Vinícius Godoy de Mendonça

MÉTODO PARA CLASSIFICAÇÃO DE UM CONJUNTO DE GESTOS USANDO KINECT

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

CURITIBA

2013

VINÍCIUS GODOY DE MENDONÇA

MÉTODO PARA CLASSIFICAÇÃO DE UM CONJUNTO DE GESTOS USANDO KINECT

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Área de concentração: Ciência da Computação

Orientador: Prof. Dr. Alceu de Souza Britto Júnior
Co-orientador: Prof. Dr. Jacques Facon

CURITIBA

2013

Mendonça, Vinícius Godoy de
MÉTODO PARA CLASSIFICAÇÃO DE UM CONJUNTO DE GESTOS USANDO KINECT Curitiba, 2013. 86p.

Dissertação de Mestrado. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática.

1. Modelos escondidos de Markov 2. Kinect 3. Reconhecimento de sinais. 4. Libras. Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e Tecnologia. Programa de Pós-Graduação em Informática Aplicada.

Dedico essa obra a minha esposa,
Camila Luiz Wolinger,
que me apoiou em todos os momentos.

Agradecimentos

Agradeço primeiramente a minha esposa, Camila Luiza Wolinger, pelo carinho e pelo suporte dado em todos os anos que passamos juntos.

Agradecimento especial à Carla Cavichiolo Flores, que me apoiou ao permitir que eu me ausentasse do trabalho para cursar as disciplinas exigidas pelo núcleo.

Ao meu orientador Dr. Alceu Britto Jr. pelas boas conversas que tivemos, por não permitir que eu desistisse nas dificuldades encontradas, pela inesgotável paciência e por estar disponível a todo o momento.

A todos só posso dizer: Muito Obrigado!

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

O	Sequência de observações
S	Conjunto de estados
λ	Conjunto completo de parâmetros de um HMM
N	Número de estados distintos do HMM
M	Número de símbolos observados distintos
T	Tamanho da sequência de observações
t	Tempo em que a observação foi vista
V	Conjunto de símbolos individuais presentes nas observações
A	Probabilidade do estado s_j ocorrer no tempo $(t+1)$ dado o estado s_i no tempo t.
B	Distribuição de probabilidade do símbolo observado no estado j
π	Probabilidade que o estado inicial seja s_j
Δ	Limitador para um salto HMM no modelo Esquerda/Direita
c	Peso de uma mistura a um estado
α	Variável *forward*
β	Variável backward
$\xi_t(i,j)$	Função da probabilidade de estar no estado s_i no tempo t, e no estado s_j no tempo $t+1$
$\Upsilon_t(i)$	Função da probabilidade de estar no estado si no tempo t, dada a sequencia de observações O e o modelo λ
δ	Função que avalia a probabilidade da observação O ter sido produzida pela sequencia mais provável de estados do modelo
ψ	Vetor usado para rastrear o caminho de máxima verossimilhança
$\vec{x}$	Vetor de amplitude contínua com valores reais
$\vec{y}$	Vetor de amplitude discreta com valores quantizados
q	Função de quantização
C_i	Centróide do vetor i
$d(\vec{x},\vec{y})$	Função de distorção da quantização de x em y
G(x)	Função gaussiana de x
Γ	Coeficiente de normalização
σ	Tamanho da função gaussiana
$H(x,y)$	Matriz Hessiana

Lista de Abreviaturas

AG	Algoritmo Genético
APADA	Associação de Pais e Amigos de Deficientes Auditivos
BGR	Blue, Green, Red – Formato de cor
DCT	Discrete Cosine Transform – Transformada discrete de cossenos
DTM	Dynamics Time Warping
FENEIS	Federação Nacional de Educação e Integração de Surdos
FLVQ	Fuzzy Learning Vector Quantization
HMM	Modelo Escondido de Markov (Hidden Markov Model)
IBGE	Instituto Brasileiro de Geografia e Estatística
Libras	Língua Brasileira de Sinais
LVQ	Learning Vector Quantization
MDI	Informação Mínima Discriminatória
MLP	Multi-layer perceptrons – Perceptron multi-camadas
MMI	Informação Mútua Máxima
ONG	Organização Não Governamental
OpenCV	Open Computer Vision
OpenNI	Open Natural Interfaces
PS	Perceptron Simples
RGB	Red, Green, Blue – Formato de cor
RGBD	Red, Green, Blue, Depth - Formato de cor
RNA	Rede Neural Artificial
SIFT	Scale Invariant Feature Transform
SURF	Speed Up Robust Features
UFPR	Universidade Federal do Paraná
VQ	Quantização de Vetores (Vector Quantization)
BW	Baum-Welch
ROI	Região de Interesse (Region of Interest)

Resumo

Por muitos anos, a comunidade surdo-muda brasileira tem lutado pelo direito de poder exercer de maneira completa sua cidadania. Embora conquistas importantes tenham sido realizadas, como ter a Língua Brasileira de Sinais (Libras) reconhecida oficialmente em 2002, seu uso ainda é muito restrito em meios de comunicação e ferramentas, inclusive nas informatizadas. Na tentativa de solucionar esse problema, a pesquisa de uma solução que rastreie a mão e identifique o sinal gesticulado em Libras, se faz necessária. Neste documento, uma abordagem para classificação de um conjunto de nove sinais será proposta, usando como base o sensor Kinect, da Microsoft.

Palavras-chave: Modelos escondidos de Markov, Kinect, Reconhecimento de sinais, Libras

Abstract

For many years, the deaf community claimed for the right of fully practicing their citizenship. Although important advances had been taken, such as the official adoption of Brasilian Sign Language (Libras) in 2002, its use is still very restrict in the media and tools, including computer based ones. In order to solve this problem, the research of a hand tracking and Libras gesture recognition solution, must be made. In this document, we propose an approach to classify a set of nine signals based on Microsoft Kinect Sensor tracking sensor.

Keywords: Hidden Markov Models; Kinect, Gesture Recognition; Libras

Capítulo 1

Introdução

Os avanços recentes das tecnologias voltadas à captura de imagens têm viabilizado a aplicação de técnicas de Visão Computacional e Reconhecimento de Padrões em diferentes áreas da atividade humana.

Tais avanços contribuíram, sobretudo, na interface do homem com a máquina, viabilizando desde jogos de computadores onde o jogador gesticula suas ações, até equipamentos de segurança que funcionam sem qualquer tipo de intervenção humana.

Nesse sentido, a visão computacional também pode fornecer uma interface natural para indivíduos que não se comunicam através de palavras, mas sim, de uma linguagem gesticulada: os surdos.

1.1. Definição do problema

A comunicação é de fundamental importância para qualquer cidadão. A linguagem está presente em nosso dia-a-dia. Em nossos lares, aprendemos a falar, antes mesmo de iniciarmos o processo de alfabetização. Esse conhecimento abre-nos as portas para o ensino da linguagem escrita e, com essa linguagem, organizamos a sociedade.

No ano de 2000 o Instituto Brasileiro de Geografia e Estatística (IBGE) apurou em seu CENSO que cerca de um 5,7 milhões de brasileiros tinham algum grau de

deficiência auditiva. Desses, um milhão possuíam sérias deficiências permanentes de audição e, aproximadamente 160.000, correspondiam a pessoas sem a capacidade de ouvir [IBG11].

O despreparo das instituições do estado sejam elas ONGs, cartórios ou mesmo escolas e faculdades, amplia esse problema durante a vida do surdo. A título de exemplo, em Bauru/SP, apurou-se em 2006 que 50% das instituições estaduais não possuíam estrutura para lidar com essa população [Lob06].

Uma estrutura ideal é formada por um conjunto de procedimentos, pessoas e ferramentas, que auxiliem na comunicação com esses cidadãos. Dentre essas ferramentas, a principal é a Língua Brasileira de Sinais (Libras), considerada forma de comunicação oficial no Brasil a partir do decreto lei nº 10.436/2002 [Bra02].

Embora essa lei tenha tornado oficial o ensino da língua em instituições de ensino, hospitais e órgãos do governo, sua promulgação teve pouca efetividade até sua regulamentação que só ocorreu três anos depois, com o decreto regulamentar nº 5626/2005. Uma das mais importantes medidas desse decreto foi a oficialização do termo surdo, considerando como tal "toda pessoa que, por ter perda auditiva, compreende e interage com o mundo por meio de experiências visuais e que manifesta sua cultura principalmente pelo uso da Língua Brasileira de Sinais – Libras." [Gom06].

Entretanto, mesmo que uma pessoa surda seja fluente em Libras, ainda apresentará dificuldade em comunicar-se [Fer07]. Em primeiro lugar, como a própria lei já reconhece, Libras é um "sistema linguístico de natureza visual-motora, com estrutura gramatical própria" [Bra02] e, portanto, não se trata de mera soletração do português ou de pura gesticulação de palavras. Trata-se de uma língua própria, diferente inclusive de outras formas de linguagem de sinal, de outros países do mundo.

A dificuldade em se comunicar já começa na infância. Desde os primeiros passos, a criança surda encontra dificuldades, como comunicar com os pais e aprender os primeiros gestos.

O primeiro passo para reduzir essa diferença entre surdos e ouvintes é certamente a educação. Atualmente, o computador tem sido poderoso aliado na educação infanto-juvenil. Segundo [Geb09] a utilização da tecnologia na educação proporcionou uma melhoria no processo de ensino-aprendizagem, tanto para alunos quanto para educadores. Através dele, é possível acelerar o processo de aprendizagem

nos primeiros estágios da vida e, ao mesmo tempo, propriciar interação social entre educandos, educadores ou familiares.

[Mat10] orienta o processo educacional eletrônico a ser gradual, e motivado por uma atividade interativa lúdica. Dentre os estilos de jogo comuns sugeridos por ele, estão os "jogos de aventura", por três motivos:

a) Este gênero convida o jogador a participar de uma narrativa simples.

b) O conjunto de palavras utilizado para interação com o jogo é pequeno.

c) A narrativa é tema de discussão entre educador e educando.

Um exemplo de jogo desse estilo pode ser visto na figura Figura 1-1, contendo o vocabulário de oito das nove palavras alvo: "entregar", "abrir", "fechar", "pegar", "olhar", "falar", "empurrar" e "puxar".

Figura 1-1: Jogo "The Secret of Monkey Island" da Lucas Arts

Ao invés do português ou inglês, essas palavras poderiam ser gesticuladas em Libras. Assim, torna-se necessário uma ferramenta capaz de reconhecer este conjunto fixo de sinais.

Assim, a criança poderia gesticular o comando desejado, aprendendo-o, e receber o *feedback* do programa na forma de falas e textos, que poderiam ser exibidos através de um vídeo em Libras, juntamente com sua legenda em português. O uso das duas línguas permite a interação entre a criança surda e seu educador, seja ele seu pai ou um professor. Entretanto, a produção de tal ferramenta passa por alguns desafios.

O primeiro deles está no fato de que a maioria das tecnologias de reconhecimento de gestos necessitam de treinamento, utilizando-se para tal uma grande

base de vídeos. É que o que se observa nos trabalhos de [Fan07] e [AnH10], que utilizaram mais de duas mil amostras de entrada.

Esse número é ainda pequeno, se comparado a pesquisas envolvendo tecnologias similares, como o reconhecimento de caracteres manuscritos, onde dezenas de milhares de amostras são utilizadas [Rab93]. Compor uma base de vídeos dessa ordem é uma tarefa que levaria anos e, portanto, torna-se necessário trabalhar com um conjunto reduzido e otimizar seu uso, de modo que seu tamanho tenha impacto menos significativo possível sobre a pesquisa.

O segundo desafio está na gravação dessa base de dados. Por se tratar de um dispositivo de captura recente, e de uma Língua nacional com características próprias, não foram encontradas bases previamente montadas.

Finalmente, o último desafio está no reconhecimento do gesto em si. Para ser efetivo como ferramenta educacional e socialmente abrangente, ele deve ser realizado num ambiente pouco controlado, como a casa do indivíduo. Ou seja, ele deve impor as mesmas limitações existentes hoje para o uso do aparelho no *vídeogame*, sem adicionar a necessidade de estúdios ou iluminação especiais.

Além disso, o Kinect não é capaz de reconhecer os dedos da mão e, portanto, uma estratégia para segmenta-la e combina-la a informação de posição deve ser traçada. Essa informação deve ser combinada a do esqueleto, já fornecida pela ferramenta, para formar um vetor de características único. Esse vetor, para ser submetido aos modelos escondidos de Markov, deve passar por algum processo de redução (quantização). Representar a informação segmentada, portanto, é uma tarefa de fundamental importância para o desenvolvimento da ferramenta.

Por último, é necessário traçar uma estratégia para validar o reconhecimento do gesto. Assim, é necessário determinar como processar e separar os vídeos da base de dados, de modo a compor grupos para treinamento e validação, evitando desperdício dos vídeos da base e, ao mesmo tempo, a contaminação dos resultados pela mistura indevida de vídeos das duas categorias.

1.2. Motivação

A primeira motivação para o desenvolvimento deste trabalho é certamente a social. Reduzir as diferenças sociais é um dos focos do governo brasileiro [Fer07]. É importante ressaltar que tal ferramenta permitiria ao surdo não só jogar um *jogo*, mas operar máquinas, fazer solicitações simples – tais como realizar o pedido em um cardápio, trocar os canais da TV ou solicitar a presença de um intérprete. Além do surdo, a gesticulação de sinais num ambiente lúdico e divertido, pode motivar qualquer pessoa a aprender o idioma, reduzindo a diferença social existente entre esses dois universos.

A segunda motivação é a tecnológica. Recentemente, a visão computacional tem ganhado foco, especialmente após a popularização de sensores variados de captura, capazes de simplificar a extração das características necessárias para obtenção desse sinal. Neste sentido, chama a atenção o dispositivo Kinect, e produzido pela Microsoft, que tornou-se extremamente popular por ser distribuído juntamente com o video-game da companhia, ou vendido separadamente. Explorar o potencial desse dispositivo, que possui sensores variados, grande abrangência e preço socialmente acessível, é certamente um grande motivador.

Em terceiro lugar, há a motivação técnica. Elaborar o método, considerando a segmentação dos gestos, extração de características, representação do sinal, sua aplicação em Modelos Escondidos de Markov, representa uma contribuição interessante.

Finalmente, há a motivação científica: podemos avaliar o impacto do uso de poucos exemplares no treinamento dos modelos. Soma-se a isso a necessidade de identificar uma forma de combinar de forma eficaz as diversas informações – que motivou a pesquisa de métodos de otimização e abordagens multi-classificadores.

1.3. Questionamentos

Com base nos problemas, levantam-se os seguintes questionamentos.

- Como combinar a informação da mão à informação 3D do esqueleto, capturados via Kinect, em modelos a serem utilizados em gestos armazenados em vídeos?

- Qual o impacto do número de amostras de treinamento na acurácia do método de reconhecimento baseado nesses modelos?

1.4. Objetivos

Nas próximas sessões será descrito o objetivo geral e os objetivos específicos a serem focados durante a pesquisa.

1.4.1. Objetivo geral

Desenvolver um método para reconhecimento de sinais de Libras capturados com o Kinect e relacionados a um vocabulário restrito, previamente definido. Espera-se reconhecer o sinal, independentemente do ator, e fornecer uma lista de palavras candidatas.

1.4.2. Objetivos específicos

Para atingir o objetivo geral proposto, foram elencados os seguintes objetivos específicos:

- Gravar uma base de vídeos contendo os nove gestos diferentes da Libras, gesticulados por diferentes atores;
- Avaliar o impacto causado pelo tamanho da base de vídeos e procurar estratégias para otimizar seu uso;
- Utilizar o algoritmo de rastreamento do Kinect para obter a posição dos braços, antebraços e mãos;
- Construir um classificador para identificar as configurações de mão;
- Combinar informação de movimento e de configuração da mão para fornecer a lista das possíveis palavras candidatas;
- Medir o impacto pela escolha do tamanho do número de amostras no treinamento dos modelos;
- Definir um método para avaliação dos resultados.

1.5. Contribuições

Para a realização dos trabalhos, uma base de dados será gravada, utilizando o hardware Kinect da Microsoft. Este equipamento foi escolhido por ser de baixo custo e alta disponibilidade, além de fornecer informações em cores, profundidade e detectar automaticamente pontos do esqueleto.

O trabalho também apresentará como contribuição a definição de um método para classificação dos sinais gesticulados, onde serão descritos:

a) Como algoritmos de detecção de pontos de interesse invariantes a escala podem ser combinados a informação de esqueleto para formar um vetor de características interessante para o problema;
b) Como melhorar os resultados obtidos pelos modelos escondidos de Markov através da calibração de seus parâmetros, e do uso de técnicas como treinamento e testes *leave-one-out* e validação cruzada;
c) Qual impacto causado pelo tamanho da base de dados, a medida que mais amostras são retirados do conjunto de testes e inseridos no conjunto de treinamento.

1.6. Estrutura do trabalho

Este trabalho está dividido da seguinte forma. O capítulo 1 apresenta a introdução, objetivos e motivação para o trabalho. No capítulo 2, apresenta-se a fundamentação teórica, que descreve os principais algoritmos e softwares usados no decorrer do trabalho. No capítulo 3 apresenta-se o estado da arte, listando os principais trabalhos similares realizados na área. O capítulo 4 detalha o método proposto. O capítulo 5 descreve a base gerada e os resultados obtidos. E, finalmente, o capítulo 6 apresenta as conclusões e sugestões de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta uma breve introdução à fundamentação teórica relevante ao método proposto nessa dissertação. Os Modelos Escondidos de Markov (HMM), que são usados na identificação da palavra gesticulada são discutidos na seção 2.1, incluindo seus conceitos principais, tipos de HMMs e os algoritmos necessários para programar essa técnica estatística na modelagem de problemas reais. Na seção 2.2 um processo de quantização é apresentado, que foi utilizado para mapear os vetores de características em valores discretos. Este processo, realizado através do algorítmo k-means, foi utilizado tanto na inclusão das características da forma da mão no vetor de movimento, quanto no vetor de características final, antes de ser submetido aos HMMs. Na seção 2.3 é descrita a técnica de treinamento e testes leave-one-out, utilizada para gerar sequencias secundárias de observações, evitando o impacto de ruídos sobre a base. Já na seção 2.4, são descritos dois algoritmos de extração de características invariantes a escala, intitulados SIFT e SURF. Estes foram utilizados para extrair pontos relevantes do contorno da mão. Finalmente, a sessão 2.5 faz um sumário de outros equipamentos e bibliotecas utilizadas durante a pesquisa.

2.1. Modelos Escondidos de Markov (HMM)

A motivação original para o uso de HMMs para o reconhecimento do gesto foi baseada em seu uso já consagrado no reconhecimento de palavras faladas, no campo do Reconhecimento de Voz [Rab89]. Desde então, seu uso em vídeo tem sido bastante difundido, como exemplo, nos trabalhos de Boreckzy [Bor98] e, mais recentemente Zafrulla [Zaf11].

Uma revisão detalhada desse tipo de modelagem estatística e suas aplicações podem ser encontradas em [Rab89]. Já em [Por88] o conceito básico foi introduzido através de exemplos simples, e o algoritmo de Baum-Welch para estimar a máxima verossimilhança do modelo foi discutido de um ponto de vista formal e intuitivo.

Uma cadeia de Markov é um método para modelar um sinal como uma sequência de saídas observáveis, produzidas por um processo chamado *origem* ou *Origem de Markov*. Nessa origem, os valores produzidos são dependentes apenas de um número fixo de símbolos, que já haviam sido produzidos anteriormente pela mesma origem. O número de símbolos precedentes levados em consideração para estimar o próximo símbolo produzido define a *ordem* do modelo de Markov. Apenas modelos de primeira e segunda ordem têm sido usados na maior parte das aplicações, pois a complexidade do modelo cresce exponencialmente com o aumento da ordem. Este trabalho se foca em modelo de Markov de primeira ordem.

Um HMM tem a mesma estrutura de uma cadeia de Markov, com a diferença de que cada estado de uma cadeia de Markov representa um único símbolo da observação, que corresponde a um evento físico observável. Numa HMM, cada estado representa uma probabilidade sobre todos os outros símbolos.

Dessa forma, numa cadeia de Markov é possível, dado uma sequencia de símbolos produzidos por um modelo, computar a sequencia de estados que o produziram. Entretanto, na maioria dos problemas reais, cada estado do modelo não pode ser definido por um único símbolo, pois mais de um símbolo pode ser observado por estado.

Para estender a cadeia de Markov para lidar com esses problemas, os estados num modelo HMM são intitulados “escondidos”, isto é, cada estado é modelado como uma distribuição probabilística de todos sobre todos os símbolos. Assim, diferentemente de uma cadeia de Markov, um HMM não pode fornecer a sequencia exata de estados

que produziram uma determinada sequencia de símbolos, mas é possível computar a sequencia de estados com a maior probabilidade de tê-lo feito.

A notação abaixo descreve o conjunto completo de parâmetros de um HMM discreto de primeira ordem:

$$\lambda = (N, M, T, A, B, \pi) \tag{2-1}$$

Onde:

N: Representa o número de estados distintos no modelo. O conjunto de estados pode ser escrito como:

$$S = \{S_1, S_2, \dots, S_n\} \tag{2-2}$$

e q_t indica o estado no tempo t.

M: Representa o número de símbolos observados distintos por estado (tamanho do alfabeto), que representam a saída física do modelo. No caso desta dissertação, os símbolos são cada uma das nove palavras da Libras a serem gesticuladas. O conjunto dos símbolos individuais pode ser representado por:

$$V = \{v_1, v_2, \dots, v_m\} \tag{2-3}$$

T: O tamanho da sequencia de observações. A sequencia pode ser denotada por:

$$O = \{o_1, o_2, \dots, o_t\} \tag{2-4}$$

onde o_t representa a observação no tempo *t*.

A: Distribuição de probabilidade de uma transição de estado. Onde A = $\{a_{ij}\}$, tal que:

$$a_{ij} = P\left[q_t + 1 = s_j \middle| qt = s_i\right], \qquad 1 \leq i, j \leq N \tag{2-5}$$

ou seja, a probabilidade do estado s_j ocorrer no tempo (*t+1*) dado o estado s_i no tempo *t*. Para cumprir os requisitos estocásticos padrão, os elementos em *A* devem apresentar as seguinte propriedades:

$$a_{ij} \geq 0, \qquad \forall j, i \tag{2-6}$$

$$\sum_{j=1}^{N} A_{ij} = 1, \qquad \forall i \tag{2-7}$$

B: distribuição de probabilidade do símbolo observado no estado *j*. $B = \{b_j(k)\}$, tal que:

$$B_j(k) = P\left[O_t = v_k \middle| q_t = s_j\right], \qquad 1 \leq k \leq M, 1 \leq j \leq N. \tag{2-8}$$

isto é, a probabilidade do símbolo v_k ser observado no tempo t, dado o estado s_j no tempo *t*. b_k deve ter as seguintes propriedades:

$$B_j(k) \geq 0 \tag{2-9}$$

$$\sum_{k=1}^{M} B_j(k) = 1. \tag{2-10}$$

π: Distribuição inicial do estado. $\pi = \{ \pi_i \}$, tal que:

$$\pi_i = P[q_1 = s_i], \qquad 1 \leq i \leq n \tag{2-11}$$

é a probabilidade que o estado inicial seja igual a s_j. Similarmente, essa probabilidade deve ser não negativa e:

$$\sum_{i=1}^{N} \pi_i = 1. \tag{2-12}$$

2.1.1. Tipos de HMMs

Existem dois tipos importantes de HMMs: ergódigos e esquerda-direita (modelo Bakis) [Rab89]. O modelo ergódigo é um caso específico de um modelo totalmente conectado, de forma que todos os a_{ij} são positivos. Nesse tipo de modelo os estados são conectados de forma que qualquer estado pode ser alcançado a partir de qualquer outro estado. A Figura 2-1(b) mostra um modelo ergódigo de quatro estados.

O modelo esquerda-direita apresenta um tipo importante de interconexão de estado para o reconhecimento de vídeo com a seguinte propriedade:

$$a_{ij} = 0, \qquad j < i. \tag{2-13}$$

Essa propriedade significa que não são permitidas transições para estados cujos índices são inferiores ao do estado atual, propriedade esta interessante para modelar sinais que variam com a passagem do tempo. A Figura 2-1 (a) mostra um modelo esquerda-direita de 4 estados. Como a sequencia de estados deve começar no estado 1, e deve terminar no estado *N*, as probabilidades iniciais tem a seguinte propriedade:
$\pi_i=0$, quando $i \neq 1$ e $\pi_i=1$, quando $i = 1$.

Ocasionalmente, limitações adicionais são impostas aos modelos esquerda-direita, tais como:

$$a_{ij} = 0, \qquad j > i + \Delta. \tag{2-14}$$

para evitar mudanças muito grandes nos índices dos estados. Δ é um valor usado como limitador para os saltos. Por exemplo, na Figura 2-1 (a) o valor de Δ é 2, portanto, nenhum salto com mais de 2 estados é permitido.

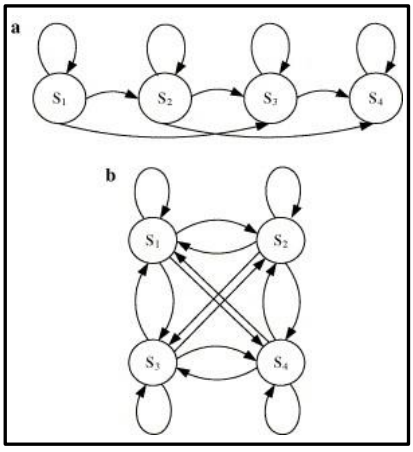


Figura 2-1: Tipos de HMM. a) Modelo Esquerda/Direita b) Modelo Ergódigo

2.1.2. Densidades de observação discretas, contínuas e semi-contínuas

Como mencionado anteriormente, cada estado HMM possui uma distribuição de probabilidade associada ao símbolo observado (parâmetro *B*), que descreve a probabilidade do símbolo v_k ser observado naquele estado. Num modelo de HMM discreto, a probabilidade de se observar um símbolo v_k enquanto no estado s_j, denotado por $b_j(k)$, é definido através da distribuição computada o conjunto de todos os possíveis símbolos no sistema. Essa distribuição é não-paramétrica e quantificada, isto é, não necessita de um conhecimento *a priori* da forma da distribuição para modelar o sinal, e cada observação pode assumir apenas um valor discreto de um conjunto finito. Por consequência, basta existir dados de treinamento suficientes para modelar o sinal. Entretanto, para a maior parte das aplicações, as observações são sinais contínuos. Nesses casos, um HMM discreto só pode ser usado após um processo de quantização do sinal, com a finalidade de criar um *codebook*. O custo desse processo é gerar distorção, levando ao risco de degradar o desempenho significativamente. Além disso, ao adicionar uma nova classe ao sistema, é necessário reconstruir o codebook e, por consequência, refazer o treinamento de todos os modelos.

O uso de HMMs contínuos torna possível evitar a distorção causada pela quantização e retreino do sistema, uma vez que o codebook não existirá. Porém, existem alguns custos: uma função de densidade de probabilidades contínua deve ser definida *a priori* e mais dados de treinamento são necessários para uma estimativa de parâmetros HMM precisa. Uma função de densidade mista, ponderada usando a soma do número de distribuições paramétricas tem sido usada para localizar a melhor maneira de refletir a distribuição das observações.

$$b_j(\underline{x}) = \sum_{m-1}^{M} c_{jm} b_{jm}(\underline{x}). \qquad (2\text{-}15)$$

Onde c_{jm} é o peso para a m-ésima mistura no estado s_j e M é o número de componentes na mistura.

HMMs semi-contínuos, ou SCHMMs, podem ser usados para evitar as distorções causadas pela quantização dos sinais contínuos e ao mesmo tempo reduzir a

quantidade de dados e a complexidade computacional necessária para treinar um HMM contínuo. O conceito por trás desse modelo é de que podem existir similaridades nos dados entre as observações que não representam a mesma origem. Isso significa que agrupar os dados de uma forma não supervisionada pode criar *clusters* com classes que se interceptam em seus limites. Podemos representar qualquer distribuição de estado como uma combinação ponderada de protótipos Gaussianos conhecidos como densidades semi-contínuas.

Nos SCHMMs, o codebook da quantização de vetores (VQ) usado para modelar o sinal contínuo com HMMs discretos é representado por um conjunto de funções de probabilidade de densidade contínuas, cujas distribuições se sobrepõem. Nesse codebook, cada palavra-chave pode ser representada por uma função de densidade de probabilidade contínua. Então, a operação VQ produz valores de funções de densidade de probabilidade contínuas $f\left(x \mid v_j\right)$ para todas as palavra chave v_j.

A função de probabilidade de saída semi-contínua pode ser considerada uma função de probabilidade de densidade mista, com *K* palavras-chave no *codebook* sendo misturadas usando o parâmetro *B* do HMM como coeficiente de peso, como descrito a seguir:

$$B(\underline{x}) = \sum_{k=1}^{K} f\left(\underline{x}\middle|v_k\right) b_j(k). \qquad \text{(2-16)}$$

O problema com esse método é que ele foca em agrupar observações similares, e não observações que fornecem informação discriminatória.

[Hua93] fez um estudo comparativo interessante desses tipos de densidades de observação. Dadas às vantagens e desvantagens de cada tipo de densidade de observações, foi decido utilizar um HMM discreto, de modo a garantir que o fato da base de dados não ser muito grande traria menor impacto possível.

Além disso, outras técnicas de redução de ruído como o teste e treino leave-one-out, descrito na sessão 2.3 poderiam ser utilizadas.

2.1.3. Treinamento dos HMMs

Existem formas diferentes de treinar um HMM. Uma descrição detalhada dos critérios de treino pode ser encontrada em [Rab89]. O critério mais comum é chamado de proximidade máxima (ML). Nesse critério, dado uma base de dados de treino composta de sequencias de observações, os parâmetros dos HMMs são inicialmente inicializados e então iterativamente re-estimados, de modo que a proximidade do modelo produzido pelas sequencias de treinamento aumente. O treinamento para quando a proximidade alcança o valor máximo.

A estimativa de ML foca em maximizar a proximidade de um modelo individual produzindo observações da origem que é modelada, mas não leva em consideração a proximidade de modelos competindo e produzindo as mesmas observações. Isso pode gerar a fronteiras de decisão não-ótimas.

Uma alternativa é usar o critério de Informação Mútua Máxima (MMI). Nesse caso, um conjunto de modelos é treinado para maximizar a habilidade de cada modelo discriminar entre as observações geradas por si mesmo e aquelas geradas por outros modelos. Esse critério é usado para distinguir o modelo correto de todos os outros modelos na sequencia de treinamento. Entretanto, uma solução analítica para esse problema não é viável.

Uma terceira opção é o critério de Informação Mínima Discriminatória (MDI), que é usado quando o sinal a ser modelado não é necessariamente gerado por uma fonte Markov. Esse critério minimiza a intra-entropia entre um conjunto de sinais de densidades de probabilidade válidas e o conjunto de probabilidades HMM. Infelizmente, obter esse mínimo não é nada trivial.

Neste trabalho, utiliza-se o algoritmo de Baum-Welch (BW) para treinar os modelos de palavras. O algoritmo é baseado no critério ML. O primeiro passo consiste em calcular P(*O* | *l*), isto é, a probabilidade da sequencia de observação O, dado o modelo *l*. A solução eficiente para calcular $P(O \mid l)$ é conhecido como procedimento *forward* (para frente). Considere a variável *forward* $\alpha_t(i)$, definida como:

$$\alpha_t(i) = P(o_1, o_2, \ldots, o_t, qt = si \mid \lambda) \qquad (2\text{-}17)$$

que representa a probabilidade da sequencia de observações parciais o_1, o_2, ... o_t, e o estado *si* no tempo t, dado o modelo λ. O algoritmo para induzir $\alpha_t(i)$ é descrito abaixo:
Inicialização:

$$\alpha_1(i) = \pi_i b_i(o_1), \qquad 1 \leq i \leq n \tag{2-18}$$

Indução:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(j) a_{ij}\right] b_j(o_{t+1}) \tag{2-19}$$

Terminação:

$$P(O\,|\lambda\,) = \sum_{i=1}^{N} \alpha_T(i) \tag{2-20}$$

O objetivo do algoritmo de Baum-Welch é ajustar os parâmetros de modelo λ=(A,B,π) para maximizar P(O | λ). Esse é o problema mais complexo no domínio do HMM. O BW é um algoritmo iterativo baseado nas probabilidades *posteriores (forward),* anteriormente definidas, e nas predecessoras (*backward*).

A variável *backward* $\beta_t(i)$, similarmente a variável *forward* $\alpha_t(i)$ é definida como:

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \dots, o_T | q_t = s_i, \lambda) \tag{2-21}$$

que é a probabilidade de uma sequencia parcial de observações a partir de *t+1* até o fim, dado o estado s_i, no tempo *t* e no modelo λ. O algoritmo para a indução de $\beta_t(i)$ é descrito a seguir:

Inicialização:

$$\beta_T(i) = 1, \qquad 1 \leq i \leq N \tag{2-22}$$

Indução:

$$\beta_T(i) = \sum_{j=1}^{N} a_{ij} b_j(o_t + 1)\beta_{t+1}(j)\,, \qquad \begin{matrix} t = T-1, T = 2, \dots 1, \\ 1 \leq i \leq N. \end{matrix} \tag{2-23}$$

Depois de definir a variável *backward*, é possível definir a probabilidade de estar no estado s_i ao tempo *t*, e no estado s_j no tempo *t+1*, dado o modelo e a sequencia de observação, denotada por:

$$\xi(i,j) = P\left(q_t = s_i, q_{t+1} = s_j \middle| O, \lambda\right) \tag{2-24}$$

e é calculada usando:

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O \mid \lambda)}. \tag{2-25}$$

Agora, considere $\gamma_t(i)$ como a probabilidade de estar no estado s_i no tempo *t*, dada a sequencia de observações O, e o modelo λ:

$$\gamma_t(i) = P(q_t = s_i | O, \lambda), \tag{2-26}$$

Que pode ser calculada usando as variáveis *forward* e *backward* como:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda),} \tag{2-27}$$

Dessa forma, o número esperado de transições de s_i é denotado por:

$$\sum_{t=1}^{T-1} \gamma_t(i) \tag{2-28}$$

e o número esperado de transições de s_i até s_j é denotado por

$$\sum_{t=1}^{T-1} \xi_t(i,j) \tag{2-29}$$

Usando o conjunto de fórmulas acima, é possível ter um método para re-estimar os parâmetros π, A e B de um HMM. O conjunto de formulas é constituído por:

1) Frequência esperada do estado s_i no tempo *t*=1

$$\pi_i = \gamma_i(i). \tag{2-30}$$

2) Transição coeficiente = número de transições esperadas do estado s_i ao s_j, dividido pelo número esperado de transições em s_i:

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{t-1} \xi_t(i,j)}{\sum_{t=1}^{t-1} \gamma_t(i)} \tag{2-31}$$

3) Probabilidade do símbolo observado = número esperado no estado *j*, enquanto observando o símbolo v_k, divididos pelo numero esperado de vezes no estado *j*:

$$\bar{b}_j(k) = \frac{\sum_{t=1,s.tO_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{t-1} \gamma_t(j)} \tag{2-32}$$

2.1.4. Avaliação dos HMMs

A avaliação de um HMM no método proposto é feito pelo algoritmo de Viterbi [Rab93]. O algoritmo de Viterbi é um método de programação dinâmica para estimar a sequencia de estados do modelo com a maior probabilidade de ter produzido uma sequencia de observações, ou seja, dada a sequencia de observações $O = \{o_1, o_2, ..., o_T\}$, e o modelo *l*, esse método encontra a sequencia de estados correspondente $Q = \{q_1, q_2, ..., q_T\}$, que melhor explica as observações. Considere as seguintes variáveis:

- $\delta_t(i)$: Avalia a probabilidade da sequencia de observação o_1, o_2, ... , o_t ter sido produzida pela sequencia mais provável de estados do modelo, que termina no estado *i* no tempo *t*.
- $\Psi_t(i)$: vetor usado para rastrear o caminho de máxima verossimilhança. Mantém o registro de estados que maximizaram a proximidade de 1 até *t*.

O algoritmo de Viterbi é descrito a seguir:
Inicialização:

$$\delta_1(i) = \pi_i b_i(o_1), \qquad 1 \leq j \leq N \tag{2-33}$$

$$\psi_t(j) = 0. \tag{2-34}$$

Recursão:

$$\delta_t(j) = \max_{1 \leq i \leq n}\left[\delta_{t-1}(i) a_{ij}\right] b_j(o_t), \qquad \begin{matrix} 2 \leq t, \leq T \\ 1 \leq j \leq N \end{matrix} \tag{2-35}$$

$$\psi_t(j) = \arg\max_{1 \leq i \leq n}\left[\delta_{t-1}(i) a_{ij}\right], \qquad \begin{matrix} 2 \leq t, \leq T \\ 1 \leq j \leq N \end{matrix} \tag{2-36}$$

Término:

$$P^* = \max_{1 \leq i \leq n}[\delta_t(i)], \tag{2-37}$$

$$q_t^* = \arg\max_{1 \leq i \leq n}[\delta_t(i)] \tag{2-38}$$

Backtracking para a sequencia de estados:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), t = T - 1, \qquad T - 2, \ldots, 1. \tag{2-39}$$

No método proposto o algoritmo de Viterbi é usado para isolar a classificação do palavra gesticulada no estágio de Verificação. O objetivo é estimar a probabilidade P^* de cada modelo ter gerado a palavra em questão, e derivar a partir daí uma lista ordenada de melhores candidatos.

2.2. Processo de quantização de vetores

Para usar os HMMs descritos na seção anterior, é necessário representar os vários classificadores escolhidos como uma sequencia de observações discretas. Para esse fim, a extração de características do método proposto – que gera números reais e vetores de amplitude contínua – precisa ser quantizada para um número de símbolos discretos disponíveis num *codebook* previamente elaborado.

Para criar esse *codebook*, é necessário aplicar o conceito de Quantização de Vetores (VQ) [Lin80] [Mak85].

Vamos supor que $\vec{x}$ [x1, x2, ... xn] é um vetor N dimensional onde os componentes $\{x_{k,}1 \leq k \leq n\}$ são variáveis de valores reais e amplitude contínua. Na quantização de vetores, o vetor $\vec{x}$ é mapeado em outro vetor $\vec{y}$ de valores reais, de amplitude discreta N dimensional. É usual dizer que $\vec{x}$ foi quantizado como $\vec{y}$, e $\vec{y}$ é o valor quantizado de $\vec{x}$. Isso pode ser denotado por:

$$\vec{y} = q(\vec{x}) \tag{2-40}$$

onde q é o operador de quantização e $\vec{y}$ é o vetor de saída correspondente a $\vec{x}$. O valor de $\vec{y}$ é um conjunto finito de valores $\{x_{i,}1 \leq i \leq L\}$, tal que $\vec{y}$=[y_{i1}, y_{i2}, ... , y_{in}]. L é o tamanho do *codebook* (ou número de níveis), e Y={y_i} é o conjunto de vetores de código. Para projetar o codebook, particiona-se o espaço N-dimensional do vetor $\vec{x}$ em L regiões ou células, e associamos cada célula C_i ao vetor $\vec{y}_i$. O algoritmo de quantização então associa o vetor $\vec{y}_i$ se $\vec{x}$ estiver em C_i.

$$q(\vec{x}) = \vec{y}_i, \qquad se\ \vec{x} \in C_i \tag{2-41}$$

O mapeamento de $\vec{x}$ em $\vec{y}$ resulta num erro de quantização, e uma medida de distorção *d(x,y)* pode ser definida entre eles, também conhecida como dissimilaridade.

$$d(\vec{x},\vec{y}) = \frac{1}{N}\sum_{k=1}^{N}(x_k - y_k)^2 \tag{2-42}$$

A quantização ótima ocorre quando a distorção é minimizada sobre todos os L-níveis quantificados. Existem duas condições necessárias para a otimização. A primeira é que o quantificador ótimo é obtido usando a regra de seleção da distorção-mínima ou vizinho mais próximo:

$$q(\vec{x}) = \vec{y}, \qquad \leftrightarrow d(\vec{x},\vec{y}_i), j \neq i, 1 \leq j \leq L \tag{2-43}$$

Isso significa que o algoritmo de quantização seleciona o vetor de códigos que resulta no mínimo de distorção com respeito a $\vec{x}$.

A segunda condição necessária para a otimização é que cada vetor código $\vec{y}_i$ deve ser escolhido de modo a minimizar a distorção média na célula C_i. A distorção média D_i é dada por:

$$D_i = \frac{1}{M_i}\sum_{x\in C_i} d(\vec{x},\vec{y}_i) \tag{2-44}$$

O vetor que minimiza a distorção da célula C_i é chamado de centroide de C_i, e é detonado por:

$$\vec{y}_i = cent(C_i) \tag{2-45}$$

Um método bastante conhecido para a criação de um *codebook* é o algoritmo de *clustering* iterativo conhecido como k-means [Mak85], sendo k=L (tamanho do codebook).

O algoritmo divide um conjunto de vetores de treinamento {x(n)} em L clusters C_i de modo que as duas condições de otimização sejam satisfeitas. Na descrição do algorítmo, *m* é o índice da iteração e $C_i(m)$ é o i-ésimo *cluster* da iteração *m*, com $y_i(m)$ sendo seu centroide. O algoritmo é representado a seguir:

Inicialização:	Defina *m*=0. Escolha um conjunto de vetores código $y_{i,} 1 \leq i \leq L.$
Classificação:	Classifique o conjunto de vetores de treinamento $\{x(n), 1 \leq n \leq M\}$ nos clusters C_i pela regra do vizinho mais próximo: $x \in C_y(m) \leftrightarrow d(\vec{x}, \vec{y}_i), j \neq i, 1 \leq j \leq L.$
Atualização do vetor de código:	$m = m+1$. Atualize o vetor de código de cada cluster computando o centroide dos vetores correspondentes de treinamento de cada cluster: $\vec{y}_i(m) = cent(C_i(m)), 1 \leq i \leq L$
Término:	Se o decréscimo na distorção geral $D(m)$ na iteração m relativo a $D(m-1)$ for menor do que um determinado limiar, pare. Caso contrário, retorne ao passo de classificação.

2.3. Treinamento e testes Leave-one-out

O processo chamado treinamento e testes *Leave-One-Out*[1], proposto por [KoA09], que realiza a geração de uma sequencia secundária de observações para cada sequencia principal, reduzindo significativamente os ruídos durante o treinamento dos HMMs.

O processo consiste em, para cada sequencia de observações, gerar uma nova sequência sem uma das observações:

Sequência original	$O = \{o_1, o_2, o_{3,} o_4, o_5\}$

[1] Este processo não deve ser confundido com a técnica de validação cruzada de mesmo nome também utilizada nesse projeto (*leave one out k-folding*), por isso, mantivemos o termo em inglês neste caso, e optamos pela tradução "deixe de um de fora" ao nos referirmos à validação.

Sequências secundárias	$O_1 = \{o_2, o_{3,} o_4, o_5\}$ $O_2 = \{o_1, o_{3,} o_{4,} o_5\}$ $O_3 = \{o_1, o_{2,} o_4, o_5\}$ $O_4 = \{o_1, o_{2,} o_3, o_5\}$ $O_5 = \{o_1, o_{2,} o_3, o_4\}$

Figura 2-2: Sequencias Leave One Out

Suponha que, para qualquer sequencia O_i, possuímos uma série de observações, $o_1, o_2 \ldots o_{t_i}$, sendo T_i é o número de observações para a sequencia O_i. Para essa sequencia, geramos uma nova simplesmente removendo uma observação por vez, como exemplificado na Figura 2-2. Assim, quanto mais longa a sequencia de observações, mais sequencias secundárias serão geradas.

Considerando um conjunto de $O = \{oi, 1 \leq i \leq N\}$ onde N é o número de sequencias e cada sequencia tem T_i observações, uma sequencia original pode gerar T_i novas sequencias, por consequência, para todas as N sequencias o número total será de:

$$\sum_{1 \leq i}^{N} T_i \tag{2-46}$$

Para o treino HMM, todas as novas sequencias, assim como as sequencias originais, são usadas, o que permite-nos obter um total de

$$\left(\sum_{1 \leq i}^{N} T_i\right) + N \tag{2-47}$$

sequencias de treinamento. As sequencias secundárias podem reduzir ou induzir ruídos no treinamento. Se reduzir, teremos uma amostragem mais confiável para o treinamento. Se induzir, os HMMs serão treinados para reconhecer as sequencias de ruído durante os testes.

O mesmo processo é realizado durante os testes. Após a geração de todas as T_i sequencias, tanto a sequencia original, quanto as secundárias, são submetidas para os testes. Assim, cada sequencia pode resultar na classificação de uma classe diferente.

É necessário traçar uma estratégia para combinar os resultados e obter-se um veredito final.

[KoA09] cita duas estratégias:

- A primeira consiste em agrupar as sequencias por classe de resultado. Então, em cada classe, gerar uma nova probabilidade baseada na soma das sequencias associadas àquela classe, similarmente ao que é feito no operador de SOMA de um sistema multiclassificadores. Nessa abordagem parte do pressuposto que todas as classificações vão realizar diversos erros, e que esses erros podem ser minimizados pela combinação das probabilidades, ou pelo voto. Essa abordagem é chamada de HARD, pois todas as amostras tem impacto sobre o resultado final.
- A segunda consiste em selecionar diretamente a classe com o melhor índice de proximidade entre todas as sequencias. Nos HMMs tradicionais, o objetivo é obter a classe V_j com a maior probabilidade entre todas as classes $V_j, 1 \leq j \leq M$. Como o teste gera T_i novas sequencias para classificação e como todas as observações tem a mesma probabilidade de ser ruído, podemos considerar que todas as sequencias tem a mesma importância. Ao selecionar a classe com a maior probabilidade de ter sido criada pelo modelo, esse método permite que uma sequencia pule qualquer observação individual para atingir a maior proximidade possível. Assim, ao comparar a maior probabilidade de cada classe, esse método pode associar a melhor classe a sequencia levando em consideração que uma observação pode ser ruído. Este processo é chamado de abordagem SOFT, e foi utilizado no decorrer deste trabalho.

É importante ressaltar que existem duas premissas no teste com *leave-one-out*. Primeiro, parte-se do pressuposto que é impossível distinguir o ruído de um sinal normal numa sequencia de observações. Assim, cada observação é tratada com chance igual de ser ruído. Segundo, parte-se do pressuposto que a sequencia é, na maior parte das vezes, reconhecível sem uma das suas observações – no caso deste trabalho, é importante notar que uma observação representa um quadro de vídeo.

2.4. Algoritmos de segmentação invariantes a escala

Um dos problemas de extração de características numa imagem é a variação de escala. Ou seja, imagens diferentes sobre um mesmo objeto podem ter sido retiradas a distâncias diferentes da câmera, seja pela movimentação do objeto, ou da câmera em si.

Considere o problema de extrair o contorno geral da mão do ator. Em cada tomada de vídeo, um ator pode se posicionar ligeiramente à frente ou atrás de outro. Os atores também possuem mãos de tamanhos diferentes e a movimentam no decorrer da cena.

Para solucionar esse problema, dois algoritmos têm sido citados na literatura. O primeiro, chamado Scale Invariant Feature Transform (SIFT) [Low04], baseado em filtros laplacianos, mais preciso e mais lento, usado nesse trabalho. O segundo, chamado Speed Up Robust Features (SURF) [Bay04], apresenta melhor performance.

As derivadas de uma imagem podem ser estimadas utilizando filtros gaussianos. Filtros Gaussianos 1D apresentam o seguinte formato:

$$G(x) = \Gamma e^{\frac{-x^2}{2\sigma^2}} \tag{2-48}$$

Onde Γ representa o coeficiente de normalização, que é escolhido de modo que a soma dos diferentes pesos seja um. O σ (sigma) representa o tamanho da função gaussiana. Quanto maior o seu valor, mais plana a função será. Esta variável também representa implicitamente a escala sobre a qual a função é avaliada.

Ao calcular a transformada Laplaciana de um determinado ponto de uma imagem, usando filtros Gaussianos em diferentes escalas, valores diferentes são obtidos. Olhando para a resposta da evolução do filtro para os diferentes fatores de escala, obteremos uma curva que em algum momento atinge um pico máximo. Se obtivermos dois máximos de duas imagens diferentes, veremos que a proporção entre eles equivale a proporção de tamanho entre as duas imagens.

O SURF implementa esse conceito através do seguinte algoritmo:
Inicialmente, para detectar características, a matriz Hessiana é calculada a cada pixel. Essa matriz mede a curvatura local de uma função e é definida como:

$$H(x,y) = \begin{bmatrix} \frac{\delta^2 f}{\delta x^2} & \frac{\delta^2 f}{\delta x \delta y} \\ \frac{\delta^2 f}{\delta x \delta y} & \frac{\delta^2 f}{\delta y^2} \end{bmatrix} \quad (2\text{-}49)$$

No caso de processamento de imagens, *f* normalmente corresponde a iluminação do pixel I, ou seja, ao valor do tom de cinza sendo processado.

O determinante dessa matriz dá a intensidade da curvatura. Assim, é possível determinar os cantos da imagem com maior curvatura (isto é, maior variação em uma direção do que em outra). Como é composta por derivadas de segunda ordem, essa matriz pode ser calculada usando-se *kernels* laplacianos de diferente escala σ. Isso torna a matriz Hessiana uma função de três variáveis $H(x, y, \sigma)$. Uma característica invariante a escala é determinada quando a máxima dessa função é obtida tanto no campo da escala quanto do espaço.

O calculo de todas essas derivadas é custoso. Como o objetivo do SURF é ser computacionalmente eficiente, os *kernels* são computados com base em aproximações, descritas abaixo:

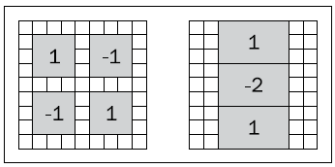


Figura 2-3: Kernels usados no SURF

O *kernel* da esquerda é usado para estimar as derivadas mistas de segunda ordem, enquanto o da direita estima as derivadas de segunda ordem na direção vertical. Para se obter as derivadas na direção horizontal, gira-se o segundo Kernel. Os menores *kernels* tem tamanho de 9x9, correspondendo a um sigma aproximado de 1,2. *Kernels* de tamanho sucessivamente maiores são utilizados.

Uma vez que o máximo local é identificado, a posição precisa de cada ponto de interesse é obtida através da interpolação nos espaços da imagem e de escala. O resultado é então um conjunto de pontos de interesse localizadas com precisão de subpixel, associados a um valor de escala.

O algoritmo de SURF foi desenvolvido como uma variação eficiente do algoritmo de SIFT.

A diferença básica entre os dois é que no lugar do determinante de Hessian, o SIFT utiliza diretamente a transformada Laplaciana. A aproximação do determinante de Hessian resulta num cálculo envolvendo apenas três operações com números inteiros, resultando num processo extremamente eficiente do ponto de vista computacional, porém, menos preciso [Bay04].

Como a base de dados apresenta tamanho reduzido, optou-se por manter o algoritmo de SIFT, já que o impacto no tempo total de processamento foi pequeno, e um impacto positivo sobre a precisão era extremamente desejado.

2.5. Equipamentos e softwares

Nesta seção, descreve-se o conjunto de equipamentos e softwares que deram suporte a execução do método proposto.

2.5.1. Sensor Kinect

A Microsoft lançou em novembro de 2010 um sensor intitulado "Kinect". Inicialmente, seu objetivo era servir de controlador para jogos, no seu vídeo-game, o X-Box, respondendo a concorrência de sensores de movimento de seus concorrentes, como o WiiMote da Nintendo.

Graças à produção em larga escala [Dea12], com 18 milhões de unidades vendidas até janeiro de 2012, o sensor possui custo mais barato que alternativas similares do mercado. Suas vendas também foram impulsionadas pelo fato do sensor ser ligado através da porta USB, o que possibilitou que usuários de PC comprassem o produto, mesmo sem o vídeo-game.

O Kinect possui uma tecnologia desenvolvida pela empresa Rare, subsidiária da Microsoft Game Studios. Seu sensor de profundidade é produzido pela companhia israelense PrimeSense [Mic10], que batizou sua tecnologia de escaneamento 3D de *Light Coding*.

O sensor é formado por um laser infravermelho e um sensor CMOS monocromático, que capta os dados 3D utilizando o método de *time of flight* (cálculo da distância com base no tempo que a luz infravermelha leva para sair do emissor, refletir no objeto e retornar até o sensor). O sensor, na versão original do produto, que será

usada neste trabalho, trabalha com um alcance de 1,2 até 3 metros. Já a versão para PCs trabalha com alcance de 15cm até 2,25 metros. Uma das características interessantes é que o sensor de profundidade não depende de quaisquer condições de iluminação, podendo ser usado inclusive na escuridão total.

O sensor possui 11 bits de precisão, o que o torna capaz de representar 2048 graus de sensibilidade. O valor informado pelo sensor é, para cada pixel da cena, à distância em centímetros a partir da câmera, sendo que o valor zero representa a ausência de sinal.

Além do sensor de profundidade, o Kinect também possui uma câmera RGB, um microfone e uma base móvel. A câmera trabalha a uma taxa de 30 quadros por segundo e sua captura é sincronizada com a do sensor. A resolução da imagem de entrada é de 640x480 pixels e o hardware da câmera utiliza o filtro de Bayer durante o processamento das cores.

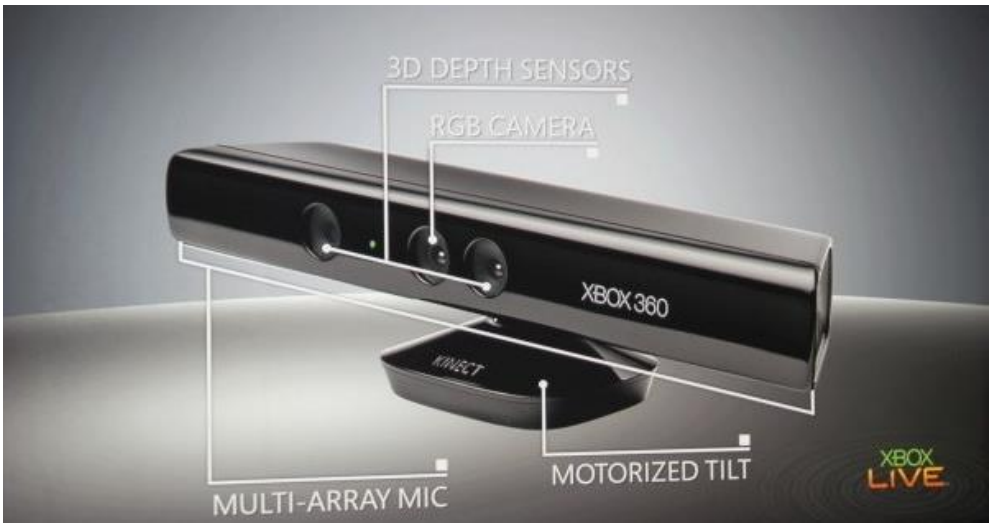


Figura 2-4: Sensor Kinect – Figura divulgada na conferência E3 da Microsoft

A base do sensor é capaz de incliná-lo 27 graus para cima ou para baixo. O microfone possui quatro células de captura, capturando áudio com taxa de amostragem de 16 bits.

A Microsoft também disponibiliza gratuitamente um kit de desenvolvimento para o aparelho. Porém, é de código fechado e disponibilizado apenas para plataforma Windows. Como alternativa, outra biblioteca será utilizada nesse trabalho, intitulada Open Natural Interfaces (OpenNI).

2.5.2. Open Natural Interfaces (OpenNI)

O sensor pode ser integrado à aplicação através de um driver de dispositivo, produzido e disponibilizado publicamente pela própria PrimeSense, associado a uma biblioteca pública chamada Open Natural Interfaces (OpenNI).

A biblioteca possui diversos exemplos, que demonstram como obter as imagens RGB e de profundidade, como gravá-las em disco, como identificar sinais simples como aceno de mão, como rastrear um membro do corpo, ou mesmo gerar o esqueleto completo do usuário.

Na Figura 2-5, vemos um exemplo de imagem de profundidade e RGB capturadas pela biblioteca. A imagem de profundidade foi redistribuída em 256 tons de amarelo, com base no histograma das distâncias:

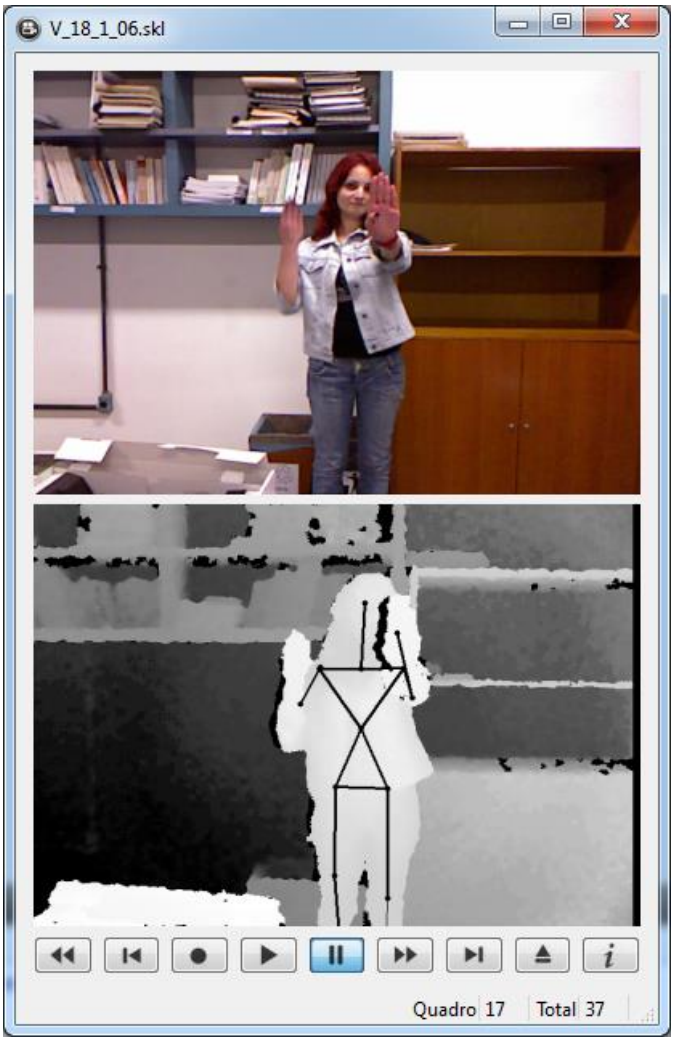


Figura 2-5: Dados de entrada do Kinect

Nesta biblioteca o suporte ao microfone e a base móvel ainda é limitado. Entretanto, opta-se por ela por ser compatível com diversos sistemas operacionais, inclusive o Linux, e por ser facilmente integrada à OpenCV.

Por ser uma plataforma de integração, vale ressaltar que a OpenNI não é restrita ao Kinect. Ela pode ser integrada a outros sensores da PrimeSense ou da ASUS, o que permite adaptar o software facilmente a outros equipamentos, evitando uma dependência exclusiva com a Microsoft.

2.5.3. Open Computer Vision (OpenCV)

A biblioteca Open Computer Vision (OpenCV), compõe uma coleção de algoritmos comuns que auxiliam na tarefa de visão computacional. Esses algoritmos incluem segmentação, obtenção de bordas, limiarização, conversão de espaços de cor, classificadores, detectores de pontos de interesse (SIFT e SURF), entre outros.

A implementação das funções da biblioteca são otimizadas, inclusive com aceleração de hardware ou integração à placa de vídeo.

A biblioteca é disponibilizada em duas versões: OpenCV, feita para a linguagem C, e OpenCV2, feita para o C++.

É na segunda versão que o desenvolvimento atual da biblioteca está focado, por isso, ela foi à escolhida para a confecção desse trabalho.

2.5.4. JAHMM

A biblioteca JAHMM [Fra13] fornece uma implementação do algoritmo HMM. Nela, estão disponíveis os algoritmos de Baum-Welch e Viterbi, assim como a medida de distância de Kullback-Leiber.

A biblioteca dá suporte a HMMs discretas e contínuas e permite a entrada de dados através de arquivos de textos simples em um formato proprietário. É gratuita e está disponível no google code: https://code.google.com/p/jahmm/

2.6. Considerações finais

Neste capítulo, apresentou-se os Modelos Escondidos de Markov, já usados com sucesso nos campos de processamento de palavras, voz e vídeo. Apresentou-se diferentes tipos de HMMs, considerando sua topologia: ergódiga ou esquerda direita, e seu tipo de função de distribuição de probabilidade: seja contínua, semi-contínua ou discreta. Também foi apresentada uma forma de avaliar os HMMs.

Em seguida, apresentou-se um processo de quantização de vetores baseado no algoritmo de k-means. Tal algoritmo permite a transformação de sinais contínuos reais em sinais discretos, necessários para o uso de HMMs discretas.

Também apresentou-se a técnica de treinamento e testes *leave-one-out*, capaz de gerar sequencias de observação artificiais, reduzindo o ruído gerado pelas sequencias de entrada e melhorando a confiabilidade ou resistência a ruídos dos modelos.

Os algoritmos de segmentação invariantes a escala SIFT e SURF, usados para extração de característica da mão, foram introduzidos.

Finalmente, mostrou-se um conjunto de equipamentos e softwares presentes no mercado utilizados no decorrer do trabalho: o sensor Kinect, a biblioteca de captura Open Natural Interfaces, a biblioteca de processamento de imagens OpenCV e a biblioteca JAHMM, responsável pela implementação dos HMMs.

Capítulo 3

Estado da Arte

Nos estudos da relação entre os seres humanos e o computador, é constante a busca por meios naturais de comunicação, que dispensem a presença de equipamentos e que explorem as habilidades naturais dos indivíduos.

Nesse contexto, a mão é objeto de especial interesse, uma vez que possui grande agilidade e é usada naturalmente para manipulação de objetos e para a comunicação.

Neste sentido, as mãos podem ser utilizadas tanto para representar uma forma de controlar o computador [AnH10], quanto para fornecer um mecanismo de entrada, através de algum tipo de linguagem de sinais convencionado [Ant01].

Para a segmentação da mão e identificação do sinal, duas abordagens têm sido utilizadas: a primeira, baseada em visão computacional. Para auxiliar essa tarefa, podem ou não ser empregados marcadores, como luvas coloridas, para simplificar o processo de segmentação. Além disso, é comum o emprego de mais de um sensor. O segundo tipo de abordagem é o uso de sensores especialmente desenhados para a tarefa de captura, tais como braços robóticos ou acelerômetros instalados em luvas.

Em ambos os casos, para realizar o reconhecimento, os dados capturados são identificados através de um classificador. É importante destacar que há duas formas de

reconhecimento comumente encontradas: a primeira é a da trajetória percorrida pela mão e a segunda, da configuração da mão e dos dedos.

A Figura 3-1 sumariza essas abordagens:

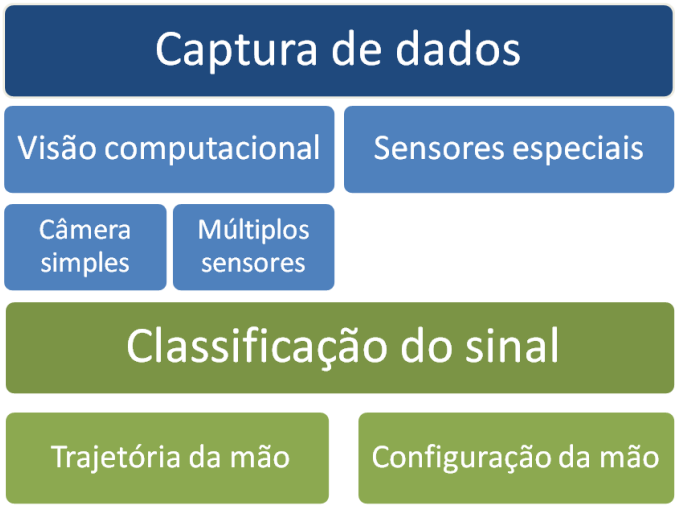


Figura 3-1 – Representação gráfica das abordagens

3.1. Visão computacional

Esta seção tem como objetivo mostrar os principais trabalhos desenvolvidos utilizando-se visão computacional. Inicialmente são apresentados os trabalhos com o uso de uma câmera simples, seguidos de trabalhos utilizando-se múltiplos sensores.

Paulraj [Pau09] elaborou uma base de dados a partir da captura do movimento de duas pessoas, cada uma gesticulando cinco vezes os mesmos 14 gestos na linguagem de sinais da Malásia. Todos os gestos foram capturados numa única câmera de vídeo, com resolução de 24 bits de cor e 320x200 pixels. Todas as imagens foram convertidas para tons de cinza, e tiverem ruído removido através de um filtro da média. Os quadros foram então segmentados em três regiões, identificadas pela mão direita, esquerda e cabeça, e então limiarizados. O autor então combinou a informação de dois quadros consecutivos, realizando sobre eles uma operação de *ou lógico*.

Para geração do vetor de características, cada área segmentada foi considerada como um evento discreto, e suas características extraídas através da transformada discreta de cossenos (DCT). Os 10 primeiros DCTs de cada uma das três áreas diferentes foram considerados como características.

Durante o processo de classificação, uma rede neural artificial foi utilizada. Para tanto, foram utilizadas 98 amostras para treinamento e os resultados foram obtidos através da classificação de 140 amostras. A técnica obteve taxa de acerto de 81,07%.

Carneiro [Car09], utiliza os Momentos Invariantes de Hu como descritores das imagens e algoritmos de classificação clássicos, a cada quadro da imagem. A mão é segmentada de maneira muito simples: a imagem é convertida para o espaço de cor YC_bC_r, e é aplicado um limiar nos canais C_b e C_r. Já os momentos invariantes de Hu descrevem 6 características, que são usadas na classificação da mão. A classificação inicia-se por um processo de aprendizagem não supervisionada usando os mapas auto-organizáveis (*Self-organizing maps, SOM*), pré-classificando as imagens em seis diferentes *clusters*. Para a classificação dentro de um *cluster*, foram testados dois classificadores, os perceptrons simples (PS) e o Perceptron Multicamadas (MLP).

A base de dados utilizada foi gerada na Federação Nacional de Educação e Integração de Surdos (FENEIS - Ce) e da Associação de Pais e Amigos de Deficientes Auditivos (APADA - Ce) em um ambiente de iluminação controlada. Foram utilizadas 50 imagens de três pessoas diferentes para cada um dos 26 gestos da datilologia Libras, totalizando 3900 imagens. O equipamento usado foi uma câmera digital VGA com resolução de 640x480 pixels. Foram obtidas taxas de acerto de 90% com o uso do perceptron simples, enquanto a da rede MLP teve acerto de 89,66%.

Similarmente, Elmezain [Elm08] propõe um sistema para o reconhecimento de gestos arábicos gesticulados em imagens estéreo capturadas por uma câmera chamada Bumblebee. Inicialmente, as mãos são segmentadas através de informações 3D e de cores, extraídas no espaço YC_bC_r em diversas amostras de pele, descartando-se o canal Y. O algoritmo de extração da cor da pele baseia-se num modelo de Gaussiana. A extração de características baseia-se na localização, orientação e velocidade da mão. Da mesma forma que [Dia09], o centroide do pixel da mão é calculado, e sua posição ao longo de vários frames descrevem a onda, que será levada em consideração para determinação do movimento.

Para classificação, foram utilizados os modelos escondidos de Markov (HMM). O autor realizou os testes considerando as topologias: totalmente conectadas, LR e LBR, sendo que a última apresentou o melhor resultado com classificação de 98,4% para gestos isolados e 95,7% para gestos contínuos.

Esse trabalho, no entanto, não descreve como foram gravadas as sequências de vídeo, ou mesmo se mais de uma pessoa foi utilizada no processo. Sua contribuição, está na combinação de informações 3D e de pele para a determinação da trajetória.

Embora o Kinect tenha surgido em 2010, e propiciado a disseminação em massa de sensores de profundidade, a literatura apresenta pesquisas com esse tipo de sensor há vários anos. Em 2007, Breuer [Bre07] apresentou em seu trabalho uma proposta para reconhecimento dos movimentos da mão baseada numa câmera *time-of-flight*. Em seu trabalho, a malha de pixels era submetida a analise PCA, e um modelo da mão era projetado com 7 graus de liberdade. Não foram divulgadas informações sobre a base ou taxas de acerto, mas o trabalho deu origem a uma série de artigos subsequentes, utilizando-se da mesma tecnologia.

Nesta linha, An [AnH10] utilizou uma câmera similar para controlar um jogo de carro. A câmera era montada em frente ao jogador. Como o jogo assume que o jogador está com os braços esticados para a frente a segmentação era realizada considerando as mãos como o objeto mais próximo do sensor. Após a localização desse objeto, uma janela que considera informação de profundidade em um tamanho definido foi usada para separar a mão dos demais elementos. Em seguida, a informação de profundidade foi binarizada, para que o contorno da mão pudesse ser analisado através do algoritmo de Kass. A curvatura dos contornos é usada para identificar a posição dos dedos.

O jogo especificou cinco possíveis comandos: Acelerar, Ré, Pare, Vire à esquerda e vire à direita. Os dois últimos eram obtidos posicionando a respectiva mão à frente da cena. A posição dos dedos indicavam os movimentos de aceleração, por exemplo, dois dedos para acelerar, cinco para ré.

Uma base de dados contendo 2500 imagens de 10 pessoas de diferentes idades, sexos e etnias foi montada, cada uma controlando o jogo de corrida por cinco minutos. Foram feitos três experimentos, para detectar três posições de mão específicas, e o algoritmo obteve sucesso em aproximadamente 98% das vezes.

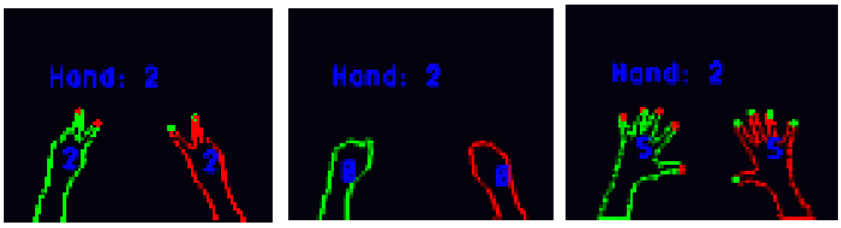


Figura 3-2 - Posições de mão detectadas nos experimentos de An[11]

Embora o vocabulário seja muito restrito, esse artigo demonstra com clareza as vantagens da utilização de um sensor de profundidade: invariabilidade para iluminação, tom de pele, facilidade de separação da informação do fundo e distinção de outras áreas de pele do corpo, como o rosto.

Em 2011, Zafrulla [Zaf11] demonstrou uma versão de seu game CopyCat, para surdos, utilizando o Kinect. Ele utilizou apenas a parte superior do torso e normalizou o esqueleto com base no tamanho dos braços. Sua base de dados era composta por 6 sinais sendo 2 sujeitos, 2 preposições e 2 objetos. Os sinais eram usados em 4 sentenças compostas de sujeito, preposição e objeto, e foram gravadas 20 amostras de cada sentença.

Um dos pontos negativos da pesquisa é que o autor utilizou apenas um único ator em todos os filmes, tornando os modelos extremamente dependentes. Ele indicou que a independência de atores seria desejável, e deveria ser pesquisada em trabalho futuros. Seus testes indicaram taxa de acerto de 98.8%.

A Tabela 3-1 sumariza as abordagens apresentadas:

Tabela 3-1: Sumário das abordagens apresentadas

Artigo	Tipo	Equipamento	Método	Base	Acerto
Paulraj [Pau09]	Configuração, 6 gestos	Câmera RGB	DCT + RNA	98 imagens de treino + 140 testes	81,07%
Carneiro [Car09]	Configuração, 26 gestos	Câmera RGB	Hu + SOM + Perceptron	50 imagens, de 26 sinais, de 3 pessoas = 3900 imagens.	90%
			Hu + SOM + MLP		89,66%
Elmezain [Elm08]	Trajetória com 10 classes	Bumblebee: 2 Câmeras RGB	HMM	200 sequencias para treino, 98 para gestos isolados e 70 para contínuos.	98,4% gestos isolados 95,7% gestos contínuos.
An [AnH10]	Configuração, 5 gestos	Sensor de profundidade	Detecção de contorno	2500 imagens de 10 pessoas diferentes, jogando por 5 minutos	Aprox. 98%
Zafrulla [Zaf11]	Trajetória, 6 classes	Kinect	HMM	80 vídeos contendo 4 tipos de sentenças. 1 único ator.	98.2%

3.2. Sensores especiais

Uma alternativa à visão computacional são sensores especialmente desenhados para a captura do movimento, tal como luvas ou acelerômetros.

Fang [Fan07] utilizou duas luvas Cybergloves e três rastreadores Pohelmus 3SPACE-Position para reconhecer um vocabulário de 5113 sinais chineses. Dois dos rastreadores eram posicionados no pulso, enquanto o terceiro, considerado o rastreador de referência, situava-se atrás da cabeça. As luvas coletavam informação de variação da posição dos dedos, orientação da mão e trajetória, considerando 18 graus de liberdade. Os dados experimentais eram compostos de 51130 amostras, baseando-se em 5113 sinais isolados extraídos da linguagem de sinais chinesa. Esse conjunto, fornecia um vetor de características de 48 dimensões, considerando-se 36 características para a posição dos dedos, 6 para a posição da mão, e outras 6 para a orientação.

Figura 3-3 – Dispositivos de entrada usados por Fang [Fan07]

Foi também utilizado um banco de dados contínuo, com 3000 amostras, considerando-se 750 sentenças diferentes formadas a partir do vocabulário de 5113 sinais.

Todos os dados foram coletados a partir de dois indivíduos, cada um realizando o gesto duas vezes. O autor utiliza um processo de *clustering* de dados usando sequências temporais. As sequencias são geradas a partir do algorítmo K-Means modificado, associado a um algoritmo de *dynamics time warping* (DTM). Em seguida, o autor compara a abordagem de TMM proposta por ele com o algorítmo HMM, obtendo taxa de reconhecimento de 91,9%, contra 78,2% do HMM. O trabalho mostra que é

possível obter altas taxas de reconhecimento, a partir de bons dispositivos de entrada, mesmo com uma quantidade expressiva de vocabulário.

Entretanto, a complexa rede de sensores, que precisa ser instalada e calibrada, associada a sua baixa disponibilidade no mercado desestimulam esse tipo de pesquisa para o uso em linguagens de sinais.

O trabalho de Kim [Kim08] avaliou o uso de sensores mais simples, disponíveis no mercado. No lugar das luvas, combinou-se em cada mão 2 eletromiogramas, comumente usados para medição cardiovascular, e 1 acelerômetro. Todos os produtos disponíveis comercialmente em lojas de equipamentos médicos.

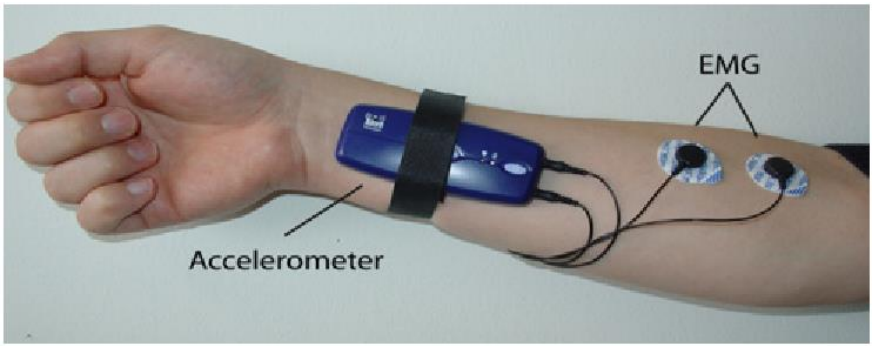


Figura 3-4 – Sensores empregados por Kim [Kim08] e suas respectivas posições

Para testes, 7 gestos da língua americana de sinais foram realizados, por 8 indivíduos, sendo seis do sexo masculino e 4 do feminino, com idades variando entre 27 até 41 anos. Cada um realizou o gesto 10 vezes em sequência, com cada gesto de duração aproximada de 1 até 3 segundos. Os dois sensores produziram um vetor de 56 características, incluindo a posição da mão, e características obtidas pelo domínio da frequência, como frequência fundamental F0 e variância de Fourier. Para classificação, foi utilizado o algoritmo do KNN, com K igual a cinco. As taxas de acerto foram de 99,82% para as abordagens baseadas em uma pessoa e de 96,31% considerando múltiplos indivíduos.

3.3. Considerações finais

A literatura demonstra que é promissor o uso de múltiplos sensores na análise de sinais. As abordagens utilizando sensores especiais invasivos, tais como luvas e acelerômetros, embora apresentem altas taxas de reconhecimento, geram ao usuário o inconveniente de vestir o sensor e, por ser incomum, de ter que carregá-lo. Ainda que o sensor fosse disponibilizado em locais públicos, haveria questionamentos quanto sua higiene. Além disso, conforme observado por Antunes [Ant01], para a completa identificação de um sinal Libras, não bastam apenas à posição das mãos, mas também a identificação da expressão facial do rosto e a movimentação do corpo. Uma solução completa seria então obrigada a recorrer a mais sensores, ou a uma abordagem que combinasse também a visão computacional.

Abordagens totalmente baseadas em visão computacional, por outro lado, não possuem esses inconvenientes. Como a literatura já demonstra [AnH10] [Bre07], o uso de um sensor de profundidade, como as câmeras time-of-flight, reduzem consideravelmente o esforço de segmentação da mão, eliminando boa parte das dificuldades envolvendo questões como iluminação ou tonalidade da pele.

Com o lançamento do Kinect, que combina um sensor de profundidade com a câmera RGB e os esforços da Microsoft de popularizá-lo, inclusive entre os usuários de PCs, pessoas comuns passaram a adquirir o equipamento, tornando a produção em escala viável e o dispositivo comercialmente acessível. Com isso, reduz-se o inconveniente do surdo ter a necessidade de carregar o dispositivo consigo e, mesmo que utilize um dispositivo público, a abordagem de visão tem a vantagem de não depender de qualquer contato físico.

Capítulo 4

Método Proposto

Neste capítulo, apresenta-se o método proposto. Este é dividido em duas etapas, a saber:

- **Etapa de preparação**: Nesta etapa, realiza-se a criação dos codebooks, criação do ensemble com diversos modelos escondidos de Markov, e o treinamento dos modelos utilizando a base de vídeos previamente gravada;
- **Etapa de execução**: Nesta etapa, utiliza-se a solução para o reconhecimento dos gestos. Nessa etapa será descrito o processo de extração das características da mão utilizando-se o algoritmo SIFT, a extração das características do esqueleto, a quantização dos vetores de características, a forma como são combinados e a geração de lista de palavras através dos HMMs previamente treinados.

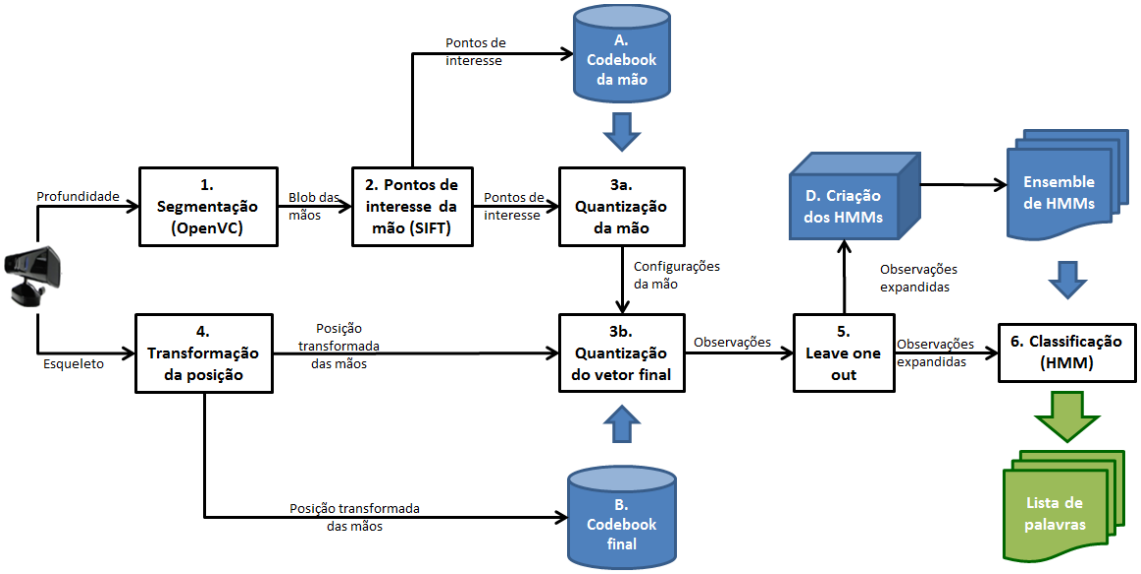


Figura 4-1: Visão geral do método proposto

4.1. Etapa de preparação

Para a detecção de cada palavra é necessário criar um conjunto de 9 HMMs, sendo cada um responsável pela identificação de uma palavra específica. Para a criação de um único conjunto desses, escolhe-se um tamanho de codebook e uma topologia em cada HMM. A variação desses parâmetros geram conjuntos melhores ou piores em diferentes aspectos.

A etapa de preparação envolve a criação de um *ensemble* desses conjuntos. Embora a palavra *ensemble* também signifique conjunto, o termo foi mantido em sua versão original em inglês para se referir ao conjunto formado pelo conjunto de HMMs.

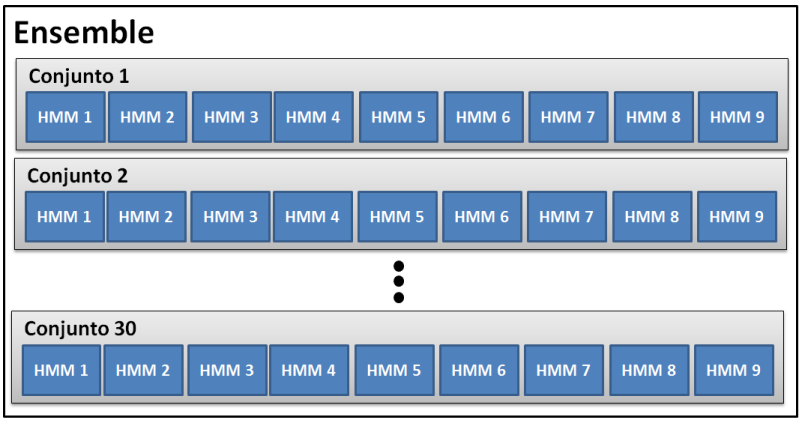


Figura 4-2: Ensemble de classificadores

Três desafios se apresentam nessa etapa. Na Figura 4-1, os principais módulos dessa etapa aparecem em azul.

O primeiro é gerar os *codebooks* para os pontos de interesse das mãos (A) e para o vetor final de características (B), utilizando o algoritmo *k-means*.

O segundo é gerar o ensemble dos modelos (C), que pode variar em sua topologia, número de estados e tamanho do *codebook*.

O último desafio está no treinamento em si (D). Devido ao pequeno número de amostras da base, é necessário utilizar técnicas como a validação cruzada e o treino *leave-one-out*, visando maximizar o número de amostras, sem comprometer o resultado dos testes.

4.1.1. Treinamento e validação dos HMMs

O processo de treinamento inicia-se gerando-se aleatoriamente o *ensemble* de HMMs. São gerados 30 conjuntos de HMMs diferentes, de maneira aleatória, sendo que:

a) É necessário sortear o tamanho correto de *codebook*, a ser usado em todas os HMMs para os vetores de movimento do braço. Os tamanhos de codebook variaram de 20 até 35 (tamanhos obtidos experimentalmente);
b) É necessário decidir, para cada um dos 9 modelos, se a topologia utilizada seria esquerda-direita ou ergódiga e qual seria o número de estados ideal, que podiam variar de 5 até 56;

Em seguida, o treinamento continua gerando as sequencias secundárias de testes necessárias para o processo de *leave-one-out*.

Para o processo de validação é utilizada a técnica de validação cruzada conhecida como k-fold [Koh95]. Neste processo, retira-se *k* elementos de cada palavra para o grupo de testes, treinam-se os HMMs sem este elemento, e realizam-se os testes utilizando este grupo separado. É importante ressaltar que o processo de leave-one-out deve ocorrer dinamicamente após a separação do grupo de testes, ou seja, deve garantir que as sequencias secundárias sejam geradas somente no momento em que os treinamentos sejam realizados. Assim, é dada a garantia de que as sequencias

secundárias de um determinado elemento não estariam presentes no grupo de treinamento, caso esse elemento tivesse sido separado para os testes.

O processo se repete até que todos os elementos tenham sido testados. Desta forma, garante-se a maximização do grupo de testes, sem contaminação por parte do grupo de treinamento.

A determinação do melhor *k*, que é de tamanho 17, equivalente ao k-folding deixe um de fora, foi feita através de experimentação e os resultados são sumarizados na Tabela 5-3: Impacto do tamanho base de treinamento e leave-one-out, na página 73.

4.2. Etapa de execução

Durante a execução, o processo de captura será realizado pelo Kinect e pela biblioteca OpenNI.

Inicialmente, o ator posiciona-se em frente ao sensor, e faz um gesto, para que a identificação de seu esqueleto seja realizada. Esta identificação é feita aumaticamente pela biblioteca OpenNI.

O software de captura é então responsável por armazenar as posições x, y e z das juntas de ambas as mãos e da cabeça. Essa gravação é feita num formato de dados binário.

Tal informação é encaminhada ao processo de segmentação (1). Durante esse processo, utiliza-se a informação do esqueleto do Kinect para definir um quadro em torno da mão (blob) e recorta-lo da imagem. A informação de profundidade é utilizada, de modo a descartar a informação do fundo.

Em seguida (2), utiliza-se o algoritmo SIFT para obter um conjunto de pontos significativos do contorno da mão. A forma geral do contorno pode ser obtida, pois o algoritmo é resistente à orientação ou escala da mão.

Cada quadro é então encaminhado à etapa de *clustering* (3a), sendo o vetor das características SIFT é agrupado através de um codebook previamente elaborado com o k-means.

O resultado serve de entrada para um segundo vetor, sendo assim associado as posições relativas (4) x, y e z das mãos direita e esquerda. Esse vetor é classificado com base num segundo codebook (3b), também elaborado utilizando o k-means.

Todos os quadros do vídeo formam o conjunto de observações. Esse conjunto é expandido pois, utilizando a técnica de *leave-one-out* (5), geram-se as sequências secundárias que também serão utilizadas nos testes.

Finalmente, todos os quadros compõem um vetor de observações, que é classificado utilizando o ensemble de HMMs previamente treinadas (6). A combinação do ensemble é feita por votação simples: Cada palavra recebe um voto se aparecer como a mais relevante de um dado conjunto. Ao final do processo, ordena-se as palavras pela quantidade de votos.

4.2.1. Segmentação da mão

Uma vez que a informação de posição aproximada da mão está disponível no esqueleto gerado pela OpenNI, a segmentação das mãos ocorre através de dois passos:

1. Definição de uma região de interesse (ROI), com base na posição do esqueleto. O ROI foi arbitrariamente definido num quadrado de 75 por 75 pixels.

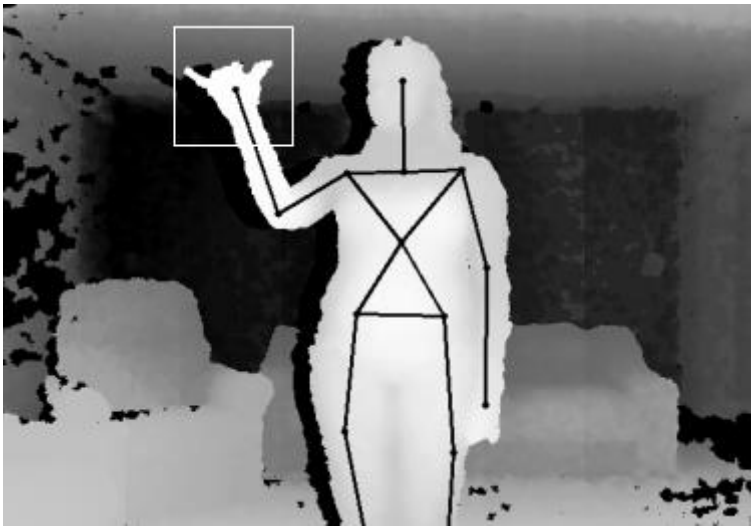


Figura 4-3: ROI da mão direita

2. Extração com base na informação de distância: Calcula-se o histograma das distâncias e isola-se o primeiro vale, que representa a mão, por estar a frente da cena. O vale é encontrando isolando-se o primeiro máximo local, e considerando um intervalo 12 cm para o fundo:

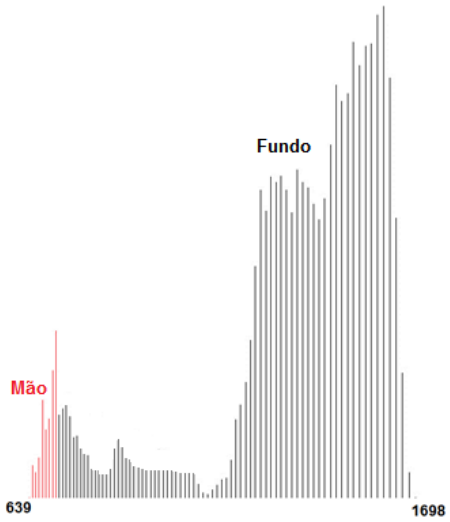


Figura 4-4: Histograma do ROI da mão

Então, separam-se todos os pixels abaixo do intervalo encontrado, o que gera a seguinte imagem:

Figura 4-5: Mão segmentada

Note que este ***não é*** um processo de limiarização binária, pois, associado a cada pixel da parte branca, que contém o desenho da mão, está o valor original de profundidade correspondente ao pixel.

4.2.2. Extração de características da mão

A extração de características é dada utilizando-se o algoritmo SIFT [Low04]. Para isso, utiliza-se a função pronta de mesmo nome da OpenCV e, com ela, solicita-se as 10 melhores características e mantem-se os demais parâmetros em seus valores padrão (nOctaveLayers em 3, limiar de contraste em 0.04, limitar de bordas em 10 e sigma em 1.6).

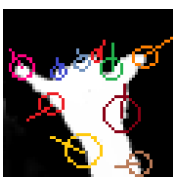


Figura 4-6: Pontos de interesse do SIFT

De cada ponto de interesse, são obtidas suas coordenadas x e y do ponto e seu ângulo, compondo assim um vetor de 30 características.

Esse vetor de características é quantizado através do algoritmo k-means, antes de ser inserido ao vetor de características de movimento. Para a criação do codebook, foram utilizados todos quadros, de todos os vídeos de treinamento.

4.2.3. Extração do movimento

O movimento das mãos direita e esquerda é detectado automaticamente pelo Kinect. Porém alguns tratamentos se fazem necessários para tornar a identificação do sinal mais resistente a características individuais.

a) O sistema de coordenadas do Kinect baseia-se na posição da câmera. Assim, os eixos x e y apresentam-se sempre no meio da imagem (centro da câmera) e a posição z representa a distância da câmera até o pixel. Essas características são indesejáveis, pois os indivíduos quase nunca estão perfeitamente centralizados na cena;
b) Indivíduos tem estatura física diferente. A detecção de pontos do esqueleto do Kinect praticamente elimina o ruído causado pelo peso do indivíduo, porém, a altura continua a variar consideravelmente, especialmente considerando o número pequeno de amostras da base.

Para resolver esses problemas, foram feitas duas transformações sobre os vértices.

Primeiramente, foi aplicada uma translação sobre os pontos, levando como referência a posição do quadril. Assim, sejam $\vec{m}$ e $\vec{q}$ os vetores de posição da mão em coordenadas da câmera e do quadril, respectivamente, então a coordenada da mão transformada $\overrightarrow{m_t}$ pode ser calculada por:

$$\overrightarrow{m_t} = \vec{m} - \vec{q} \qquad (4\text{-}1)$$

A segunda transformação realizada é a de escala. A motivação dessa transformação parte da constatação que seres humanos saudáveis, embora tenham estaturas diferentes, tem proporções corpóreas similares [Bog10]. Assim, ao tomarmos a distância da altura até o quadril do indivíduo como referência, e normalizar os vetores

com base nesse fator, podemos homogeneizar a estatura dos esqueletos, minimizando o ruído causado pelas diferentes estaturas.

Assim, seja $\vec{c}$ o vetor de posição da cabeça em coordenadas da câmera, calculamos a altura h da cabeça ao quadril através da seguinte fórmula:

$$h = \|\vec{q} - \vec{c}\| \tag{4-2}$$

A transformação de escala é feita dividindo-se os vetores transformados das mãos pelo resultado:

$$\overrightarrow{m_f} = \frac{\overrightarrow{m_t}}{h} \tag{4-3}$$

Dessa forma, cria-se um sistema coordenado centrado no quadril, de forma que o ponto 1 refira-se a cabeça, independente da altura do indivíduo e sua posição em relação a câmera.

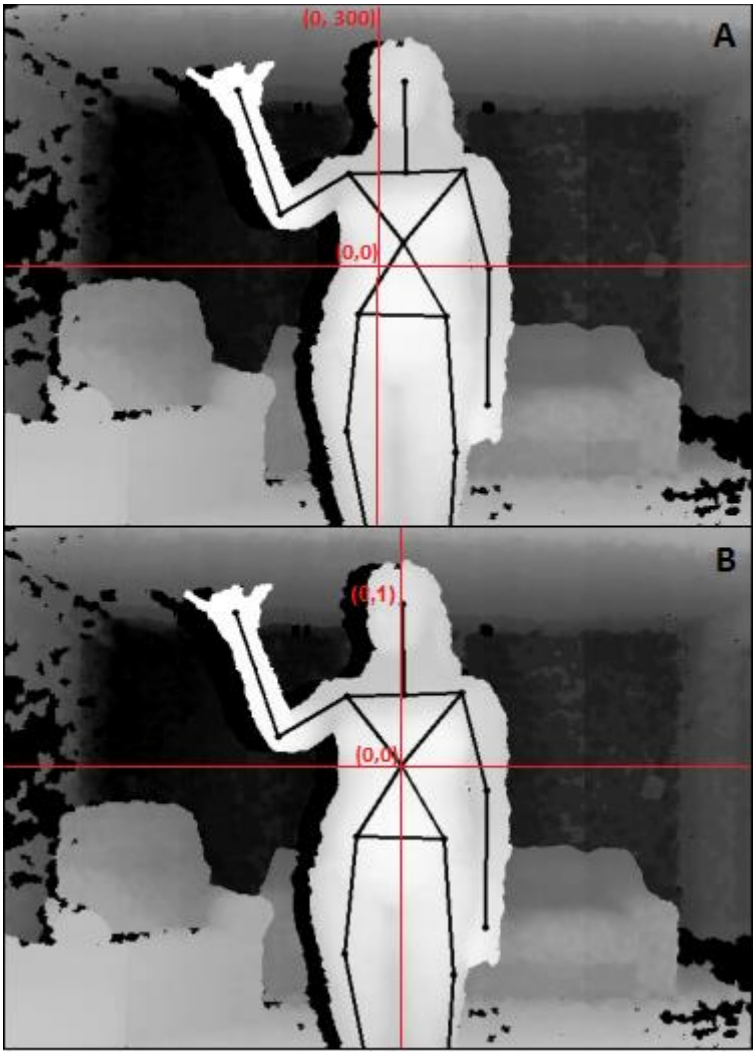


Figura 4-7: a) Coordenadas originais b) Coordenadas transformadas

4.2.4. Reconhecimento do sinal

Como resultado das fases anteriores, a cada quadro, as seguintes características foram obtidas, para cada mão:

a) Um índice de codebook representando os pontos significativos da forma da mão, obtidos pelo SIFT;
b) As posições x,y e z das mãos, transformadas.

Tais características são novamente agrupadas num codebook de 33 entradas, compondo assim o índice final da observação.

A sequencia de observações de um vídeo é produzida através da junção de todos os quadros que o compõe. São produzidas sequencias diferentes para a mão direta e esquerda.

Para o reconhecimento do sinal, foram criados dois grandes grupos de HMMs, um para mão direta e outro para a esquerda.

Cada grupo contém um modelo para cada palavra, treinado através da técnica e leave-out-out training. Assim sinal é submetido a conjunto de HMMs, através dos seguintes passos:

- Expandem-se as observações através do processo de leave-one-out;
- Cada observação expandida é submetida ao algoritmo de Viterbi, de modo que a probabilidade ter sido gerada com o modelo seja calculada;
- A maior taxa de probabilidade é escolhida como correta, uma vez que representa o modelo de menor ruído [KoA09].

Ao final do processo, obtém-se a probabilidade do sinal ter sido gerado em cada palavra, em cada mão. As taxas das duas mãos são combinadas multiplicando-se uma pela outra.

Finalmente, ordenando as palavras pelas probabilidades de terem sido geradas em cada modelo, obtém-se uma lista classificada de palavras para aquele conjunto. A malhor palavra dessa lista é votada. Ao final do ensemble, obtém-se uma lista ordenada das palavras mais votadas. Aquela com maior número de votos é considerada a correta.

4.2.5. Considerações finais

Neste capítulo foi proposto um método, o qual foi descrito em detalhes. O método é dividido em duas etapas, sendo uma de preparação – onde são gerados os codebooks e os modelos – e outra de execução.

Demonstrou-se como fazer a segmentação da informação das mãos direita e esquerda. Para cada mão, demonstrou-se como realizar a extração das características segmentando-se a imagem de profundidade e utilizando o algoritmo SIFT sobre o resultado.

Para resolver o problema de como combinar essa informação com o esqueleto, demonstrou-se que os pontos de interesse da mão, ao serem quantizados, geram uma informação que pode ser encarada como o formato geral do *blob* segmentado. Essa

informação pode então ser diretamente combinada ao vetor contendo as posições x, y e z da mão do indivíduo, gerando um vetor final de características.

Após o processo de quantização, toda essa informação compõe os vetores de observações utilizados nos HMMs. O método propõe o uso da técnica de *leave-one-out* durante os testes e treinamento, de modo a tornar os HMMs mais tolerantes a ruídos e mais confiáveis, reduzindo o impacto da base de dados ser pequena.

Finalmente, durante a etapa de execução, mostrou-se como combinar os resultados dos modelos da mão esquerda e direita, e compor a lista de palavras, em ordem de votação, obtendo-se assim o resultado final esperado.

4.3. Variação do método

Inclui-se uma variação ao método proposto para que fosse possível compará-lo. No lugar da geração do *ensemble* de classificadores, buscou-se obter o melhor conjunto possível de classificadores.

Este conjunto teria os valores de codebook, estados e topologias que maximizassem os resultados. Encontrar esse conjunto, entretanto, é uma tarefa computacionalmente desafiadora, dado o imenso número de combinações.

Assim, ao invés de sortear as características de cada HMM, como foi feito no *ensemble*, os tamanhos de codebooks, estados e topologias deveriam ser localizados.

Decidiu-se encarar esse problema como uma forma de busca não linear. Assim, tornou-se viável utilizar algoritmos genéticos e pesquisar pela configuração ótima de todos esses parâmetros juntos.

4.3.1. Características do algoritmo genético

O algoritmo utilizou um genoma binário, composto em sua versão final da seguinte forma:

- 4 bits para o tamanho de codebook, o que permitia obter tamanhos entre 20 e 35. Alguns experimentos chegaram a considerar codebooks maiores e menores, mas observou-se que o resultado apresentava perdas significativas fora desse intervalo de valores;
- E, para cada HMM:
    - 1 bit para determinar se a topologia seria esquerda-direita ou ergódiga;
    - 5 bits para determinar o número de estados (5 até 36)

O algoritmo utilizou seleção assintótica, conforme proposto por [Men08], taxa de cruzamento de 75%, mutação de 1% e elitismo de um único indivíduo. A configuração final obtida e os testes realizados podem ser vistos no item 5.3, pág. 72.

A função de avaliação baseou-se na taxa de acerto principal (*top 1*), calculada com base no método de treinamento e validação propostos.

Durante a preparação, é importante executar o processo até que haja convergência.

4.3.2. Limitações dessa técnica

A técnica, apesar de servir como base para comparação, apresenta várias limitações:

- O tamanho do genoma cresce linearmente com o número de HMMs, aumentando o tempo de convergência para uma lista muito maior de palavras;
- Pode gerar um genoma com *overffiting*, a menos que se expanda ainda mais a base de dados e possa-se separar parte das amostras e utiliza-las para critério de parada.

Apesar das limitações, pode-se comparar o desempenho do ensemble versus o desempenho de um único conjunto de classificação. Além disso, o modelo gerado pelo algoritmo genético também é útil para a validação de outras técnicas, tais como o uso de leave-one-out ou testes sobre o tamanho da base de validação.

Capítulo 5

Resultados Experimentais

Neste capítulo, será apresentada a base de vídeos, o processo utilizado na sua criação e os resultados experimentais que permitiram a construção e validação do método proposto.

5.1. Descrição da Base de Vídeos

A primeira contribuição deste projeto é a gravação de uma base de dados com um grupo de palavras em Libras. Nas sessões seguintes, descreve-se o processo de aquisição dessa base e suas características.

5.1.1. Características dos vídeos

Durante o processo, foram utilizados 23 atores de diferentes idades variando entre 17 e 60 anos, de variadas estaturas, sexos e cores de pele. Cada ator gravou de um até três vídeos. Exemplos desses atores podem ser vistos na Figura 5-1:

Figura 5-1: Diferentes atores e cenários

Foram gerados 181 vídeos, divididos conforme descreve a Tabela 5-1:

Tabela 5-1: Quantidade de vídeos e atores

Palavra	Número de vídeos	Atores diferentes
Entregar	22	12
Pegar	20	11
Abrir	20	11
Olhar	18	12
Empurrar	20	14
Fechar	19	10
Falar	23	14
Puxar	19	8
Trabalhar	20	16

Como a gesticulação da Libras é feita com a mão de maior agilidade, há atores destros e canhotos. É importante ressaltar que somente o braço principal é trocado, porém, a direção para qual o movimento é feito não muda.

Todos os vídeos foram gravados em resolução de 640x480 pixels e taxa de amostragem de 30Hz.

As imagens RGB foram gravadas sem compressão e com resolução de cores de oito bits por canal. A informação de profundidade possui resolução de 16 bits e foi gravada utilizando um algoritmo de compressão sem perdas (16 ET). O esqueleto é gravado utilizando um formato de arquivo binário.

A base também contém vídeos que contemplam algumas características indesejáveis da câmera RGB, mas que são comuns no manuseio do aparelho: perda de iluminação devido à calibração automática de luminosidade, faixas verdes em alguns quadros devido a falhas de sincronismo da câmera e brilho intenso de luz de fundo. A Figura 5-2 ilustra esses problemas:

Figura 5-2: Condições adversas do aparelho e do ambiente

Finalmente, os vídeos são gravados com os atores inicialmente na posição de repouso. O gesto é realizado e os atores retornam a posição de repouso. Essa posição é ilustrada nas duas últimas imagens da Figura 5-2.

5.1.2. Método de aquisição

O ambiente de captura foi montado utilizando-se:

- Um computador voltado para o usuário que realiza a captura.
- O Kinect, apontado em direção ao ator. O ator era posicionado de modo que, em posição de repouso, sua cabeça e suas mãos aparecessem na imagem.
- Acesso a acervo do Dicionário da Língua Brasileira de Sinais, que contém a referência para as palavras gesticuladas.

O processo de captura se dava da seguinte forma:

- Os atores praticam o gesto antes da captura, até que possam realiza-los com exatidão e sem confusões;
- No software de captura, escolhe-se o ator, palavra e tomada sendo gravadas;
- O software indica ao ator a posição inicial onde deve ser enquadrada sua cabeça e suas mãos. O usuário que fará a captura ajusta o enquadramento;
- O ator aguarda que seu esqueleto seja detectado pelo software de captura.
- O usuário que realiza a captura executa um comando no computador para que o sinal seja gravado.
- O ator gesticula o sinal. O sinal inicia e termina com o ator em posição de repouso.

Utilizou-se a seguinte estratégia para a identificação dos usuários, sinais e arquivos.

- A cada ator atribuiu-se um identificador único, numérico e sequencial. Esse identificador é previamente cadastrado no software de captura, junto ao nome do ator, para evitar erros humanos;
- Atribuiu-se a cada palavra um número identificador único, também associado ao software de captura.
    - 0 – Entregar
    - 1 – Pegar
    - 2 – Usar: Observação descobriu-se que a palavra usar não existe fora de contexto em Libras. Ela foi substituída pela palavra Trabalhar.
    - 3 – Abrir
    - 4 – Olhar
    - 5 – Empurrar
    - 6 – Fechar
    - 7 – Falar
    - 8 – Puxar
    - 9 - Trabalhar
- Cada tomada é associada a um contador, iniciado em um. Com se trata de um contador e não de um identificador, tomadas descartadas não foram contadas.
- Nomeou-se o arquivo da seguinte forma:

- Letra V, indicando uma captura de vídeo;
- Identificador do ator
- Número da tomada
- Identificador da palavra
- Separação usando underscores
- Formato:
  - RGBD: V_ATOR_TOMADA_PALAVRA.oni
  - Esqueleto: V_ATOR_TOMADA_PALAVRA.skl

- Por exemplo, o arquivo V_03_2_5.oni indica que o terceiro ator, na segunda tomada, gravou a palavra empurrar.

5.2. Softwares para captura e manipulação

No decorrer do trabalho, foram elaborados dois softwares específicos para a captura e manipulação das imagens. O primeiro trata-se de uma biblioteca que integra a OpenNI e a OpenCV, além de implementar uma série de funções úteis para desenho e armazenamento de informações.

O segundo é uma aplicação completa de captura, que permitiu a gravação dos vídeos seguindo o protocolo descrito nesta dissertação.

5.2.1. Biblioteca de Integração das bibliotecas OpenNI e OpenCV 2

Devido à necessidade de integrar duas bibliotecas distintas e compreender os dados de entrada, criou-se uma biblioteca de integração entre a OpenNI e a OpenCV, intitulada xncv.

Na biblioteca, foram acrescentadas funções como:

- Captura da imagem RGB e da matriz de profunidade do Kinect, e conversão para o formato cv::Mat, da OpenCV;
- Conversão do formato RGB, padrão da OpenNI, para o formato BGR, padrão da OpenCV;
- Funções para cálculo do histograma de profundidade;
- Função para desenho da matriz de profundidade em escala de cinza em dois métodos distintos: quantização simples e quantização por histograma;

- Classes para facilitar a leitura de vídeo, seja a partir do Kinect ou a partir de arquivos;
- Classes para a gravação, leitura e desenho do esqueleto;

A biblioteca foi disponibilizada publicamente no endereço:
https://github.com/ViniGodoy/xncv

5.2.2. Gravação dos vídeos

Para a aquisição dos dados, foi programado um software de captura utilizando-se a ferramenta QT Creator, a biblioteca OpenGL e a biblioteca xncv. O software é capaz de gravar as informações RGB e de profundidade do ator no formato .oni, proprietário da biblioteca OpenNI.

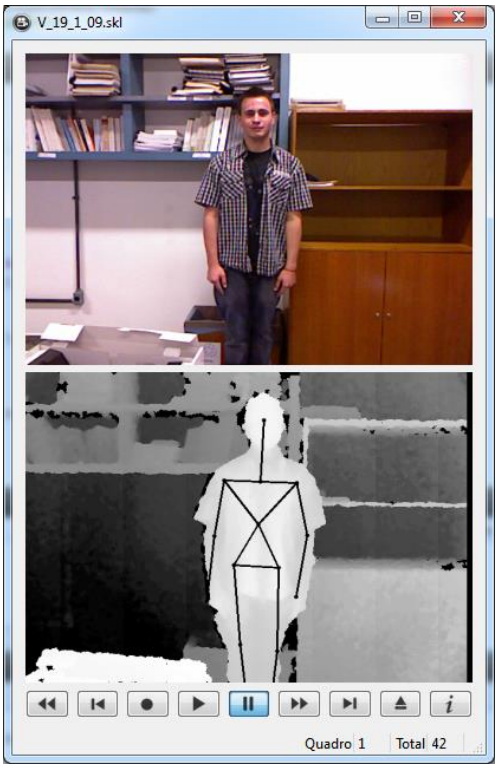


Figura 5-3: Aplicação de captura de dados

O formato .oni não armazena informações de esqueleto. Por isso, foi definido um formato binário de armazenamento intitulado .skl.

Além disso, o software gera automaticamente o nome do arquivo com base no protocolo definido. Durante o processo de gravação, o usuário segue um *wizard*, que orienta o usuário nas etapas de gravação – informação dos dados, enquadramento do ator, detecção do esqueleto e captura – reforçando o protocolo definido.

O software também permite a reprodução posterior dos vídeos, desenhando inclusive desenhando a informação de esqueleto, caso o arquivo de esqueleto esteja disponível.

5.3. Avaliação do Ensemble de Modelos (HMM)

Abaixo o comparativo da taxa de acertos onde a palavra correta ocorre classificada como primeira opção (top 1), primeira ou segunda opções (top 2) ou primeira, segunda ou terceira opções (top 3).

Tabela 5-2: Comparativo entre a abordagem proposta e os algoritmos genéticos

Vídeos na base de treinamento (Por palavra)	Abordagem proposta			Algoritmos genéticos		
	Top 1	Top 2	Top 3	Top 1	Top 2	Top 3
1	29,0%	29,6%	33,3%	27,8%	25,4%	26,5%
3	46,9%	49,4%	50,6%	25,97%	28,7%	30,3%
5	59,9%	64,8%	66,0%	37,5%	43,0%	57,5%
10	85,0%	87,7%	90,7%	63,5%	77,9%	79,0%
14	88,5%	98,8%	98,8%	69,1%	84,5%	86,2%
18	**90,2%**	**100%**	**100%**	76,2%	87,3%	92,82%

Observa-se que a abordagem proposta, utilizando o ensemble de classificadores, possui taxa de acerto muito superior aos algoritmos genéticos. Além disso, a abordagem é também mais resistente a mudanças no tamanho da base de treinamento. Nota-se, por exemplo, que os resultados com 10 vídeos na base de treinamento já são superiores aos melhores resultados obtidos pelo classificador individual classificado com os algoritmos genéticos.

O melhor resultado se repetiu ao utilizar-se 17 vídeos na base de treinamento e, com 16 vídeos, obteve-se apenas a confusão de um único vídeo no top 2, fazendo sua taxa cair para 99,38%.

Finalmente, a abordagem proposta não requer o longo tempo de calibração dos algoritmos genéticos. Em contrapartida, a abordagem multi-classificador proposta consome aproximadamente 30 vezes mais espaço em memória, e leva cerca de 10 vezes mais tempo para realizar a classificação.

5.4. Estratégia de Geração de Novas Sequências de Observações

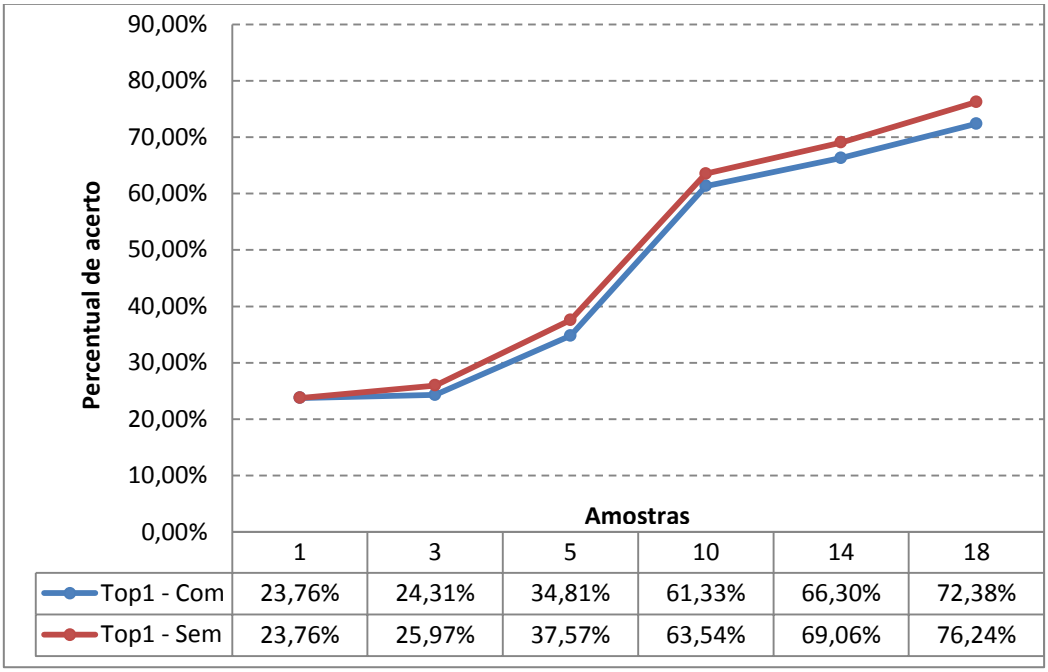
Esta sessão avalia o impacto da estratégia de teste e treino leave-one-out sobre os resultados. Para a avaliação, foi utilizado o classificador com o algoritmo genético.

Os resultados em negrito, na Tabela 5-3, referem-se ao mesmo processo utilizado como função de desempenho para o algoritmo genético, descrito nos capítulos 4.3 e **Erro! Fonte de referência não encontrada.**.

Tabela 5-3: Impacto do tamanho base de treinamento e leave-one-out

Vídeos na base de treinamento (Por palavra)	Sem leave-one-out			Com leave-one-out		
	Top 1	Top 2	Top 3	Top 1	Top 2	Top 3
1	23,76%	24,86%	25,97%	23,76%	25,41%	26,52%
3	24,31%	25,97%	28,18%	25,97%	28,73%	30,39%
5	34,81%	41,44%	53,59%	37,57%	43,09%	57,46%
10	61,33%	76,80%	78,45%	63,54%	77,90%	79,01%
14	66,30%	81,22%	83,98%	69,06%	84,53%	86,19%
18	72,38%	83,98%	88,40%	**76,24%**	87,29%	92,82%

A figura abaixo compara a evolução do desempenho do top 1, com e sem o leave one out, de acordo com o tamanho da base de treinamento. Considera-se o número mínimo de *k* (1 amostra para treinamento) até o número máximo (17 amostras). O número máximo é limitado pela quantidade de vídeos presentes na *menor* classe de palavras (18 vídeos).



	1	3	5	10	14	18
Top1 - Com	23,76%	24,31%	34,81%	61,33%	66,30%	72,38%
Top1 - Sem	23,76%	25,97%	37,57%	63,54%	69,06%	76,24%

Figura 5-4: Impacto da base de treinamento e leave-one-out (top 1)

De acordo com a Tabela 5-3 e com a Figura 5-4, observa-se que:

a) A presença do Leave-one-out interfere positivamente no resultado causando uma melhora na taxa de reconhecimento de aproximadamente 3%. Essa taxa tornou-se mais significativa com um conjunto maior de dados;
b) O tamanho da base de treinamento tem impacto significativo sobre o resultado. Esse impacto tornou-se mais pronunciado a partir do momento em que o número de amostras de treinamento superou aproximadamente metade do número total de vídeos disponíveis;
c) O impacto é crescente até o tamanho máximo da base.

Com estes resultados, é possível concluir que para uma base de tamanho reduzido, é vantajoso utilizar *k* com tamanho igual ao número de elementos da base subtraído de um.

5.5. Análise de Erros

Abaixo é apresentada a matriz de confusão para a solução proposta, considerando-se correta a palavra apenas na primeira colocação:

Tabela 5-4: Matriz de confusão (Ensemble)

	entregar	pegar	abrir	olhar	empurrar	fechar	falar	puxar	trabalhar
entregar	12,5%	0,0%	0,6%	0,0%	1,1%	0,6%	0,0%	0,0%	1,1%
pegar	0,0%	6,8%	0,0%	0,0%	0,0%	0,0%	0,0%	**3,3%**	0,0%
abrir	0,0%	0,0%	11,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
olhar	0,0%	0,0%	0,0%	9,9%	0,0%	0,0%	0,0%	0,0%	0,0%
empurrar	0,0%	0,0%	0,0%	0,0%	11,0%	0,0%	0,0%	0,0%	0,0%
fechar	0,0%	0,0%	0,0%	0,0%	0,0%	10,5%	0,0%	0,0%	0,0%
falar	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	12,7%	0,0%	0,0%
puxar	0,0%	**4,7%**	0,0%	0,0%	0,0%	0,0%	0,0%	5,2%	0,0%
trabalhar	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%	11,0%

Observa-se facilmente que os erros, representando 8,0% dos vídeos da base, concentram-se nas palavras “pegar” e “puxar”. As duas palavras possuem movimentos similares, diferindo apenas pela sua velocidade e posição lateral do braço.

Devido a isso, a palavra “pegar” foi classificada corretamente em apenas 65% das vezes, enquanto a palavra “puxar” em 52,6%. Entretanto, a classificação correta já ocorre em segundo lugar.

A Figura 5-5, demonstra essa diferença. A linha superior demonstra o movimento “pegar”, com sua posição final de mão ocorrendo no quadro 11. Observe que ela é muito similar à posição final do verbo “puxar”, que ocorre já no sexto quadro. A diferença está apenas na lateralidade do braço e na velocidade do movimento.

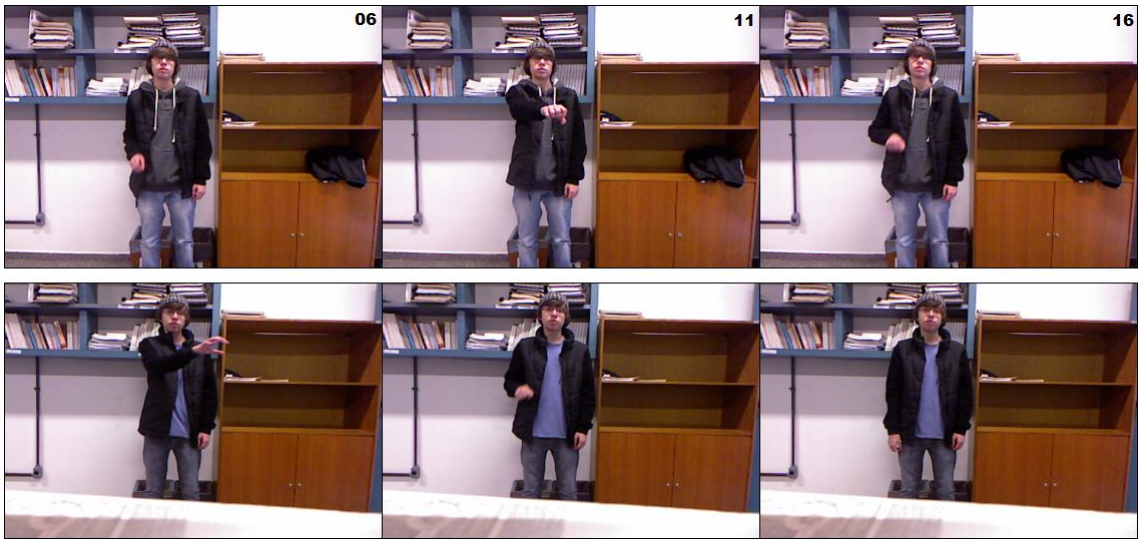


Figura 5-5: Comparação entre pegar e puxar

Finalmente, a tabela abaixo mostra os percentuais de acerto de cada uma das palavras, considerando que foram classificadas corretamente até a primeira ou segunda opções (top1, top2).

Tabela 5-5: Percentual de acerto das palavras

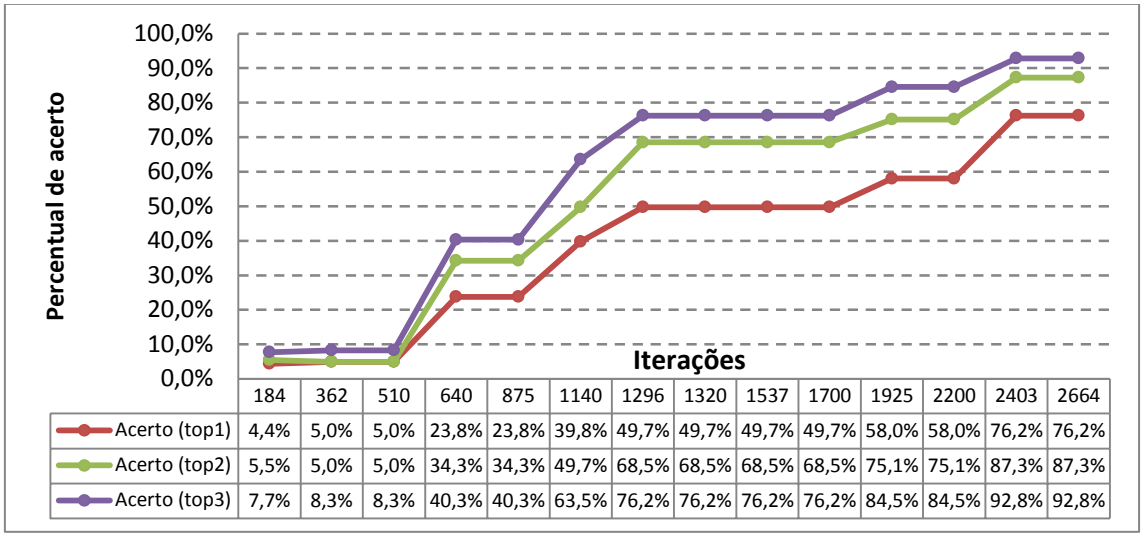
Palavra	Top 1	Top 2
entregar	100%	100%
pegar	65,0%	100%
abrir	100%	100%
olhar	100%	100%
empurrar	100%	100%
fechar	100%	100%
falar	100%	100%
puxar	52,6%	100%
trabalhar	100%	100%
Total	**90,2%**	**100%**

Observa-se que os erros de classificação restringiram-se a duas palavras, portanto, o que indica que a falta de alguma característica que melhore a diferenciação dessas duas palavras.

5.6. Otimização dos Modelos Escondidos de Markov (HMM)

Por fim, demonstra-se a evolução do conjunto de modelos escondidos de Markov submetido aos algoritmos genéticos.

A versão final do algoritmo convergiu em 2664 iterações, e o impacto da evolução dos modelos pode ser observado no gráfico da Figura 5-6, que demonstra a evolução do percentual de acerto, pelo número de iterações:



	184	362	510	640	875	1140	1296	1320	1537	1700	1925	2200	2403	2664
Acerto (top1)	4,4%	5,0%	5,0%	23,8%	23,8%	39,8%	49,7%	49,7%	49,7%	49,7%	58,0%	58,0%	76,2%	76,2%
Acerto (top2)	5,5%	5,0%	5,0%	34,3%	34,3%	49,7%	68,5%	68,5%	68,5%	68,5%	75,1%	75,1%	87,3%	87,3%
Acerto (top3)	7,7%	8,3%	8,3%	40,3%	40,3%	63,5%	76,2%	76,2%	76,2%	76,2%	84,5%	84,5%	92,8%	92,8%

Figura 5-6: Convergência do algoritmo genético

No gráfico são exibidos os acertos levando em consideração que o elemento correto esteja classificado somente na primeira opção, na primeira ou segunda posições (top 2) ou entre as três primeiras posições (top 3).

O resultado final foi obtido com um codebook de tamanho 33 e convergência resultou nas seguintes HMMs:

Tabela 5-6: Configuração final das HMMs

Palavra	Topologia	Estados
Entregar	Esquerda-Direita	23
Pegar	Esquerda-Direita	21
Abrir	Esquerda-Direita	22
Olhar	Ergódiga	23
Empurrar	Ergódiga	9
Fechar	Esquerda-Direita	23
Falar	Esquerda-Direita	27
Puxar	Ergódiga	19
Trabalhar	Esquerda-Direita	25

Esse modelo apresentou a matriz de confusão descrita na Tabela 5-7:

Tabela 5-7: Matriz de confusão (Algoritmo Genético)

	entregar	pegar	abrir	olhar	empurrar	fechar	falar	puxar	trabalhar
entregar	8,8%	0,0%	0,6%	0,0%	1,1%	0,6%	0,0%	0,0%	1,1%
pegar	0,0%	6,1%	0,0%	0,0%	0,0%	0,0%	0,0%	**3,9%**	0,0%
abrir	0,0%	0,0%	8,28%	0,6%	0,0%	1,1%	1,1%	0,0%	0,0%
olhar	0,0%	0,0%	1,1%	8,3%	0,6%	0,0%	0,0%	0,0%	0,0%
empurrar	0,0%	0,0%	1,1%	0,0%	9,4%	0,0%	0,0%	0,0%	0,6%
fechar	0,0%	0,0%	1,6%	0,6%	0,6%	7,7%	0,0%	0,0%	0,0%
falar	0,0%	0,0%	1,1%	0,6%	0,6%	0,0%	10,5%	0,0%	0,0%
puxar	0,0%	**4,4%**	0,0%	0,0%	0,0%	0,6%	0,0%	6,6%	0,0%
trabalhar	0,0%	0,0%	0,6%	0,0%	0,0%	0,0%	1,1%	0,0%	9,4%

Que evidencia que um único conjunto de HMMs, por melhor que seja, ainda apresentará dificuldades de diferenciação de movimentos similares, tais como pegar e puxar e abrir e fechar.

5.7. Considerações finais

Os resultados demonstram que o método proposto, utilizando o ensemble de classificadores, é mais robusto e tem melhores taxas de acerto do que o uso de um único classificador, mesmo considerando um processo de calibração exaustiva de seus parâmetros de classificação.

Esse resultado se deve pelo fato de que, classificadores diferentes apresentam diferentes ergonomias, resultando em taxas de erro diferentes distribuídas ao longo das palavras candidatas.

Como resultado, ao combiná-los, as taxas de erros tendem-se a dispersar, enquanto a palavra corretamente classificada destaca-se em primeiro lugar.

Para demonstrar essa fato, observemos a tabela que indica as votações para a palavra “Trabalhar”, que foi classificada corretamente apenas em 85% dos casos nos algoritmos genéticos, contra 100% de acerto no método proposto:

Tabela 5-8: Votações da palavra trabalhar

Arquivo	entregar	pegar	abrir	olhar	empurrar	fechar	falar	puxar	trabalhar
V_01_1_09	6	2	1	1	3	0	3	0	14
V_03_1_09	3	0	2	0	2	3	4	0	16
V_04_1_09	1	1	5	3	1	1	0	1	17
V_05_1_09	5	3	0	2	2	2	2	0	14
V_06_1_09	1	2	2	1	3	1	0	1	19
V_06_2_09	3	2	4	1	1	2	2	2	13
V_07_1_09	3	2	1	1	1	2	1	1	18
V_08_1_09	1	3	1	4	2	2	3	2	12
V_09_1_09	5	3	1	2	2	0	2	6	9
V_09_2_09	3	3	1	2	0	1	4	2	14
V_11_1_09	2	3	2	4	1	1	3	0	14
V_12_1_09	3	3	0	2	0	1	1	2	18
V_13_1_09	3	5	6	0	3	1	1	2	9
V_16_1_09	2	0	4	1	2	1	2	1	17
V_17_1_09	0	1	2	0	3	5	1	3	15
V_18_1_09	1	0	6	2	2	3	0	4	12
V_19_1_09	1	6	2	3	1	1	2	1	13
V_20_1_09	1	2	2	3	4	3	5	0	10
V_20_2_09	1	5	2	1	2	2	3	2	12
V_20_3_09	2	0	1	3	4	1	0	1	18

Observa-se que, apesar de existirem 30 classificadores, os vídeos foram classificados corretamente por aproximadamente 14 HMMs em cada caso – o que foi suficiente para eleger a palavra correta. Já as classificações incorretas distribuíram-se entre as demais palavras da base.

Capítulo 6

Conclusão e Trabalhos Futuros

Este trabalho apresentou um método para identificação de um conjunto fixo de gestos de Libras. O método é capaz de fornecer uma lista de palavras candidatas, com base em sua probabilidade de ser correta.

Para que esse processo fosse possível, foi gravada uma base de vídeos com 183 exemplares, de 23 atores distintos, que representa a primeira contribuição dessa pesquisa.

A estratégia de teste e treino leave-one-out foi utilizada para evitar ruídos durante o treinamento e melhorar a eficiência da base, o que se confirmou ao melhorar em aproximadamente 3% a taxa de reconhecimento.

Além disso, respondendo a uma das perguntas do questionamento, avaliou-se o impacto gerado pelo tamanho da base durante o processo de validação cruzada e os resultados demonstraram que a melhora dos percentuais de acerto está diretamente relacionada a quantidade de vídeos de treinamento. Os resultados mostraram que, a partir de metade do tamanho da base (10 vídeos), as taxas de acerto passam a ser superiores a 60%, o que não ocorre anteriormente.

A base foi gravada com o hardware Kinect, da Microsoft, que possui um algoritmo de rastreamento do esqueleto. Esse algoritmo, no entanto, não é capaz de identificar configurações da mão e, para isso, esta pesquisa contribuiu ao propor um

classificador baseado no algoritmo SIFT. Este algoritmo produz um conjunto de pontos de interesse invariantes à escala, que foram utilizados para determinar de maneira indireta a forma da mão. Seu vetor de características foi quantizado, antes de ser aplicado ao vetor de características do esqueleto. Assim, respondeu-se ao questionamento sobre como essas informações poderiam ser utilizadas junto aos Modelos Escondidos de Markov.

Por fim, os resultados foram medidos calculando os percentuais de acerto quando a palavra era classificada corretamente na primeira opção (top 1), obtendo taxa de reconhecimento de 92,2%, quando a palavra aparecia corretamente classificada até a segunda opção (top 2), representando taxa de acerto de 100%.

A análise da matriz de confusão também destacou que os erros de classificação se concentraram em duas palavras, “pegar” e “puxar”. Erro que poderia ser eliminado ao utilizar-se informações de contexto.

Dentro desse contexto, vários trabalhos futuros são sugeridos:

Uma possibilidade de trabalho futuro está na confecção de uma base de dados ainda maior, considerando uma lista de palavras mais diversificada ou aumentando o número de amostras em cada palavra;

Sugere-se a futuras equipes que busquem melhorias no classificador de modo a evitar a principal confusão obtida no trabalho, referente às palavras “pegar” e “puxar”;

Outra possibilidade é buscar melhorias nos resultados através de técnicas de one shot learning [Fei06], que permitam melhorias na criação e separação dinâmica das palavras.

Outra extensão possível para esse trabalho está em adaptar o algoritmo para identificação de sentenças simples, formadas por duas ou mais palavras. Uma abordagem comum em jogos educacionais é treinar o classificador para identificar uma sequencia de sujeito + verbo + objeto, similarmente ao que foi realizado por [Zaf11].

Finalmente, na linha educacional, sugere-se a construção do jogo educacional e avaliação de sua capacidade de educar.

Referência bibliográfica

[AnH10] An, H. e Kim, D. , "Hand Gesture Recognition using 3D Depth Data," in *In Proceeding of 10th POSTECH-KYUTECH JOINT WORKSHOP*, 2010, pp. 357-359.

[Ant01] Antunes, Diego R., Guimarães, Cayley , Garcia, Laura S. , Oliveira, Luiz Eduardo S. e Fernades, Sueli , "A Framework to Support Development of Sign Language Human-Computer Interaction - Building Tools for Effective Information Access and Inclusion of the Deaf," in *IEEE RCIS 2011. Fifth IEEE International Conference on Research Challenges in Information Science*, Guadeloupe, 2001, pp. 1-12.

[Ant11] Antunes, Diogo R., "Um modelo para descrição computacional da fonologia da língua de sinais brasileira," UFPR, Curitiba, 2011.

[Bay04] Bay, H. , Ess, A. , Tuytelaars, T. e Gool, L. Van , "SURF: Speed Up Robust Features," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346-359, 2008, 2004.

[Bog10] Bogin, Barry e Silva, Maria Inês Varela , "Body proportions and health: A review with note on beauty," *International Journal of Environmental Research and Public Health*, no. 7, pp. 1047-1075, 2010.

[Bor98] Boreczky, John S. e Wilcox, L. D. , "A hidden Markov model framework for video segmentation using audio and image features," *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 6, pp. 3741 - 3744, Maio 1998.

[Bra02] Legislação Brasileira,. (2002, Abril) LEI Nº 10.436. Disponível em: http://www.planalto.gov.br/ccivil_03/Leis/2002/L10436.htm

[Bre07] Breuer, Pia , Eckes, Christian e Müller , Stefan , "Hand Gesture Recognition with a novel IR Time-of-Flight Range Camera – A pilot study," in *Computer vision/computer graphics collaboration techniques : third international conference, MIRAGE 2007*, Recquencourt, France, 2007, pp. 247-260.

[Car09] Carneiro, Alex T. S., Cortez, Paulo C. e Costa, Rodrigo C.S. , "Reconhecimento de Gestos da LIBRAS com Classificadores Neurais a partir dos Momentos Invariantes de Hu," in *Interaction 09 - South America*, São Paulo, 2009, pp. 190-195.

[Dea12] Takahashi, Dean. (2012, Jan.) Xbox 360 surpasses 66M sold and Kinect passes 18M units. Disponível em: http://venturebeat.com/2012/01/09/xbox-360-surpassed-66m-sold-and-kinect-has-sold-18m-units/

[Dia09] Dias, Daniel B., Madeo, Renata C. B. , Rocha, Thiago , Biscaro, Helton H. e Peres, Sarajane M. , "Hand Movement Recognition for Brazilian Sign Language: A Study Using Distance-Based Neural Networks," in *Proceedings of International Joint Conference on Neural Networks*, Atlanta, 2009, pp. 14-19.

[Elm08] Elmezain, M. , Al-Hamadi, A. , Appenrodt, J. e Michaelis, B. , "A Hidden Markov Model-Based Continuous Gesture Recognition System for Hand Motion Trajectory," in *ICPR 2008: 19th International Conference on Pattern Recognition*, Tampa, 2008, pp. 1,4,8-11.

[Fan07] Fang, Gaolin , Gao, Wen e Zhao, Debin , "Large-Vocabulary Continuous Sign Language Recognition Based on Transition-Movement Models," *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS*, vol. 37, pp. 1,9, 2007.

[Fei06] Fei-Fei, L. , Fergus, R. e Perona, P , "One-Shot learning of object categories," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28(4), pp. 594 - 611, 2006.

[Fer05] Fernandes, Anita Maria da Rocha, *Inteligência Artificial: Noções Gerais*. Florianópolis, Brasil: Visual Books, 2005.

[Fer07] Fernandes, S. F., "Avaliação em língua portuguesa para alunos surdos: algumas considerações," 2007.

[Fra13] Francois, J. M. (2013, Nov.) Jahmm - An implementation of Hidden Markov Models in Java. Disponível em: https://code.google.com/p/jahmm/

[Geb09] Gebran, Maurício Pessoa, *Tecnologias Educacionais*. Curitiba: IESDE Brasil S.A, 2009.

[Gom06] Gomes, C. C. S., "Entendendo a legislação de LIBRAS," *Arqueiro, nr. 14*, p. 17, 2006.

[Hol75] Holland, John , *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Cambridge, Massachussets: MIT Press, 1975.

[Hua90] Huang, X. D., Ariki, Y. e Jack, M. A. , "Hidden Markov Models For Speech Recognition," *Edinburgh Information Technology Series*, p. 276, 1990.

[Hua93] Huang, X. D., Hon, H. W. , Hwang, M. Y. e Lee, K. F. , "A comparative study of discrete, semicontinuous, and continuous hidden Markov models," *Computer Speech and Language*, vol. 7, pp. 359-368, 1993.

[IBG11] IBGE. Censo demográfico 2000 - Instituto Brasileiro de Pesquisa Estatística. Disponível em: http://www.ibge.gov.br/home/presidencia/noticias/27062003censo.shtm

[Jon02] Jones, Michael J. e Rehg, James M. , "Statistical Color Models with Application to Skin Detection," *Int. J. of Computer Vision*, vol. 46(1), pp. 81-96, Janeiro 2002.

[Kim08] Kim, Jonghwa , Wagner, Johannes , Rehm, Matthias e André, Elisabeth , "Bi-channel Sensor Fusion for Automatic Sign Language Recognition," in *FG '08. 8th IEEE International Conference on Face & Gesture Recognition*, Augsburg, 2008, pp. 1-6.

[KoA09] Ko, Albert Hung-Ren, Cavalin, Paulo Rodrigo , Sabourin, Robert e Britto Jr., Alceu Souza , "Leave-One-Out-Training and Leave-One-Out-Testing Hidden Markov Models for a Handwritten Numeral Recognizer: The Implications of a Single Classifier and Multiple Classifications," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. 31, p. 2168, Dezembro 2009.

[Koh95] Kohavi, Ron , "A study of cross-validation and bootstrap for accuracy estimation and model selection," *IJCAI'95 Proceedings of the 14th International joint Conference on artificial intelligence*, vol. 2, no. 14, pp. 1137–1145, 1995.

[Lin80] Linde, Y. , Buzo, A. e Gray, R. M. , "An algorithm for vector Quantization Design," *IEEE Transations on Communications*, vol. COM-28, pp. 84-95,

Jan. 1980.

[Lob06] Lobato, Martins C, "O surdo e o processo de formação e qualificação profissional," *Arqueiro, nr.14*, p. 14, 2006.

[Low04] Lowe, David G., "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, no. 2, pp. 91-110, Janeiro 2004.

[Mak85] Makhoul, J. , Roucos, S. e Gish, H. , "Vector Quantization in Speech Coding," *Proceedings of the IEEE*, vol. 73, pp. 1551-1558, Novembro 1985.

[Mat10] Mattar, João , *Game em educação: como os nativos digitais aprendem*, Hall, Prentice , Ed.: Pearson, 2010.

[Men08] Mendonça, Vinícius Godoy e Pozzer, César Tadeu , "A Framework for Genetic Algorithms in Games," *VII Brazilian Symposium on Computer Games and Digital Entertainment*, pp. 72-75, 2008.

[Mic10] Microsoft Press. (2010, Mar.) PrimeSense Supplies 3-D-Sensing Technology to "Project Natal" for Xbox 360. Disponível em: http://www.microsoft.com/Presspass/press/2010/mar10/03-31PrimeSensePR.mspx?rss_fdn=Press%20Releases

[Pau09] Paulraj, M. P., Yaacob, Sazali , Desa, Hazry e Majid, Wan Mohd Ridzuan Wan Ab , "Gesture Recognition System for Kod Tangan Bahasa Melayu (KTBM) Using Neural Network," in *5th International Colloquium on Signal Processing & Its Applications (CSPA)*, Kuala Lumpur, 2009, pp. 19-22.

[Pim10] Pimenta, Nelson e Quadros, Ronice Müller , *Curso LIBRAS 1*, 4th ed.: Vozes, 2010.

[Por88] Poritz, A. B., "Hidden Markov Models: A Guided Tour," *Proceedings of the IEEE International Conf. On Acoustic, Speech, and Signal Processing*, pp. 7-13, 1988.

[Rab89] Rabiner, Laurence R., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE, Vol. 77,No. 2*, pp. 257-286, Fevereiro 1989.

[Rab93] Rabiner, L. R. e Juang, B-H , *Fundamentals of speech recognition*. New Jersey, Englewood Clifss: Prentice Hall Inc, 1993.

[Sch09] Schwab, Brian , *AI Game Engine Programming*, 2nd ed.: Charles River Media, 2009.

[Tom09] Tomaschitz, Jouglas Alves e Facon, Jacques , "Skin Detection Applied to Multi-Racial Images," *16th International Conference on Systems, Signals and Image Processing* , pp. 1-3, 2009.

[Zaf11] Zafrulla, Zaor , Brashear, Helene , Starner, Ted , Hamilton, Harley e Presti, Peter , "American sign language recognition with the kinect," *ICMI '11 Proceedings of the 13th international conference on multimodal interfaces*, pp. 279-286, 2011.