo HSI apresentou os melhores resultados, e portanto foi o método utilizado para segmentar as mãos e a face nos vídeos.

Depois de segmentadas, as imagens apresentam ruídos causados por sombras e diferenças na luminosidade do local. Assim, as mãos e a face não ficam muito bem definidas, apresentando "buracos" e *pixels* extras em volta do seu contorno. Com essas características, não se obtém representações fiéis à forma atual das mãos, o que compromete o resultado das etapas seguintes. Portanto, após a segmentação, cada imagem passa por processos de erosão e dilatação. Observe na Figura 23, um teste de segmentação seguido de um pós-processamento para a remoção de ruídos. Novamente, a quantidade ideal de iterações dessas operações variou conforme o ambiente de filmagem. Porém, a sequência de 2 erosões, 5 dilatações e 3 erosões utilizando um elemento estruturante quadrado de tamanho 3x3 se mostrou satisfatória para a maioria dos casos, e especialmente para o pós-processamento dos vídeos da base.



(a) (b) (c) (d)

Figura 23 – Pós-processamento: aplicação de erosões e dilatações. (a) Imagem original. (b) Imagem segmentada. (c) Imagem binarizada. (d) Imagem pós-processada.

O *software* criado para o módulo de rastreamento permite que o monitoramento das mãos possa ser acompanhado visualmente através de duas janelas. A Figura 24 exibe seis momentos da execução do gesto "Zebra". Na coluna da esquerda, as imagens mostram os *blob*s e as elipses de cada quadro. Cada elipse é unicamente identificada por uma cor, e cada mão é ligada à cabeça por uma linha da cor branca. Na coluna da direita, é exibida a saída correspondente de cada quadro, contendo a imagem original sobreposta pelo contorno detectado das mãos e da cabeça. Observe a correspondência correta das cores, o que indica um rastreamento bem-sucedido.

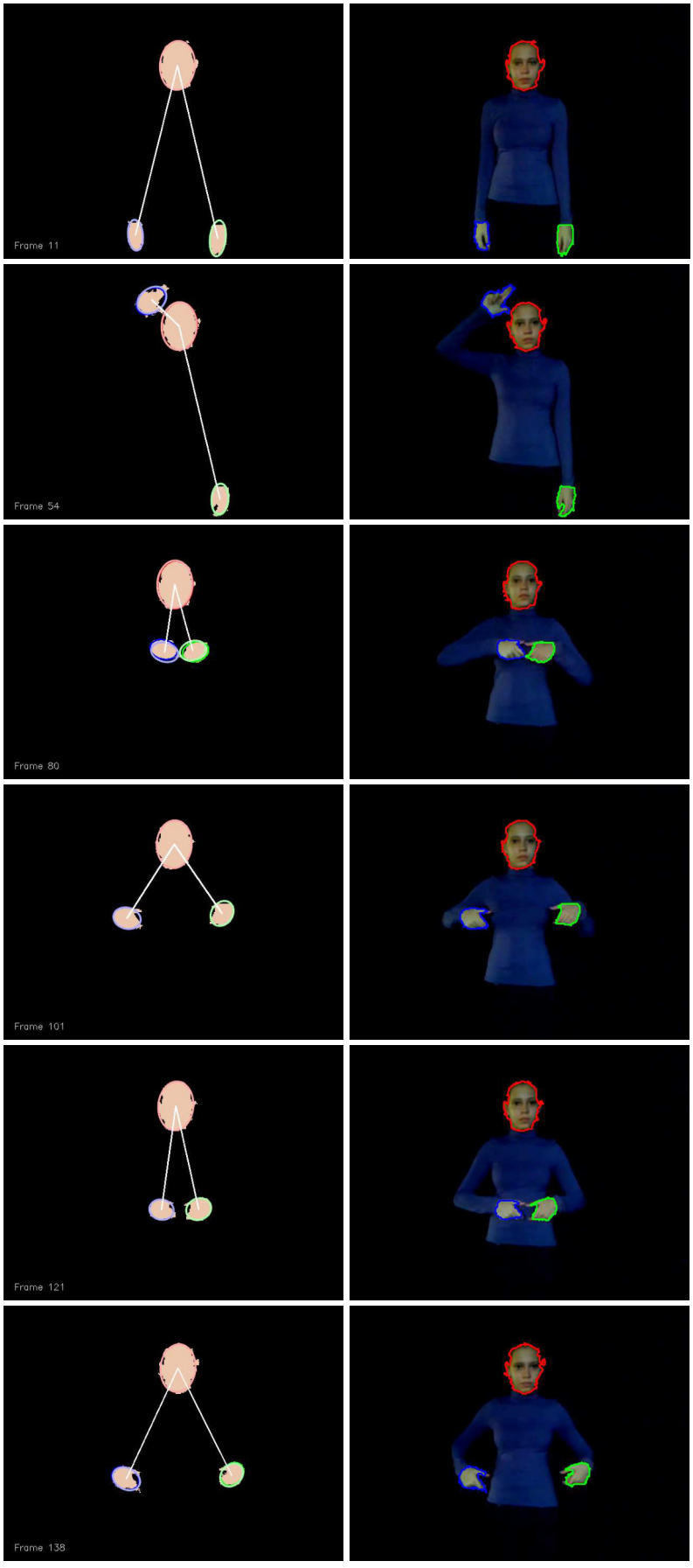


Figura 24 – Alguns quadros da execução do gesto "Zebra".

Nos casos em que há sobreposição das mãos entre si ou com o rosto, o sistema obteve sucesso em algumas situações, mas apresentou falhas em vários momentos. A Figura 25 apresenta quatro casos em que há oclusão. Observe que as duas primeiras imagens são casos de sucesso, em que as diferentes regiões estão corretamente delimitadas por diferentes cores. Os vídeos cujo rastreamento das regiões não foi realizado de forma correta foram descartados.

Figura 25 – Casos de sobreposição de regiões.

Os 32 gestos selecionados foram executados cinco vezes, gerando um total de 160 vídeos. Destes, 96 foram utilizados para treinamento e teste, e 64 exclusivamente para teste. Ajustando-se o parâmetro da quantidade de estados de cada HMM antes do treino para 8 estados, obteve-se o melhor resultado.

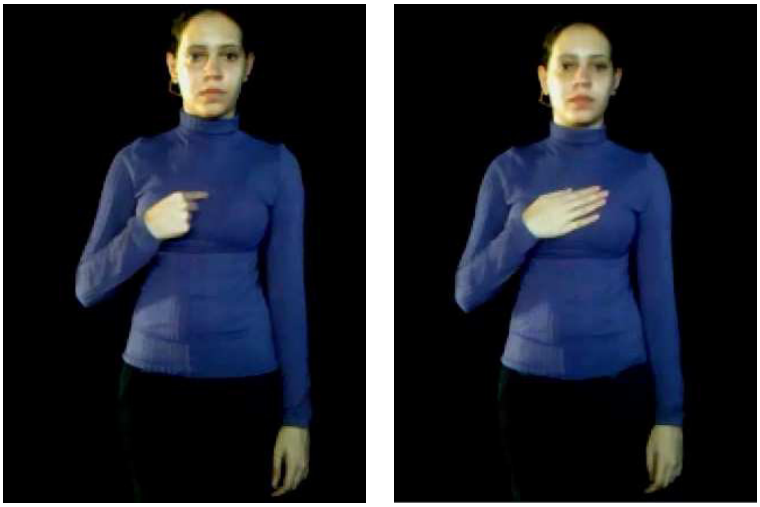
A classificação da base de treino obteve uma taxa de acerto de 100%. Na classificação dos vídeos reservados para teste, o sistema acertou 81,2%. Totalizando, sobre todos os vídeos testados, o sistema obteve uma taxa de acerto de 92,5%. A Tabela 3 resume estes resultados.

Tabela 3 - Resultado da classificação.

Base de Vídeos	Total de Vídeos	Total de Acertos	Porcentagem de Acerto
Treino	96	96	100%
Teste	64	52	81,2%
Total	160	148	**92,5%**

Alguns gestos sempre foram reconhecidos pelo sistema. Outros, cujas configurações são parecidas em algum momento foram confundidos. Por exemplo, os gestos “Eu” e “Gostar” são gestos de curta duração em que a mão esquerda é posicionada na região verticalmente central da imagem, como ilustrado na Figura 26. Se a configuração das mãos fosse um fator melhor analisado, este erro provavelmente não aconteceria. Outros confusões

acontecem principalmente entre os gestos: “Automóvel” e “Cinto de Segurança”; “Hoje” e “Semáforo” e “Acender” e “Semáforo”.



(a) (b)

Figura 26 – Comparação dos gestos confundidos.
(a) Eu. (b) Gostar.

6 CONCLUSÃO

Este trabalho apresentou o desenvolvimento de uma metodologia para o reconhecimento de gestos em Libras, o qual engloba o rastreamento das mãos e a classificação de vídeos com execuções de um grupo de gestos pré-determinados.

Na etapa de rastreamento, a metodologia se mostrou rápida e eficaz nos casos em que a segmentação das regiões foi realizada com perfeição, e para os gestos em que não ocorreu a sobreposição das mãos, ou das mãos e da cabeça.

Falhas na segmentação causadas por problemas de iluminação afetam todas as etapas posteriores. Qualquer método baseado no pigmento individual dos *pixels*, sob qualquer sistema de cor, está suscetível a esse problema. Além disso, mesmo com iluminação adequada, detectar a pele pela cor exige que não haja outras regiões cor da pele além das regiões alvo, ou então que o fundo seja estático para que se realize uma subtração. Todas estas restrições são incômodas e diminuem a naturalidade da interação com o sistema. Desta forma, futuramente pode-se analisar outras formas de segmentação, preferencialmente favorecendo a forma ou a textura e não a cor individual dos *pixels*.

Quanto aos gestos em que há sobreposição, o rastreamento falhou em alguns momentos, o que requer um melhoramento do algoritmo utilizado para o monitoramento das mãos.

Na etapa de classificação dos gestos, utilizando os Modelos Ocultos de Markov, a metodologia se mostrou extremamente bem-sucedida. Como um trabalho futuro, são sugestões: a extensão da quantidade de gestos para reconhecimento e a elaboração de uma base maior, com vídeos executados por mais de um intérprete, no intuito de tornar o método menos dependente de usuário.

Além disso, pode-se elaborar um método que permita a execução sequencial dos gestos, e possivelmente com base gramatical e até semântica. Isso exigirá bastante pesquisa, mas passos iniciais podem ser dados com HMMs dentro de HMMs, uma vez que, sem uma análise muito profunda, a transição entre palavras também pode possivelmente ser modelada como uma transição entre estados. Para investir na sintaxe, outra sugestão são técnicas utilizando gramáticas e autômatos finitos.

De maneira geral, a metodologia apresentada neste trabalho se mostrou bem sucedida e bastante promissora para o desenvolvimento de um grande sistema auxiliar na busca pela melhoria da comunicação e integração social dos deficientes auditivos do país.

REFERÊNCIAS

ARGYROS, A. A., & LOURAKIS, M. I. (2004). Real-Time Tracking of Multiple Skin-Colored Objects with a Possibly Moving camera. *ECCV*, 368-379.

AZEREDO, E. (2006). *Língua Brasileira de Sinais "Uma Conquista Histórica"*. Brasília: Senado Federal.

BRADSKI, G., & KAEHLER, A. (2008). *Learning OpenCV*. Sebastopol, CA: O'Reilly Media.

DIAS, J. B., SOUZA, K. P., & PISTORI, H. (16-18 de Outubro de 2006). Conjunto de Treinamento para Algoritmos de Reconhecimento de LIBRAS. *II Workshop de Visão Computacional*. São Carlos, São Paulo.

GONZALEZ, R. C., & WOODS, R. C. (2010). *Processamento Digital de Imagens* (3ª Edição ed.). São Paulo: Pearson Prentice Hall.

IBGE. (2010). Características Gerais da População, Religião e Pessoas com Deficiência. *Censo Demográfico 2010*.

JACK, K. (2007). *Video Demystified: A Handbook for the Digital Engineer*. Burlington: Elsevier Inc.

JUANG, B. H., & RABINER, L. R. (August de 1991). Hidden Markov Models for Speech Recognition. *Technometrics, 33*, pp. 251-272.

LIENHART, R., & MAYDT, J. (2002). An Extended Set of Haar-like Features for Rapid Object Detection. *IEEE ICIP*, *1*, pp. 900-903.

NIELSEN, E. S., CANALÍS, L. A., & TEJERA, M. H. (2004). Hand Gesture Recognition for Human-Machine Interaction. *WSCG*, (pp. 395-402).

OpenCVWiki: cvBlobsLib. (s.d.). Acesso em 23 de Julho de 2012, disponível em http://opencv.willowgarage.com/wiki/cvBlobsLib/

RIBEIRO, H. L. (2006). Reconhecimento de Gestos Usando Segmentação de Imagens Dinâmicas de Mãos Baseada no Modelo de Misturas de Gaussianas e Cor de Pele. *Dissertação de mestrado, Escola de Engenharia de São Carlos da Universidade de São Paulo*. São Carlos.

SHU, H. (February de 1997). On-Line Handwriting Recognition Using Hidden Markov Models. *M.S. thesis, Massachusetts Institute of Technology*.

SIOLA, F. B. (2010). *Desenvolvimento de um Software para Reconhecimento de Sinais em LIBRAS através de Vídeo.* Trabalo de Conclusão de Curso (Ciência da Computação) submetido à Universidade Federal do ABC, Santo André.

SOUZA, K. P., DIAS, J. B., & PISTORI, H. (22-24 de Outubro de 2007). Reconhecimento Automático de Gestos da Língua Brasileira de Sinais utilizando Visão Computacional. *III WVC - Workshop de Visão Computacional*. São José do Rio Preto, São Paulo.

STARNER, T., WEAVER, J., & PENTLAND, A. (Dezembro de 1998). Real-Time American Sign Language Recognition Using Desk and Wearable Computer-Based Video. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 20*(12), pp. 1371-1375.

TANGUARY Jr., D. O. (August de 1995). Hidden Markov Models for Gesture Recognition. *PhD thesis, Massachussetts Institute of Tecnology*.

VEZHNEVETS, V., SAZONOV, V., & ANDREEVA, A. (2003). A Survey on Pixel-Based Skin Color Detection Techniques. *Proceedings of GraphicCon*, (pp. 85-92). Moscow.

VIOLA, P., & JONES, M. (2004). Robust Real-Time Face Detection. *International Journal of Computer Vision, 57*, pp. 137-154.

Site: O que é Libras. (2000). Acesso em 20 de Julho de 2012, disponível em Portal de Libras: http://www.libras.org.br/libras.php

Site: Curso de Libras Gratuito. (2009). Acesso em 20 de Julho de 2012, disponível em ICursos Online: http://www.icursosonline.com/curso-de-libras-gratuito/