RAFAEL DE SOUZA

RECONHECIMENTO DE GESTOS USANDO DISTÂNCIA DE CADEIAS DE DESCRITORES

Dissertação 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

2014

RAFAEL DE SOUZA

RECONHECIMENTO DE GESTOS USANDO DISTÂNCIA DE CADEIAS DE DESCRITORES

Dissertação 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: Visão Computacional.

Orientador: Prof. Dr. Alceu Souza Britto Júnior
Coorientador: Prof. Dr. Jacques Facon

CURITIBA
2014

SOUZA, Rafael de
Reconhecimento de Gestos usando Distância de Cadeias de Descritores. Curitiba, 2014.

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

1. Reconhecimento de gestos 2. Descrição de características 3. Classificação de gestos 4. Comparação de cadeia de caracteres. I. Pontifícia Universidade Católica do Paraná. Escola Politécnica. Programa de Pós-Graduação em Informática

Dedico este trabalho à minha família pelo amor, suporte e incentivo para o desenvolvimento deste trabalho. E a meus amigos pelo apoio.

Agradecimentos

Agradeço primeiramente à minha família pelo apoio, carinho e suporte para a realização deste trabalho.

Agradeço aos meus amigos pessoais e de trabalho que incentivaram e compreenderam os momentos de ausência necessários para o estudo e a dedicação durante as etapas de estudo.

Agradeço também ao meu orientador Prof. Dr. Alceu Souza Britto Júnior pelo tempo dedicado a me orientar nesse trabalho, à sua disponibilidade para analisar o que estava sendo desenvolvido e indicar os caminhos que poderiam ser seguidos para a conclusão do mesmo. Aos professores e colegas da pós-graduação que em cada área contribuíram compartilhando seus conhecimentos e experiências.

Agradeço à CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pela concessão da bolsa durante todo o período de realização deste mestrado e à Pontifícia Universidade Católica do Paraná por me acolher como aluno neste período.

Muito obrigado.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

σ	Variância de suavização da função Gaussiana
$\mathcal{H}(\boldsymbol{x}, \sigma)$	Matriz Hessian
$\boldsymbol{x}$	Ponto na imagem
$L_{xx}(\boldsymbol{x}, \sigma)$	Convolução da derivada de segunda ordem de Gaussian
d	Transformadas da sub-região das imagens
$\mathcal{LD}(s_1, s_2)$	Função de distância de Levenshtein
s_1, s_2	*Strings*
$D\mathcal{LD}(s_1, s_2)$	Função de distância de Damerau-Levenshtein
$S(s_1, s_2)$	Função de similaridade entre *strings*
$\Phi_j(s_1, s_2)$	Função de distância Jaro-Winkler
K	Número de classes para o método k-Means que limita o número de entradas do *codebook* de descritores
L	Número de classes para o método k-Means que limita o número de entradas do *codebook* de histogramas
$V(n, o)$	Matriz de histograma dos vídeos
$C(m, o)$	Matriz do *codebook* de histogramas
n	Tamanho da matriz de histogramas
m	Tamanho da matriz de *codebook*
o	Tamanho do histograma
$P(p)$	Vetor de proximidade
$B(p)$	Vetor de cadeias de caracteres de vídeos de treinamento
p	Tamanho do vetor de cadeias de caracteres de vídeos de treinamento
v	Cadeia de caracteres de um vídeo de validação
D	Dicionário de caracteres utilizado para assinatura dos gestos

Lista de Abreviaturas

3D	Três Dimensões
ASCII	*American Standard Code for Information Interchange*
AVI	*Audio Visual Interleave*
BSD	*Berkeley Software Distribution*
CBK	*Codebook*
DARPA	*Defense Advanced Research Projects Agency*
E-MHI	*Extended Motion History Image*
FSM	*Finite State Machine*
GEI	*Gait Energy Information*
GLOH	*Gradient Location and Orientation Histogram*
HMDB	*Human Motion Database*
HMM	*Hidden Markov Model*
HoF	*Histogram of Flow*
HoG	*Histogram of Gradients*
HOSVD	*Higher-Order Singular Value Decomposition*
IBGE	Instituto Brasileiro de Geografia e Estatística
INV	*Inversed Recording*
KTH	*Kungliga Tekniska Högskolan*
LIBRAS	Língua Brasileira de Sinais
MHI	*Motion History Image*
MSE	*Multiview Spectral Embedding*
ONI	*Open Natural Interaction*
OpenCV	*Open Source Computer Vision Library*
OpenNI	*Open Natural Interaction*

RGB	*Red Green Blue*
SDK	*Software Development Kit*
SIFT	*Scale-Invariant Feature Transform*
SKL	*Skeleton*
SURF	*Speeded Up Robust Features*
U-SURF	*Upright Speeded Up Robust Features*

Resumo

A área de reconhecimento de gestos tem se expandido com a popularização de variados meios de captura de informação visual. Esta disseminação do reconhecimento de gestos em sistemas permite diversas aplicações para interação humano-computador. Como um meio para auxiliar o aprendizado da Linguagem Brasileira de Sinais (LIBRAS) vê-se a possibilidade de desenvolver sistemas de jogos educacionais para alfabetização de surdos em LIBRAS e na Língua Portuguesa. Este trabalho propõe um método de classificação de gestos, baseado no cálculo de distância entre cadeias de caracteres geradas a partir de descritores extraídos de gestos em vídeo. A abordagem proposta utiliza um vocabulário de nove gestos capturados com Microsoft Kinect™.

Palavras-chave: Reconhecimento de gestos, Descrição de características, LIBRAS, Comparação de cadeia de caracteres.

Abstract

The area of gesture recognition has expanded with the popularization of various means of capturing visual information. This spread of gesture recognition allows systems apply many uses for human-machine interaction. One possible application is an educational game to assist deaf people to learn Brazilian Sign Language (LIBRAS) and Portuguese Language. This work proposes a method of gesture classification, using a set of specific words, based on the comparison of *strings* generated on descriptors extracted from gesture videos. The proposed approach uses a vocabulary with nine gestures recorded in depth videos captured with Microsoft Kinect ™.

Keywords: Gesture recognition, Feature extraction, LIBRAS, String distance.

Capítulo 1

Introdução

O reconhecimento de padrões é uma das grandes características do cérebro humano. Reconhecer formas e símbolos permitiu ao ser humano classificar outros animais, reconhecer outros seres humanos, principalmente, estabelecer formas de comunicação.

Um aspecto que se destaca na comunicação são os gestos. O reconhecimento de gestos permite um meio de comunicação independente da linguagem falada e com formatos universais de comunicação.

Na área de visão computacional o reconhecimento de gestos segue alguns pontos básicos. Para o computador o reconhecimento requer processamento de imagens, o reconhecimento de padrões e a categorização dos padrões reconhecidos.

Este trabalho apresenta um método para o reconhecimento de gestos visando um conjunto de palavras específico e suas possibilidades na solução de problemas de interpretação de linguagens de sinais.

1.1. Definição do Problema

Os gestos são utilizados como uma forma de comunicação comum na vida cotidiana. O ser humano está acostumado a interpretar gestos baseado em seu conhecimento e experiência, permitindo assim o reconhecimento e compreensão da mensagem passada. Os gestos também podem ser aplicados como forma de interação entre o humano e o computador, aumentando a quantidade de recursos e possibilidades de interação com sistemas. Alguns padrões de gestos são utilizados também como uma forma de comunicação e definidos como uma forma de linguagem auxiliar à falada.

No Brasil uma das linguagens utilizada é a LIBRAS, definida como um padrão de linguagem de sinais nacional que se tornou obrigatória no ensino escolar fundamental. A LIBRAS é uma linguagem de sinais adotada no Brasil e reconhecida como meio de comunicação legal em [BRA02] pela Lei nº 10.436 desde 2002, e detalhado em [BRA05] pelo Decreto nº 5.626, incluindo a LIBRAS como disciplina curricular na formação de professores de nível médio, superior e nos cursos de fonoaudiologia.

A LIBRAS é uma linguagem independente e possui construções e estruturas diferentes. Segundo Pereira [PER08] os alunos surdos devem ser expostos tanto a leitura como a produção de diferentes gêneros e tipos textuais que devem ser vivenciadas primeiramente na LIBRAS, a sua primeira língua. A escola deve proporcionar o aprendizado em LIBRAS e com base nela a Língua Portuguesa em sua modalidade escrita. Crianças surdas podem chegar à escola sem uma linguagem escrita adquirida expostas apenas à linguagem oral.

Um sistema computacional capaz de auxiliar a comunicação entre pessoas surdas e ouvintes, principalmente no ensino pré-escolar, pode facilitar a inclusão social da criança surda e adiantar o ensino para a fase escolar. A base de vídeos desenvolvida por Mendonça [MEN14] sugere a utilização do estilo de jogos de aventura para a construção de um sistema utilizando um conjunto restrito de palavras. A escolha do estilo possui como base a proposta de Mattar [MAT10], a qual indica os três motivos:

1. Jogo de aventura possui uma narrativa simples;
2. Conjunto pequeno de palavras para interação;
3. Utilização do tema para discussão aluno-professor.

Reconhecer gestos depende de uma grande quantidade de informações e um processo de treinamento de classificadores para melhora de resultados. Porém, isto aumenta o custo dos sistemas e pode até inviabilizar a sua aplicação.

Com o uso da base de vídeos capturados a partir do Kinect é apresentado o problema de reconhecer gestos utilizando os quadros dos vídeos de forma completa limitando apenas a definição de escala de cores, sem utilização de dados de esqueleto ou métodos de rastreamento da mão.

O segundo problema inerente ao reconhecimento de gestos é a codificação dos vídeos em cadeias de caracteres de modo a manter uma quantidade de características do movimento executado que faça possível a diferenciação de cada gesto. Outro aspecto do problema abordado é a classificação baseado nas cadeias de caracteres de cada vídeo utilizando algoritmos de distância de edição.

1.2. Motivação

Os sistemas de reconhecimento de gestos dependem geralmente de bases de vídeos e um extensivo processo de treinamento de classificadores, exigindo assim uma grande base de vídeos e um classificador capaz de diferenciar as características de cada gesto com uma velocidade de resposta suficiente para a interação rápida.

Um método de extração de características e classificação baseado em uma cadeia de caracteres utilizada como identificador do vídeo simplificaria o processo de classificação e facilitaria a popularização de sistemas de reconhecimento de gestos.

Sistemas de reconhecimento de gestos podem ser utilizados também para reconhecer linguagem de sinais facilitando a aprendizagem. Segundo o IBGE [IBG10] 5,1% das pessoas declaram ter deficiência auditiva, sendo que 1,7 milhões ouvem com grande dificuldade e mais de 7 milhões declaram alguma dificuldade na audição. No Censo 2010 [IBG10] mais de 344.206 pessoas declararam que não conseguem ouvir de modo algum, e dentre estes 63.709 são crianças em idade escolar.

A aprendizagem da linguagem de sinais se torna importante também para a alfabetização do indivíduo. Segundo Pereira [PER08] a LIBRAS tem mesma função que a Língua Portuguesa, na modalidade oral, tem para os ouvintes. E por meio dela as crianças surdas atingem os objetivos propostos pela escola, incluindo o aprendizado da língua escrita. Um exemplo de utilização do reconhecimento de gestos são os jogos educacionais que utilizam uma quantidade bem definida de palavras e podem ser desenvolvidos com uma base específica de gestos.

1.3. Desafio

O principal desafio desse estudo é definir um conjunto de descritores que permita a identificação do gesto em vídeo, considerando os aspectos identificados em cada quadro do vídeo e, para tal utilizar as características a partir de pontos de interesse identificados nas mãos do interlocutor. Evitando o uso de características específicas como o esqueleto calculado pelo Kinect® ou a necessidade de maior resolução para o tratamento das configurações da mão.

Os descritores utilizados são definidos a partir de exemplares de gestos em vídeos, exigindo assim uma extração de características que possa representar a distinção de cada gesto. O descritor deve desconsiderar as características físicas de cada indivíduo.

Outro desafio deste trabalho é a definição de um método de classificação baseado na comparação das cadeias de caracteres dos exemplares de gestos em vídeos utilizados a partir de uma base de dados com a cadeia de caracteres do vídeo a ser classificado.

1.4. Objetivos

Nos tópicos a seguir são descritos os objetivos gerais e específicos a serem alcançados com a pesquisa.

1.4.1. Objetivo geral

Desenvolver um método de reconhecimento de gestos em vídeo capturados com o Kinect™ para um conjunto específico de palavras comumente utilizadas em jogos educacionais. O método tem como premissa a classificação dos vídeos usando distâncias de cadeias de descritores baseados em vídeos de referência dos gestos que representam uma palavra específica. Para tal, espera-se definir um conjunto de descritores para representação dos gestos em forma de uma cadeia de caracteres e então aplicar algoritmos que permitam classificar tais cadeias.

1.4.2. Objetivos específicos

Os objetivos específicos propostos para este trabalho são os seguintes:

- Avaliar a extração de características sem utilização das informações de esqueleto e rastreamento das mãos.
- Configurar e aplicar o descritor de características SURF por quadro para definição do descritor de cada sinal.
- Criar um *codebook* balanceado com as características extraídas dos vídeos.
- Implementar um método para descrever uma assinatura que referencia cada sinal em vídeo utilizando histogramas das características.
- Comparar algoritmos de cálculo de distância de cadeias de caracteres aplicados a classificação dos vídeos.
- Desenvolver um método de classificação das assinaturas que referenciam os gestos definidos utilizando algoritmos de comparação de cadeias.
- Definir um método de avaliação dos resultados.

1.5. Contribuições

Este trabalho tem como contribuição social a apresentação de uma proposta para utilização de recursos de reconhecimento de gestos aplicados à área de jogos educacionais. Jogos educacionais podem ser desenvolvidos utilizando métodos que exigem poucos recursos computacionais para o reconhecimento de sinais específicos da linguagem de LIBRAS permitindo a criação de uma ponte entre a língua gestual e a linguagem escrita de forma lúdica e interativa.

Pelo aspecto tecnológico um sistema de reconhecimento de um conjunto específico de gestos é esperado como contribuição do projeto, fornecendo uma lista de possibilidades de gestos reconhecidos para escolha do usuário.

Este trabalho tem como contribuição a definição de um método para classificação de gestos baseado em comparação de cadeias de caracteres, onde serão descritos:

a) A avaliação da utilização do algoritmo de descrição de características de imagem SURF associado à informação espacial como descritor de quadros de vídeo capturados com o Microsoft Kinect® sem a utilização da informação de esqueleto do personagem e rastreamento de partes específicas da imagem.
b) A implementação e análise de algoritmos para a comparação de cadeias de caracteres, considerando aspectos como desempenho, alinhamento e custo.
c) Definição de uma forma de representar um vídeo de profundidade em cadeia de caracteres mantendo os aspectos de movimentos de cada gesto em vídeo.

1.6. Estrutura do Trabalho

No Capítulo 1 são apresentados os objetivos do trabalho, o problema, desafios e contribuições da pesquisa. O Capítulo 2 descreve as fundamentações teóricas necessárias para completa compreensão do conteúdo deste trabalho e o os trabalhos desenvolvidos similares ao trabalho apresentado. No Capítulo 4 é descrita a base de vídeos a ser utilizada na execução do trabalho e o método de implementação do trabalho seguido do Capítulo 5 onde são apresentados os testes para configuração do método proposto e os resultados obtidos no desenvolvimento do trabalho. O Capítulo 6 apresenta as conclusões e sugestões para trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

Neste capítulo são apresentados alguns fundamentos necessários para melhor compreensão dos componentes e métodos aplicados, assim como trabalhos relacionados disponíveis na literatura.

2.1. Bibliotecas de Processamento de Vídeos

O pré-processamento dos vídeos é realizado para separar o formato do vídeo em cores do vídeo de profundidade e armazenar em um formato mais simplificado e padrão de vídeo.

2.1.1. OpenNI

Segundo Falahati [FAL13] o consórcio OpenNI foi estabelecido em 2010, pela parceria de empresas como a PrimeSense Natural Interactions, Willow Garage, Open Perception entre outras, para padronização, compatibilização e interoperação de dispositivos e aplicações de *Natural Interaction*. O OpenNI 2.2 SDK foi desenvolvida em C++ com o intuito de estabelecer uma biblioteca para interação com sensores 3D, como pode ser observado na Figura 2.1.

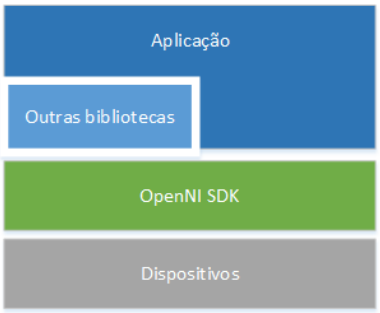


Figura 2.1: Arquitetura do OpenNI SDK.

A camada OpenNI SDK é composta por classes para acesso a dispositivos com vídeos de fontes variadas, arquivos de vídeo e leitura dos quadros do vídeo. As principais classes e suas funcionalidades são:

- OpenNI: Fornece uma única entrada para a API e também fornece acesso a dispositivos, eventos relacionados aos dispositivos, versão e informações de erro;
- Device: Fornece uma interface para um dispositivo conectado no sistema e fornece acesso aos vídeos;
- VideoStream: Classe abstrata de definição de vídeo obtido de um dispositivo;
- VideoFrameRef: Classe abstrata para acesso aos quadros do vídeo obtido;

A biblioteca OpenNI identifica também a posição das juntas chave do corpo humano como as mãos, ombros, joelho, cabeça etc. para representação do esqueleto da imagem. O arquivo vídeo capturado é gravado em formato ONI com a imagem RGBD.

A principal vantagem da biblioteca é o grande suporte e interoperação com outras bibliotecas de processamento de vídeo como o OpenCV.

2.1.2. OpenCV

O OpenCV [OPE14] é uma biblioteca de código aberto, sob a licença BSD, para visão computacional e aprendizagem de máquina.

Desenvolvida em C e C++, com interfaces para Python e Java, a biblioteca possui compatibilidade entre plataformas, dentre elas sistemas operacionais para computadores pessoais como Windows, MacOS, Linux, FreeBSD e sistemas móveis como Android, Maemo e iOS.

A biblioteca contém disponível mais de 2.500 algoritmos otimizados para leitura de vídeos, processamento de imagem e algoritmos proprietários desenvolvidos com função acadêmica. As principais classes e suas funcionalidades são:

- Elementos básicos: principais funcionalidades para processamento de dados como vetores, matrizes, pontos etc.;
- Processamento de imagem: métodos para processamento de imagem como detecção de borda, filtros etc.;
- Interface com usuário: componentes para criação de interface com o usuário;
- Vídeo: ferramentas para leitura e análise de vídeo;
- Câmera: ferramenta para calibração de câmera e reconstrução 3D;
- Características 2D: algoritmos para detecção de características e objetos auxiliares para pontos de interesse;
- Detecção de objetos: algoritmos para detecção de objetos em cena;
- Aprendizado de máquina: algoritmos para aprendizagem de máquina, SVM, árvores de decisão etc.;
- Agrupamento e busca: algoritmos para agrupamento e busca em espaço multidimensional;
- GPU: algoritmos para aceleração por GPU;
- Fotografia: algoritmos para restauração de fotografias;
- Algoritmos patenteados: algoritmos patenteados para utilização acadêmica como o SURF e o SIFT;
- Entre outros: suporte ao legado, superresoluções, visualizadores 3D.

Os principais recursos a serem utilizados neste trabalho são as classes de elementos básicos, métodos para leitura de vídeo e processamento de imagens e a implementação disponível de algoritmos patenteados para detecção de características de imagem.

2.2. Descrição de Características

A descrição de características é o processo utilizado para simplificar as informações de uma imagem. O processo é realizado para identificar partes da imagem que se destacam variando ou não a rotação e posicionamento na imagem. Ao identificar os pontos de interesse o algoritmo cria vetores que descrevem os pontos identificados. Com o vetor de descrição é possível processar outra imagem e identificar se os mesmos aspectos se correspondem. Por exemplo, se uma imagem de uma capa de livro é processada e seu descritor é detectado, é possível localizar este livro em outra imagem contendo vários objetos diferentes.

No processo de reconhecimento humano algumas características são importantes na descrição da imagem, como por exemplo a independência da escala e a variação da rotação. Os algoritmos encontrados na literatura que descrevem as características com essas propriedades são o SIFT e o SURF.

2.2.1. SIFT

O algoritmo SIFT foi desenvolvido por David Lowe [LOW99] para detectar e descrever características de uma imagem de forma independente da escala, orientação, iluminação e oclusão do objeto. Existem diversas aplicações de descritores para a visão computacional, dentre elas estão o reconhecimento de objetos, mapeamento e navegação, modelagem 3D, rastreamento de vídeo e reconhecimento de gestos.

O primeiro passo é extrair os pontos chave dos objetos em uma imagem de treinamento e armazenar em uma base de dados. Segundo Lowe [LOW99] a abordagem escolhida é selecionar as chaves baseado no ponto máximo e mínimo da função Gaussiana aplicada no espaço escalar. A seleção das chaves pode ser calculada, de modo eficiente, construindo uma pirâmide de imagem com escalas alteradas em cada nível. Após a construção da pirâmide de imagem os pontos-chave são localizados nas regiões de alta variação na escala.

$$g(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-x^2/2\sigma^2} \qquad (2.1)$$

A Equação 2-1 representa a função Gaussiana, onde σ é a variância, aplicada em uma dimensão nos sentidos horizontais e verticais. Para Lowe [LOW99] na localização da chave, todas as operações de suavização podem ser feitas usando um $\sigma = \sqrt{2}$, que pode ser aproximada com precisão suficiente usando um núcleo com 7 pontos de exemplo.

Conforme conclui Lowe [LOW99], dada a localização, escala e orientação estável para cada ponto-chave é então possível descrever a região da imagem de modo não variável às transformações. Além das transformações já verificadas no ponto-chave outros aspectos devem ser observados, como a projeção 3D e outras pequenas mudanças na geometria do objeto. Com base nos experimentos de Edelman, Intrator e Poggio [EDE97] que consideram as propriedades de resposta dos neurônios no córtex visual,

onde a característica de posição pode variar por uma região pequena enquanto a orientação e frequência espacial específica são mantidas. A implementação utiliza os mesmo gradientes e orientações para cada nível da pirâmide utilizada na seleção da orientação.

O passo seguinte do algoritmo executa a indexação dos pontos-chave e a comparação com pontos de uma nova imagem, identificando os pontos correspondentes usando distância Euclidiana.

Em uma comparação apresentada por Mikolajczyk e Schimid [MIK05] o GLOH apresenta os melhores resultados, seguido de perto pelo algoritmo SIFT, que apresenta melhor robustez e o caráter distinto do descritor por região SIFT.

Um outro algoritmo, proposto por Bay et. al. [BAY08], mostrou desempenho similar ao SIFT e com processamento mais rápido.

2.2.2. SURF

Segundo Bay et. al. [BAY08] o SURF (...) se baseia em propriedades similares ao SIFT, reduzindo a complexidade. O processo se divide em 3 passos: localizar os pontos de interesse, definir a orientação dos pontos e extrair o descritor para comparação. Uma variação do descritor é o U-SURF que não utiliza a rotação e pode ser calculado de modo mais rápido.

No primeiro passo são detectados os pontos de interesse (cantos, bolhas e junções em T), e então são armazenados vetores de características para cada ponto de interesse criando o descritor de características e no último passo os descritores de diferentes imagens são comparados.

Bay [BAY08] propõe o uso da matriz Hessian para o detector pelo bom desempenho no cálculo e precisão, usando a determinante da matriz Hessian para determinar a escala e localização. A Equação 2.2 mostra a matriz de Hessian aplicada a um dado ponto $\boldsymbol{x}$ de uma imagem *I* em uma escala σ,

$$\mathcal{H}(\boldsymbol{x},\sigma) = \begin{bmatrix} L_{xx}(\boldsymbol{x},\sigma) & L_{xy}(\boldsymbol{x},\sigma) \\ L_{xy}(\boldsymbol{x},\sigma) & L_{yy}(\boldsymbol{x},\sigma) \end{bmatrix} \tag{2.2}$$

sendo $L_{xx}(\boldsymbol{x},\sigma)$ a convolução da derivada de segunda ordem de Gaussian conforme a Equação 2.3, aplicada à imagem *I* no ponto **x**.

$$\frac{\partial^2}{\partial x^2} g(\sigma) \qquad (2.3)$$

Segundo Bay et. al. [BAY08] Gaussian são otimizados para uma análise de espaço-escala, entretanto precisa ser discretizado e cortado. A Figura 2.2 apresenta (a) o filtro Gaussian de segunda ordem parcial derivada na direção-y (L_{yy}), (b) direção-xy (L_{xy}) e as aproximações equivalente com (c) direção-y (D_{yy}) e (d) direção-xy (D_{xy}). As regiões cinza equivalem a 0. Uma aproximação usando os filtros (c) e (d) demonstrados na Figura 2.2, é aplicada para atingir melhor desempenho usando imagem integral.

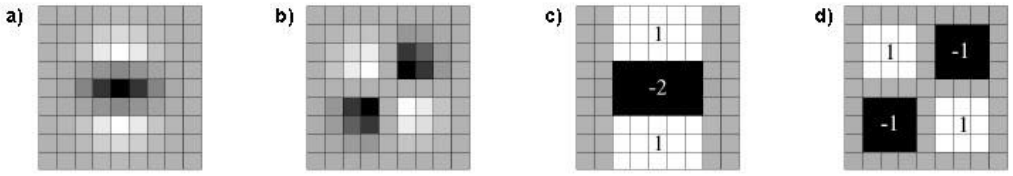


Figura 2.2: Filtros gaussianos de segunda ordem e a aproximação equivalente.

Ao invés de aplicar o filtro de forma iterativa reduzindo o tamanho da imagem, os tamanhos dos filtros são aumentados alterando o valor de escala σ. As localizações dos pontos de interesse são feitas usando a determinante da matriz de *Hessian*. O segundo passo é identificação da orientação dos pontos de interesse, um passo ignorado na aplicação do U-SURF, para isso são determinadas as direções calculando a transformada de *Haar* nas direções *x* e *y* ao redor dos vizinhos de raio 6*s*, com *s* sendo a escala do ponto de interesse detectado. O resultado é representado como um vetor da transformada horizontal e vertical. Bay et. al. [BAY08] aplicam o cálculo da orientação do ponto é estimada calculando a soma de todas as transformadas dentro de uma janela cobrindo um ângulo de $\frac{\pi}{3}$.

Um quadrado ao redor do ponto de interesse seguindo a orientação selecionada e então o quadrado é subdividido em pedaços de 4 x 4 subpartes quadradas e em um espaço regular de 5 x 5. As transformadas calculadas da horizontal (d_x) e vertical (d_y) de cada sub-região é somada e são extraídos os valores absolutos $|d_x|$ e $|d_y|$. Assim cada subparte tem um vetor de quatro dimensões que representa o descritor.

2.3. Distância de *Strings*

Funções de distância de *strings* são uma forma de medir a similaridade e diferença entre duas cadeias de caracteres. Em um trabalho apresentado por Levenshtein [LEV66], na década de sessenta, uma abordagem foi proposta para a comparação de duas cadeias de caracteres usando uma comparação entre códigos binários por remoção, inserção e inversão, conhecido como distância de Levenshtein ou distância de edição.

Na mesma década de sessenta, alguns anos antes Damerau [DAM64] apresentou um estudo sobre detecção e correção de erros de grafia em sistemas de indexação coordenadas e recuperação. Uma proposta de algoritmo foi realizada acrescentando o tratamento da operação de transposição ao algoritmo de distância de Levenshtein, nomeado distância de Damerau-Levenshtein.

Outros algoritmos foram desenvolvidos para comparação de *strings*, incluindo elementos mais complexos como fonética, marcadores, métodos estatísticos baseados em gramática e são descritos a seguir.

2.3.1. Distância de Levenshtein

Levenshtein [LEV66] desenvolveu um método para avaliar transmissão de informação binária considerando três tipos de falhas: inversão, remoção e inserção; criando um código capaz de corrigi-los. O conceito proposto busca definir o menor número de operações necessárias para que um dado transmitido seja igualado a um segundo dado. O método representado por $\mathcal{LD}(s_1, s_2)$ recebe duas *strings* como parâmetro e tem como retorno o número que representa quantas operações são necessárias para deixar a *string* s_1 igual a *string* s_2 descrito no Algoritmo 2.1.

Algoritmo 2.1 Cálculo de distância de Levenshtein

```
{recebe as strings de parâmetros S1 e S2}
n ← Tamanho(S1)
m ← Tamanho(S2)
D ← Matriz(n, m)

para i ← 0 até n faça
    D(i, 0) ← i

para j ← 0 até m faça
    D(0, j) ← j

para i ← 1 até n faça
```

```
    para j ← 1 até m faça
        se (S1(i – 1) == S2(j – 1)) então
            custo =  0;
        senão
            custo =  1;
        fim se

        D(i, j) ← min(D(i – 1, j) + 1, D(i, j – 1) + 1, D(i – 1, j – 1) + custo)
    fim para
fim para

{retorna D(n, m)}
```

Este método pode ser exemplificado utilizando as *strings* “abcdef” e “abdfe”, inicialmente são armazenados os tamanhos de cada *string* e criada uma matriz correspondente às distâncias. Depois é percorrida a matriz verificando cada valor e considerando o custo de alteração sendo 1 se os caracteres forem diferentes e 0 se forem iguais e as posições seguintes e anteriores para verificar por caracteres faltantes.

Tabela 2.1: Matriz de resultado do algoritmo Levenshtein

	a	**b**	**d**	**f**	**e**
a	0	1	1	1	1
b	1	0	1	2	2
c	1	1	1	2	3
d	1	2	1	2	3
e	1	2	2	2	2
f	1	2	3	2	**3**

A Tabela 2.1 demonstra cada passo do algoritmo e o resultado da matriz de comparação. A primeira coluna demonstra a comparação entre o caractere da primeira *string* e todos os outros caracteres da segunda *string* considerando o *custo = 1* caso os caracteres sejam diferentes. Quando o caractere ‘d’ é comparado ao caractere ‘c’ o erro resulta em um custo, o próximo caractere comparado é identificado como igual, os dois são caracteres ‘d’ que indica a operação de remoção do caractere ‘c’. A próxima comparação realizada entre o caractere ‘e’ e o caractere ‘f’ resulta em um acréscimo do custo. Para o último caractere ‘f’ em comparação com o caractere ‘e’ resulta em outro acréscimo, ou seja, para o método aplicado de $\mathcal{LD}("abcdef","abdfe")$ a distância de edição é igual a 3.

2.3.2. Distância de Damerau-Levenshtein

Damerau [DAM64] apresentou um trabalho sobre detecção e correção de erros de grafia em sistemas de indexação coordenadas e recuperação. Analisando as rejeições do sistema que necessitam de verificação de erros dos dados de entrada Damerau [DAM64] chegou à conclusão que 80% dos erros pertencem a quatro classes:

1. Uma letra estava errada;
2. Uma letra estava faltando;
3. Uma letra adicional estava inserida;
4. Duas letras adjacentes estavam invertidas.

Considerando esses erros foi proposto um algoritmo que utiliza operações de inserção, exclusão, substituição e transposição. O Algoritmo 2.2 apresenta o pseudocódigo para aplicação do método de distância de edição incluindo a verificação de transposição de caracteres adjacentes, o processo utiliza dois parâmetros de entrada em formato de *string* e retorna o número que representa a quantidade de operações necessárias para igualar os parâmetros.

Algoritmo 2.2 Cálculo de distância de Damerau-Levenshtein

```
{recebe as strings de parâmetros S1 e S2}
n ← Tamanho(S1)
m ← Tamanho(S2)
D ← Matriz(n, m)

para i ← 0 até n faça
    D(i, 0) ← i

para j ← 0 até m faça
    D(0, j) ← j

para i ← 1 até n faça
    para j ← 1 até m faça
        se (S1(i – 1) == S2(j – 1)) então
            custo =  0;
        senão
            custo =  1;
        fim se

        D(i, j) ← min(D(i - 1, j) + 1, D(i, j – 1) + 1, D(i - 1, j – 1) + custo)

        se (i > 1 e j > 1e S1(i) == S2(j – 1) e S1(i – 1) == S2(j)) então
            D(i, j) ← min(D(i,  j), D(i – 2, j – 2) + custo)
        fim se
```

fim para
fim para

retorna *D*(*n*, *m*)

Como pode ser observado na Tabela 2.2 o processo de comparação entre as colunas é semelhante ao do algoritmo de Levenshtein, diferenciado apenas pela comparação dos últimos dois caracteres da *string* onde é identificado pelo algoritmo de Damerau-Levenshtein a transposição dos caracteres '*fe*' da segunda *string* em comparação com os últimos caracteres '*ef*' da primeira. Esta identificação resulta em *custo = 1*, diferente do algoritmo anterior que identifica um custo para o penúltimo caractere e outro para o último. Aplicando o método $\mathcal{DLD}("abcdef","abdfe")$ é obtido um valor de distância de 2 edições.

Tabela 2.2: Matriz de resultado do algoritmo Damerau-Levenshtein

	a	**b**	**d**	**f**	**e**
a	0	1	1	1	1
b	1	0	1	2	2
c	1	1	1	2	3
d	1	2	1	2	3
e	1	2	2	2	2
f	1	2	3	2	**2**

A característica principal dos algoritmos de Damerau e Levenshtein é a comparação de uma cadeia de caracteres completa, que pode causar uma distância muito grande entre as *strings*, caso o tamanho de cada uma das *strings* sejam diferentes. Para otimizar a comparação entre *strings* de tamanhos diferentes foram criados algoritmos que consideram pedaços das *strings* na comparação buscando a maior similaridade.

2.3.3. Algoritmo de *Dynamic Time Warping*

DTW é o nome dado para uma série de algoritmos para medir a similaridade entre duas sequências temporais que podem variar em tempo ou velocidade. Segundo Giorgino [GIO08] algoritmos para DTW foram propostos em meados de 1970 para o reconhecimento de fala para explicar as diferenças entre a taxa da pronúncia. Desde então a aplicação mais bem difundida para o algoritmo é o reconhecimento automático de fala, para tratar diferentes velocidades de fala.

Outras aplicações incluem reconhecimento de voz e reconhecimento de assinatura e vídeos onde as variações de tempo indicam um problema que precisa ser alinhado. Por exemplo, as similaridades no padrão de caminhada podem ser detectadas usando DTW, mesmo se uma pessoa estiver andando mais rápido que outra, ou se houver acelerações e desacelerações durante o percurso observado.

De forma geral o DTW é um método que calcula a melhor similaridade entre duas sequências dadas com certas restrições. As sequências são deformadas de forma não linear no tempo para determinar a medida de sua similaridade independente de certas variações não lineares na dimensão temporal. Esse alinhamento da sequência é frequentemente usado para classificação série temporal.

No Algoritmo 2.3 é descrito um método DTW aplicado a duas *strings* de entrada S_1 e S_2. A matriz de distância é criada com valores iniciados com uma variável infinito que pode representar um valor maior que a extensão de valores utilizados na cadeia de caracteres. Após a inicialização a matriz é percorrida e para cada elemento é calculado o custo que envolve uma função de similaridade a ser implementada e retorna a distância absoluta entre os dois símbolos. A posição calculada na matriz de distância é calculada então somando o custo ao mínimo valor vizinho.

Algoritmo 2.3 Cálculo de distância usando DTW

```
{recebe as strings de parâmetros S1 e S2}
n ← Tamanho(S1)
m ← Tamanho(S2)
DTW ← Matriz(n, m)
DTW(0, 0) ← 0

para i ← 1 até n faça
    DTW (i, 0) ← infinito

para j ← 1 até m faça
    DTW(0, j) ← infinito

para i ← 1 até n faça
    para j ← 1 até m faça
        custo =  similaridade(S1(i), S2(j));

        DTW (i, j) ← custo + min(DTW (i - 1, j), DTW (i, j – 1) , DTW (i - 1, j – 1))
    fim para
fim para

retorna DTW (n, m)
```

Utilizando algumas alterações o algoritmo pode ser desenvolvido também usando o recurso de janela de tempo. A janela de tempo limita a execução da comparação utilizando um valor máximo em *w* que representa a quantidade de caracteres utilizado para cada iteração.

2.3.4. Algoritmo de Needleman–Wunsch

O algoritmo proposto por Needleman e Wunsch [NED70] busca otimizar a comparação de *string* localizando a maior similaridade entre duas *strings* para aplicação no sequenciamento de aminoácidos. Segundo Neddleman e Wunsch [NED70] a comparação direta de duas sequências de aminoácidos não é suficiente para estabelecer a relação genética entre duas proteínas. A menor unidade de comparação é o aminoácido e a combinação máxima pode ser definida pelo maior número de aminoácidos de uma proteína que pode ser comparada com outras e ainda permitindo todas as possibilidades de exclusão.

Para otimizar a comparação de aminoácidos a proposta encontrada por Needleman e Wunsch [NED70] foi utilizar um processamento para alinhamento global da *string*. O processo consiste em buscar o maior número de referências considerando uma matriz de similaridade, que identifica uma tabela onde a comparação de cada elemento recebe uma pontuação devido a sua importância do acerto, como o exemplo descrito na Tabela 2.3.

Tabela 2.3: Exemplo de tabela de similaridade para ácidos nucleicos

	A	**G**	**C**	**T**
A	10	-1	-3	-4
G	-1	7	-5	-3
C	-3	-5	9	0
T	-4	-3	0	8

Com base na matriz de similaridade cada elemento da *string* é comparado gerando a matriz de pontuação para localizar o melhor alinhamento.

A criação da matriz de pontuação é dada pela Equação 2-4. Considerando $S(a, b)$ a matriz de similaridade, onde a e b são os caracteres comparados, d sendo a penalização do espaçamento.

$$
\begin{aligned}
F_{0,0} &= 0 \\
F_{i,0} &= i \,.\, d \\
F_{0,j} &= j \,.\, d \\
F_{ij} &= max \begin{cases} F_{i-1,j-1} + S_{A_i B_j} \\ F_{i-1,j} - d \\ F_{i,j-1} - d \end{cases}
\end{aligned}
\tag{2.4}
$$

Após o cálculo da matriz F é processada também a matriz de alinhamento. Para calcular o alinhamento o algoritmo inicia do ponto F_{mn}, onde m é o tamanho da primeira *string* e n é o tamanho da segunda *string*, e compara com os valores das três possíveis origens seguindo a matriz em ordem reversa. Neste caso são avaliados os valores para identificação do resultado da operação realizada:

- se A_i e B_j são alinhados (igual);
- se A_i é alinhado com a penalidade (remoção)
- se B_j é alinhado com a penalidade (inserção).

Mais de uma operação possível pode resultar na comparação de valores, nesse caso são consideradas os múltiplos alinhamentos para encontrar a melhor possibilidade.

A Tabela 2.4 apresenta o resultado da matriz de pontuação apresentado por Calhau et. al. [CAL08] onde as células em azul demonstram o caminho descrito por um dos melhores alinhamentos conseguidos pelo algoritmo de Needleman-Wunsch para as sequências CGATAAC e AACGTTAC.

Tabela 2.4: Alinhamento global obtido pelo Algoritmo Needleman-Wunsch.

	-	A	A	C	G	T	T	A	C
-	0	-1	-2	-3	-4	-5	-6	-7	-8
C	-1	-1	-2	-1	-2	-3	-4	-5	-4
G	-2	-2	-2	-2	0	-1	-2	-3	-4
A	-3	-1	-1	-2	-1	-1	-2	-1	-2
T	-4	-2	-2	-2	-2	0	0	-1	-2
A	-5	-3	-1	-2	-3	-1	-1	1	0
A	-6	-4	-2	-2	-3	-2	-2	0	0
C	-7	-5	-3	-1	-2	-3	-3	-1	1

O algoritmo de Needleman-Wunsch executa o alinhamento baseado em toda a sequência de caracteres em toda *string*, não considerando a possibilidade de espaçamentos diversos e melhores combinações em pequenos trechos de cada *string*. Para

otimizar o alinhamento de cadeias de proteínas Smith e Waterman [SMI81] propuseram a execução de um alinhamento local.

2.3.5. Algoritmo de Smith-Waterman

O alinhamento local proposto por Smith e Waterman [SMI81] procura alinhar fragmentos da cadeia de caracteres. Usando o mesmo conceito de matriz de similaridade e matriz de pontuação cada elemento da *string* é comparado.

A criação da matriz de pontuação é dada pela Equação 2-5. Considerando $S(a, b)$ a matriz de similaridade, onde a e b são os caracteres comparados, d sendo a penalização do espaçamento.

$$F_{0,0} = 0$$
$$F_{i,0} = i \,.\, d$$
$$F_{0,j} = j \,.\, d$$
$$F_{ij} = max \begin{cases} 0 \\ F_{i-1,j-1} + S_{A_i B_j} \\ F_{i-1,j} - d \\ F_{i,j-1} - d \end{cases} \quad (2.5)$$

A diferença proposta por Smith e Waterman é que os resultados negativos são zerados tornando os alinhamentos locais visíveis na matriz de pontuação. A Tabela 2.5 apresenta nas células verde o melhor e mais longo caminho, nas células com borda azul e vermelho estão os caminhos descritos por um dos melhores alinhamentos conseguidos pelo método aplicados por Calhau et. al. [CAL08].

Tabela 2.5: Alinhamento local obtido pelo Algoritmo Smith-Waterman.

	-	A	A	C	G	T	T	A	C
-	0	0	0	0	0	0	0	0	0
C	0	0	0	1	0	0	0	0	1
G	0	0	0	0	2	1	0	0	0
A	0	1	1	0	1	1	0	1	0
T	0	0	0	0	0	2	2	1	0
A	0	1	1	0	0	1	1	3	2
A	0	1	2	1	0	0	0	2	2
C	0	0	1	3	2	1	0	1	3

Após o cálculo da matriz F é processada também a matriz de alinhamento. Para o alinhamento local procura-se na matriz a posição com maior pontuação e executa o processo de comparação das operações como aplicado no alinhamento global.

O alinhamento global e local possuem características que diferenciam a comparação de cadeias de caracteres usando as particularidades da comparação de sequências de proteínas utilizando matrizes de similaridade. Para a comparação de códigos de vídeo a relação entre os descritores extraídos dificultam a geração de uma matriz de similaridade, diminuindo assim a importância da matriz de similaridade na comparação.

2.4. Trabalhos Relacionados

O interesse em encontrar processos para o reconhecimento de gestos aumentou consideravelmente nos últimos anos devido ao avanço das tecnologias de processamento de imagem e os diversos sensores de captura de movimento. Segundo Lisetti [LIS00] o reconhecimento de gestos tem uma ampla variedade de aplicação, como:

1. Desenvolvimento de sistemas para auxílio aos deficientes auditivos.
2. Facilitar o uso do computador para crianças.
3. Técnicas para computação forense.
4. Reconhecimento de linguagens de sinais.
5. Monitoramento médico de pacientes.
6. Detecção de mentira.
7. Navegação e manipulação em ambientes virtuais.
8. Comunicação em vídeo conferência.
9. Auxílio em sistemas de educação a distância.
10. Monitoramento de motoristas.

Mitra e Acharya [MIT07] examinam diversas abordagens utilizadas no reconhecimento de gestos e expressões humanas pela movimentação das mãos, braço, corpo e cabeça, tais como: HMM, filtro de partículas e algoritmos de condensação, abordagem com FSM e *Soft Computing and Connectionist*; descrevendo também as abordagens aplicadas ao reconhecimento de gestos de mão para linguagem de gestos, controle de janelas e controle de robôs. Cada técnica apresentada tem sua aplicação com vantagens e desvantagens, algumas necessitando de algum trabalho e revisão.

Em todas essas abordagens existem duas etapas que mais se destacam apresentadas na Figura 2.3. A primeira é a extração das características das imagens de vídeo usando diversos algoritmos para determinar características diferenciadas para cada tipo de movimento. A segunda etapa consiste na classificação dos gestos utilizando as características identificadas, treinando modelos de classificação e comparando os vídeos.

Figura 2.3: Passos básicos para reconhecimento de gestos

2.4.1. Extração de características

Nesta seção são apresentados os recursos de extração de características utilizados em trabalhos usando diferentes abordagens. Com a popularização de sensores como o Microsoft Kinect® que permite a gravação de imagem de vídeo e imagem de profundidade vários recursos se tornaram disponíveis para novas aplicações. Diversificados tipos de dados podem ser usados para a gravação de gestos para reconhecimento, como: vídeos com imagens coloridas, vídeos de profundidade, rastreamento de mão em cena, esqueleto do ator na imagem e algoritmos desenvolvidos para este processo ou adaptados para essa aplicação.

A abordagem proposta por Malgireddy et al. [MAL12] utiliza as bases de vídeos KTH, HMDB e ChaLearn para validação do processo desenvolvido de classificação e reconhecimento de gestos. O método desenvolvido usa os vídeos de profundidade obtendo como pontos de interesse os pontos que diferem na comparação entre cada quadro com o quadro seguinte. Após identificar esses pontos, os vetores gerados pelos algoritmos de Histograma Orientado a Gradiente (HoG) e Histograma de Fluxo (HoF) são extraídos de cada ponto e agrupados.

Konečný e Hagara [KON13] utilizando os mesmos algoritmos HoG e HoF utilizados por Malgireddy, se diferenciam na forma de junção dos dados resultantes, onde eles aplicam a distância Quadratic-Chi para calcular distância entre histogramas.

Lui [LUI12] apresenta em seu trabalho a possibilidade de utilizar os tensores como objetos de dados multidimensionais possibilitando aplicar análise fatorial múltipla. O

vídeo é então representado como um tensor de terceira ordem e assim é aplicado o algoritmo para decomposição de valores HOSVD para fatorar o tensor. A Figura 2.4 representa os elementos de HOSVD fatorados de um vídeo de gesto de mão, a imagem à esquerda representa a aparência, no centro os movimentos horizontais e na imagem à direita os movimentos verticais.

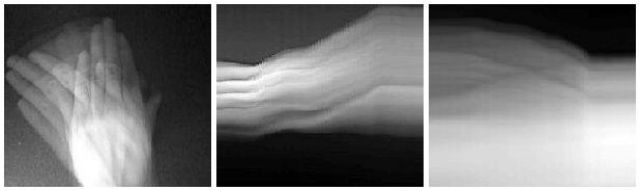


Figura 2.4: Elementos fatorados do HOSVD de um gesto em vídeo [LUI12].

Na abordagem adotada por Wu, Zhu e Shao [WUZ12] o processo utilizado para detecção de características do vídeo é uma variação do algoritmo MHI, que consiste em um processamento para detecção de histórico de movimento da imagem, denominada E-MHI. A imagem de profundidade recebe uma filtragem espacial e um pré-tratamento morfológico para redução de ruídos obtidos pelo sensor. Na sequência, para a separação dos gestos foi utilizada uma abordagem baseada em semelhança da imagem inicial do movimento com a imagem final. Todos os quadros são então projetados como modelos de movimento. “O modelo de movimento foi ampliado para E-MHI como uma fusão do MHI com mais dois elementos: GEI e INV” Wu, Zhu e Shao [WUZ12]. O modelo INV representa o processamento MHI invertido utilizando a parte inicial da imagem e o GEI representa a média entre os resultados de MHI e INV. A Figura 2.5 apresenta um quadro de um vídeo de gesto com a projeção do MHI que destaca os últimos quadros do gesto, o INV que destaca os primeiros quadros e o GEI que codifica a informação de média que é pouco representada por MHI e INV.

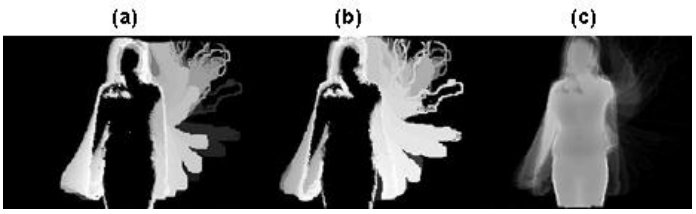


Figura 2.5: Projeção dos métodos (a) MHI, (b) INV e (c) GEI em um vídeo de gesto.

Neste estudo foi explorada a propriedade intrínseca entre os diferentes espectros e incorporada de forma significativa utilizando MSE para junção das imagens de cores com as imagens correspondentes de profundidade.

Uma proposta de extração de características foi proposta por Wan et. al.[WAN13] que funde as imagens do vídeo com escala de cor RGB e as imagens do vídeo de profundidade. A representação denominada 3D EMoSIFT aplica o método 3D MoSIFT desenvolvido por Ming et. al. [MIN12] que é uma extensão do descritor SIFT.

Em cada um dos trabalhos apresentados após a etapa de extração e características, responsável por identificar as informações que diferem cada um dos gestos, é necessário executar o processo de treinamento dos modelos e classificação dos gestos.

2.4.2. Classificação

Com as características extraídas para cada movimento o processo de classificação se faz necessário. Para cada modelo diversos métodos estão disponíveis, alguns que exigem um maior processo de treinamento e carregamento de dados e outros com um modelo de verificação e comparação simplificado e otimizado para o formato de dados gerados na extração.

No trabalho de Malgireddy et. al. [MAL12] após a extração de cada quadro e o agrupamento dos vetores é gerada uma *bag-of-words* e a partir dela um modelo de aprendizagem multinominal temporal gerando uma assinatura probabilística para cada atividade. A Figura 2.6 apresenta o modelo proposto para o fluxo de treinamento separados por gestos dos vídeos.

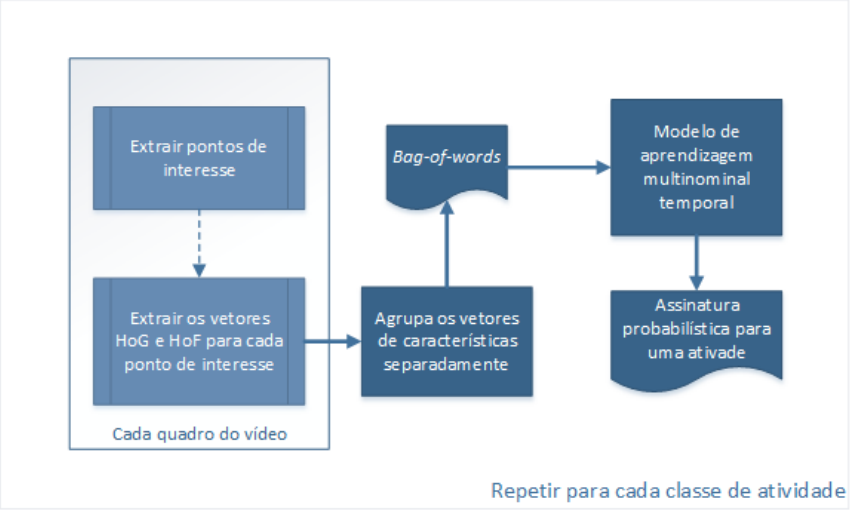


Figura 2.6: Fluxo do ciclo de treinamento.

Malgireddy et al. [MAL12] utilizam um agrupamento *k-Means* para os vetores HoG/HoF extraídos convertendo cada aspecto em um esquema de *bags-of-words*. O processo de classificação é então realizado treinando um modelo baseado em Hidden Markov Model utilizando múltiplos canais e diversos estados, denominado mcHMM. O reconhecimento da atividade é reduzido então a um problema de inferência. Dado um vídeo desconhecido de teste o objetivo é identificar qual atividade foi utilizada para geração do vídeo. Essa inferência é obtida usando o algoritmo Viterbi. Para a identificação de gestos foi criada uma rede de modelos conectados para cada gesto aprendido e utilizado cálculo de distância para teste de desempenho.

Para o modelo usando HOSVD apresentado por Lui [LUI12] a classificação dos gestos é obtida usando um gesto dividido em três modelos de sub regressão separando em três variedades Grassmann onde a regressão é executada. O modelo de regressão dos mínimos quadrados é um método não linear em variedade para o reconhecimento de gestos. A distância entre a regressão e o gesto é obtida aplicando a distância cordal.

Segundo Lui [LUI12] a regressão linear é uma técnica fundamental na análise estatística e apresenta um desempenho considerável em modelos que possuem poucos exemplos de treinamento. Com o uso de regressão dos mínimos quadrados é enfatizado o aspecto subjacente da geometria, coletando padrões apresentados nos gestos em um espaço não Euclidiano.

O trabalho desenvolvido por Wu, Zhu e Shao [WUZ12] apresenta uma abordagem para o reconhecimento de gestos utilizando *One-shot Learning* em vídeos capturados com sensor Kinect™. Os vídeos utilizados no treinamento foram obtidos da base de dados disponível para o evento *ChaLearn Gesture Challenge*. O algoritmo MSE é utilizado para fundir os dados obtidos do E-MHI, esse espectro é então comparado usando Coeficiente de Correlação usando treinamento dos vídeos sem identificação.

Konečný e Hagara [KON13] apresentam dois métodos para classificação usando as características HoG e HoF combinadas comparando com uma variação do algoritmo de Dynamic Time Warping. O primeiro método denominado SM onde cada quadro é representado por um nó e é incluído um nó para representar o quadro onde o ator está parado. Ao processar um novo vídeo para comparação os nós de todos os vídeos são comparados com o do exemplar de validação usando distância Quadratic-Chi. Para Konečný e Hagara [KON13] as comparações são representadas então como uma matrix de tamanho $N \times (f-1)$ onde N é o número total de nós dos modelos e f é o número total de quadros do vídeo para validação. Usando o algoritmo Viterbi é possível

determinar o menor caminho na matriz identificando a possível classe da qual o vídeo pertence.

Wan et. al. [WAN13] utilizam um modelo diferenciado para comparação de *bag-of-feature*, que tradicionalmente usam quantização de vetores para mapear cada característica em um código, com um método de espaçamento nomeado SOMP aplicado para que cada característica pode ser apresentada como uma combinação de pequenos códigos.

2.4.3. Considerações finais

Baseado no desafio ChaLearn[1] de reconhecimento de gestos, que consiste na detecção de gestos utilizando um exemplar de treinamento, Guyon et. al. [GUY12] apresentam como foi organizado o desafio, a base de vídeos disponível, o método de validação dos trabalhos participantes e os resultados alcançados pelos primeiros colocados da primeira etapa do desafio.

Guyon et. al. [GUY12], diz que: "Nós retratamos um usuário em frente a uma câmera fixa, interagindo com o computador desempenhando gestos para um jogo, controle remoto de aplicação ou robôs, ou aprendizado de gestos para um software educacional". Como protocolo de avaliação para a primeira etapa do ChaLearn foi calculado a distância de *Levenshtein* de acordo com o resultado apresentado por um participante comparado ao resultado verdade.

"A avaliação de desempenho final da primeira etapa do desafio de reconhecimento de gestos é aproximadamente 10% de erro, ainda distante do desempenho humano, que é abaixo de 2% de erro". Guyon et. al. [GUY12]. Diversas técnicas foram implementadas pelas equipes participantes do projeto utilizando as informações de RGB e profundidade do sensor, na análise final as técnicas de melhor resultado utilizaram HMM para processar as características e apenas a informação de profundidade ou cor.

Os resultados apresentados variam em desempenho e assertividade, cada método contém sua própria forma de tratar os processos para o reconhecimento de gestos. A Tabela 2.6 apresenta de forma resumida dos trabalhos estudados que participaram do desafio ChaLearn para reconhecimento de gestos.

[1] Desafio de reconhecimento de gestos utilizando um exemplar de treinamento para cada gesto. ChaLearn é uma organização isenta de impostos sob a seção 501 (c) (3) do Código dos Estados Unidos da Américc.

Tabela 2.6: Comparação de resultados dos trabalhos participantes do ChaLearn

Referência	Base	Extrator de características	Classificador	Taxa de acerto (%)
Malgireddy et. al. [MAL12]	ChaLearn [CHA11]	HOG / HOF	Multi-channel HMM	81,54
Lui [LUI12]	ChaLearn [CHA11]	Least Squares Regression	HOSVD	76,97
Wu, Zhu e Shao [WUZ12]	ChaLearn [CHA11]	Extended-MHI e MSE	Correlation Coefficient Comparison	74,36
Konečný e Hagara [KON13]	ChaLearn [CHA11]	HOG / HOF	Quadratic-Chi distance com Single Model	89,02
			Quadratic-Chi distance com Multiple Model	81,53
Wan et. al. [WAN13]	ChaLearn [CHA11]	3D EMoSIFT	SOMP	87,41

No estudo proposto por Guyon et. al. [GUY12] foi executada a comparação do desempenho dos trabalhos que atingiram a melhor colocação no *ChaLearn Gesture Challenge*, dentre os oito primeiros colocados apenas um dos competidores utilizaram todos os recursos disponíveis para o Kinect™ na extração de características, tratando os dados do espaço de cor e o valor de distância capturada pelo infravermelho. Segundo Guyon et. al. [GUY12] a avaliação de desempenho da primeira etapa do desafio atingiu cerca de 10% de erro, distante se comparado aos erros humanos na faixa de 2%.

Conforme apresentado por Mitra e Acharya [MIT07], a implementação do HMM e do FSM em conjunto são um potencial estudo para melhoria do processo de reconhecimento de gestos.

No trabalho apresentado por Malgireddy et. al. [MAL12] o processo de identificação dos pontos de interesse aplicado apenas para os pontos que são modificados entre dois frames ajuda a reduzir a quantidade de pontos e agilizar o processamento seguinte de identificação de características e com a criação da "*bag of words*" é possível reduzir a quantidade de dados a serem trabalhados no processo de classificação.

Entre as propostas apresentadas em diversos trabalhos para extração de características é notável a importância desta etapa nos resultados. Uma ampla variedade de métodos de classificação pode ser utilizada, mas a dependência de resultados é diretamente relacionada a diversificação dos dados das características específicas.

Conforme mostra a literatura o reconhecimento de gestos utilizando poucos exemplares de treinamento podem atingir altas taxas de acerto com poucos recursos computacionais, permitindo assim o uso de equipamentos simples e cotidianos para o desenvolvimento de equipamentos de suporte a comunicação incluindo sistemas para a acessibilidade e ensino.

Capítulo 3

Método Proposto

O método proposto consiste na classificação de vídeos de gestos de palavras na linguagem LIBRAS. Como apresentado na Figura 3.1 a base de vídeos é separada em modelos e validação, o processo é então dividido em quatro etapas. A primeira etapa consiste na geração dos *codebooks* de características e histogramas utilizados para o processamento dos vídeos. Na segunda etapa é executado o processamento dos vídeos de exemplares de gestos de palavras e validação com a extração de características de cada quadro, o cálculo dos histogramas de cada quadro, o cálculo do histograma do vídeo e a geração da *string* que representa o vídeo. Após o processamento é executada a terceira etapa de classificação dos vídeos com base na comparação das distâncias de caracteres e por último a quarta etapa consiste na análise dos resultados das comparações.

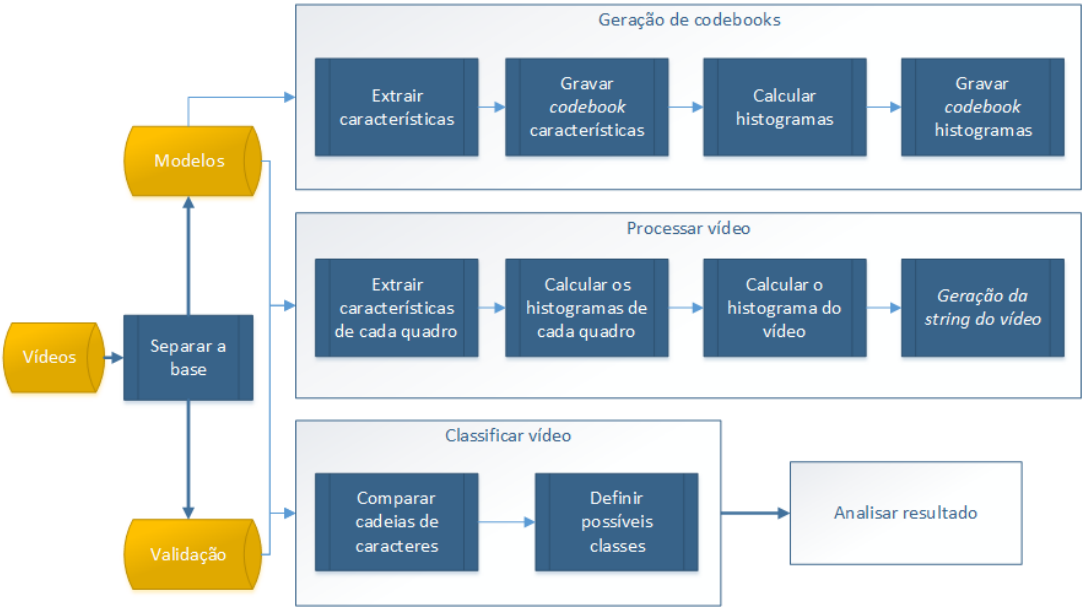


Figura 3.1: Método proposto

Os códigos dos vídeos de referências são gerados como uma identificação do vídeo. Os códigos dos vídeos de validação são então comparados a cada um dos códigos de exemplos e então são classificados pelo cálculo de distância de strings do mais próximo ao mais distante.

A base de vídeos de palavras em LIBRAS [MEN14] utilizada neste trabalho foi gravada com o Microsoft Kinect™ em conjunto com a biblioteca OpenNI e será aplicado um pré-processamento para divisão do arquivo OpenNI em dois arquivos, um contendo o espaço de cor RGB e outro com a informação de profundidade em escala de cinza.

3.1. Base de Vídeos

A base [MEN14] contém 9 palavras de gestos da linguagem LIBRAS e consistem em verbos no infinitivo formando um total de 181 vídeos detalhados na Tabela 3.1.

Tabela 3.1: Total de vídeos e atores por palavra

Índice	Palavra	Quantidade	Atores diferentes
0	Entregar	22	12
1	Pegar	20	11
2	Abrir	20	11
3	Olhar	18	12
4	Empurrar	20	14
5	Fechar	19	10
6	Falar	23	14
7	Puxar	19	8
8	Trabalhar	20	16

As palavras selecionadas foram escolhidas por serem comuns em jogos educacionais utilizados pela comunidade de deficientes auditivos. A Figura 3.2 apresenta uma imagem do jogo *The Secret of Monkey Island,* da empresa LucasArts Entertainment Company, que serviu como orientação na escolha do conjunto de palavras para a base de vídeos [MEN14]. Dentre as nove palavras o vocabulário proposto contém: “entregar”, “abrir”, “fechar”, “pegar”, “olhar”, “falar”, “empurrar”, “puxar”.

Figura 3.2: Cena do jogo *The Secrets of Monkey Island*.

A gravação foi realizada em ambiente fechado e com iluminação artificial ambiente em cenário não preparado. A base de vídeos foi gravada por 23 atores variando sexo, idade, estatura e cor de pele. Cada ator gravou de um a três vídeos.

Os equipamentos utilizados para a gravação da base foram:

- Notebook com software para captura;
- Monitor para a visualização dos gestos realizados;
- Sensor Microsoft Kinect™.

Para a gravação foram estabelecidos alguns padrões de formato e características para o vídeo resultante. O formato estabelecido para a gravação desta base foi:

- Tamanho de 640 x 480 pixels com proporção 4:3;
- Captura do vídeo em RGB entre 15 à 30 quadros por segundo;
- Captura do vídeo de profundidade entre 15 à 30 quadros por segundo;
- Captura da sequência de posição do esqueleto para cada quadro do vídeo;
- Formato de arquivo utilizado para o vídeo é o ONI e para o esqueleto é um formato próprio denominado SKL.
- Nomenclatura: V_ATOR_TOMADA_PALAVRA.oni

A Figura 3.3 apresenta um quadro do resultado da gravação no espaço de cores RGB e o seu equivalente em profundidade usando escala de cinza.

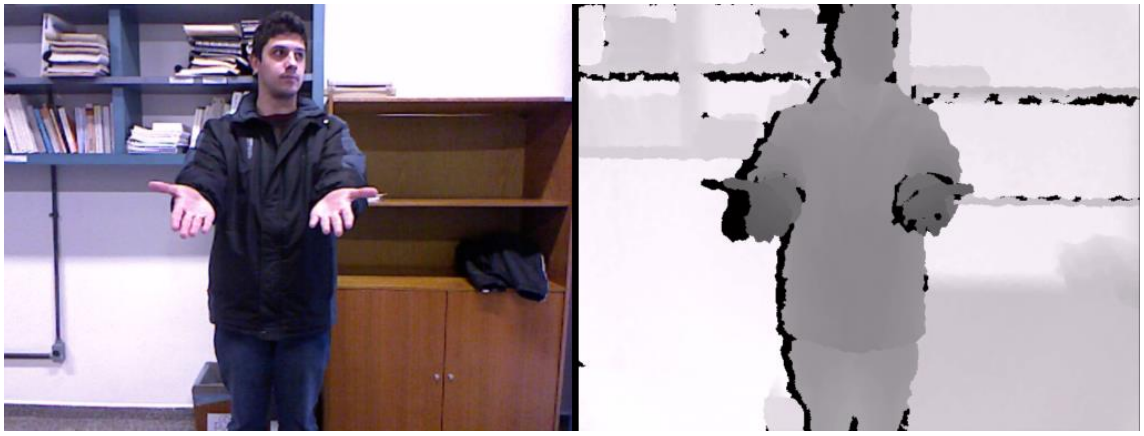


Figura 3.3: Exemplo da base de vídeos.

As duas imagens são armazenadas dentro do arquivo de vídeo ONI utilizando o espaço de cores em 24 bits por pixel e a informação de profundidade em 16 bits por pixel. O arquivo de esqueleto armazena em um formato de texto cada ponto do esqueleto e sua posição, por quadro do vídeo.

3.1.1. Conversão da base

A base de vídeos [MEN14] está armazenada em formato ONI dos vídeos capturados a partir do Microsoft Kinect™ e requerem o uso da biblioteca OpenNI para a extração das imagens. A Figura 3.4 representa o pré-processamento executado para conversão do formato ONI para o formato AVI.

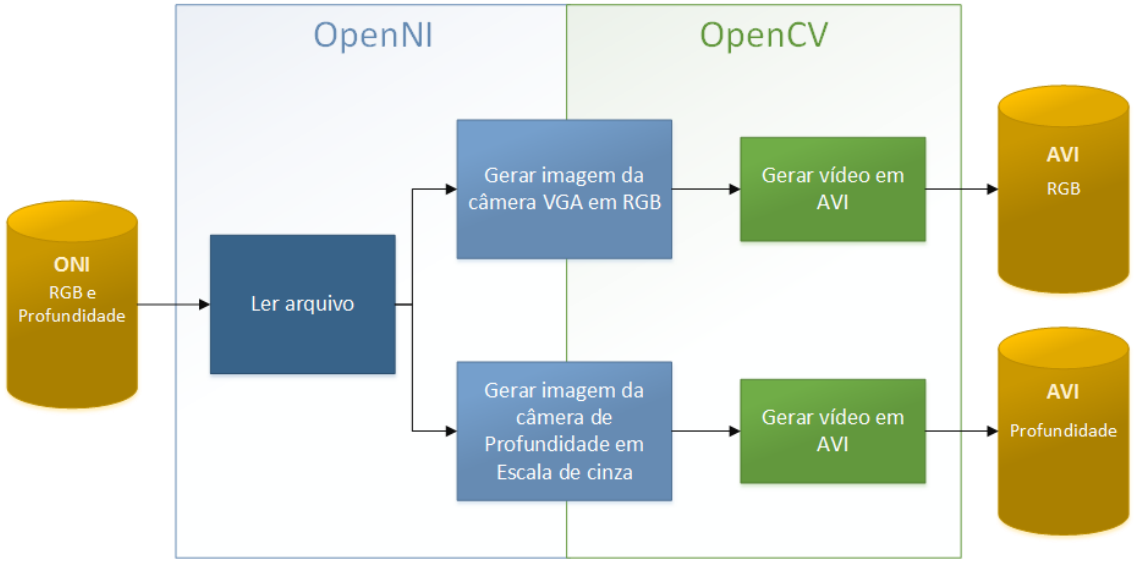


Figura 3.4 - Pré-processamento da base de dados

Os vídeos capturados na base são convertidos para o formato AVI, um formato de container de áudio e vídeo definido pela Microsoft®. Este formato foi escolhido devido a sua ampla aplicação em diversos aparelhos e aplicativos.

A conversão inclui a diminuição de quadros por segundo, removendo assim os quadros que apresentarem problemas de codificação como a parte inferior visível na Figura 3.5. Esses erros são gerados pelo equipamento de captura alterando a codificação da imagem em RGB.

Figura 3.5: Exemplo de quadro com problema de codificação no espaço de cor.

A base de vídeo é reprocessada e os vídeos são convertidos considerando as seguintes características:

- Sem alteração do tamanho de vídeo, mantendo tamanho e proporção.
- AVI com codec RGB sem compactação.
- Taxa padrão de 10 quadros por segundo, removendo quadros com codificação alteradas.
- Conversão do vídeo de profundidade para o formato RGB em escala de cinza.
- Nomenclatura:
    - RGB: V_ATOR_TOMADA_PALAVRA_C.avi;

- Profundidade: V_ATOR_TOMADA_PALAVRA_D.avi;

A leitura dos vídeos é feita utilizando o OpenNI 2.0 SDK em conjunto com a biblioteca OpenCV referenciados no capítulo 2.

3.1.2. Separação da base

Os vídeos então são separados entre base de exemplares para a criação dos *codebooks* e cadeia de caracteres de exemplo e validação para o processo de classificação. Para a separação da base foi utilizada uma adaptação do método de validação cruzada *Leave-p-out*, este método aplica a divisão do número total de dados em dois conjuntos mutuamente exclusivos onde *p* representa a quantidade de dados utilizada para validação.

Neste caso a aplicação do método é invertida, utilizando *p* para a quantidade de exemplares na geração dos *codebooks* e códigos de vídeos de exemplo para classificação. A base de vídeos é então separada em exemplares e classificação utilizando *p* com os valores: 1, 14, 15, 16 e 18 exemplares de vídeos por execução. Os vídeos são selecionados de maneira aleatória, mantendo a proporção de separação por palavra.

O método proposto é aplicado no vídeo de profundidade gerando a cadeia de caracteres para cada vídeo que serão usados para a comparação na etapa de classificação.

A primeira etapa do método é a extração das características dos vídeos, para isso são descritas as características de cada quadro do vídeo.

3.2. Processamento dos Quadros de Cada Vídeo

O processamento dos vídeos é separado em três etapas que consistem no processamento de quadros identificando as características, na geração do *codebook* de características e no cálculo de histogramas para cada quadro.

3.2.1. Descrição das características

O Kinect™ possui a característica de registrar dois tipos de informação: a imagem capturada pela câmera em formato RGB colorido e a informação de profundidade do vídeo capturado pela câmera de infravermelho e interpretado como imagem em escala de cinza. O descritor SURF utiliza informações de borda, intersecção de pixel, intensidade e outras características independentes da cor da imagem, devido a esse aspecto do descritor a imagem de profundidade oferece características diferenciadas para o descritor, utilizando a informação de profundidade registrada como o tom do pixel na imagem.

O processo de descrição das características é iniciado com um pré-processamento da imagem para cálculo do centro de massa e exclusão do cenário. A Figura 3.6 exemplifica o primeiro quadro do vídeo V_00_1_00_D e o histograma correspondente dos canais RGB. Utilizando a imagem do primeiro quadro é gerada uma máscara, selecionando o personagem do vídeo e excluindo o fundo, baseado na variação de cores.

Figura 3.6: Primeiro quadro do vídeo V_00_1_00_D usado para máscara.

A Figura 3.7 representa a imagem da máscara gerada a partir da Figura 3.6 utilizando apenas os valores entre as variações de tons de cinza em 60 até 200 representado em preto no histograma da figura.

Para determinar o centro de massa todos os pontos brancos da imagem de máscara são somados e divididos pela quantidade de pontos em branco, retornando a posição das coordenadas *x* e *y* da imagem.

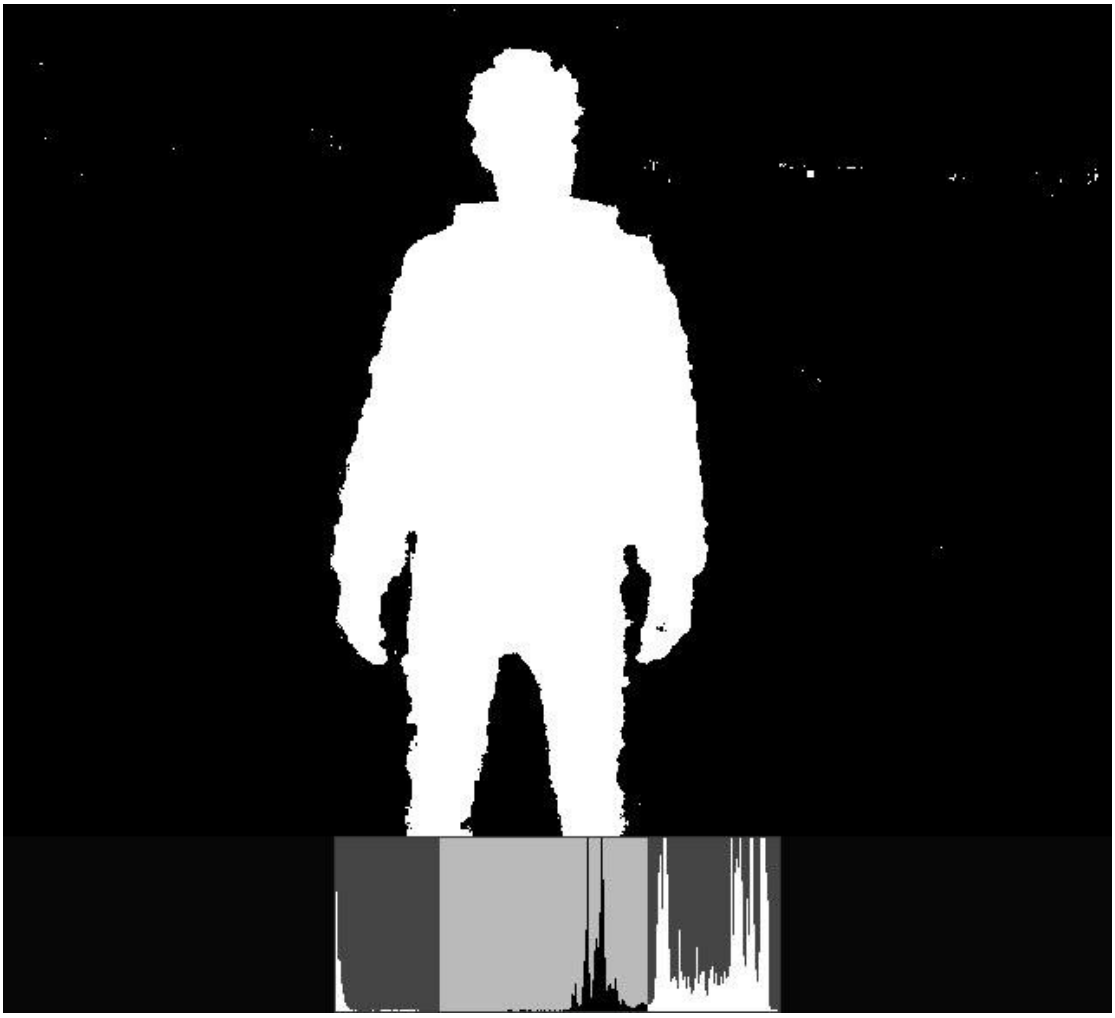


Figura 3.7: Máscara e o histograma original com a variação selecionada.

O Algoritmo 3.1 executa o cálculo do centro de massa recebendo a imagem de máscara *I* como parâmetro, calcula a largura e altura da imagem e inicializa uma variável para acumular o total das coordenadas *x* e *y* e um contador de pontos. Para cada posição *x* e *y* da imagem é verificado se o ponto específico *I(x, y)* são da cor branca (com valor 255). Se o ponto é branco são somadas as coordenadas *x* e *y* do ponto em seus respectivos totais e o contador é incrementado. No final do algoritmo é retornado o ponto de duas dimensões com a coordenada *x* recebendo o valor total das coordenadas de x somadas dividido pelo contador de pontos e o valor total das coordenadas de y somadas dividido pelo contador de pontos.

Algoritmo 3.1 Cálculo do centro de massa

```
{recebe imagem I}
n ← Largura(I)
m ← Altura(I)
totalX ← 0
totalY ← 0
contador ← 0

para y ← 0 até n faça
    para x ← 0 até m faça
        se (I(x, y) == 255) então
            totalX ← totalX + x
            totalY ← totalY + y
            contador ← contador + 1
        fim se
    fim para
fim para

{retorna Ponto(totalX / contador, totalY / contador)}
```

A Figura 3.8 demonstra o centro de massa da imagem marcado em azul. Os pontos então são armazenados como x_0 e y_0 para o cálculo de cada característica localizada na imagem.

Figura 3.8: Representação do ponto do centro de massa.

Após a definição do centro de massa cada quadro do vídeo é processado com uma máscara para remoção dos valores que estão atrás da mesma linha de profundidade que o centro de massa e a frente da escala utilizada na máscara com o tom de cinza 200.

Com a máscara definida as características SURF e SIFT da imagem são extraídas. A Figura 3.9 apresenta os quadros do vídeo V_00_1_00_D.avi com a máscara utilizada para a imagem e os pontos de interesse identificados pelo algoritmo SURF.

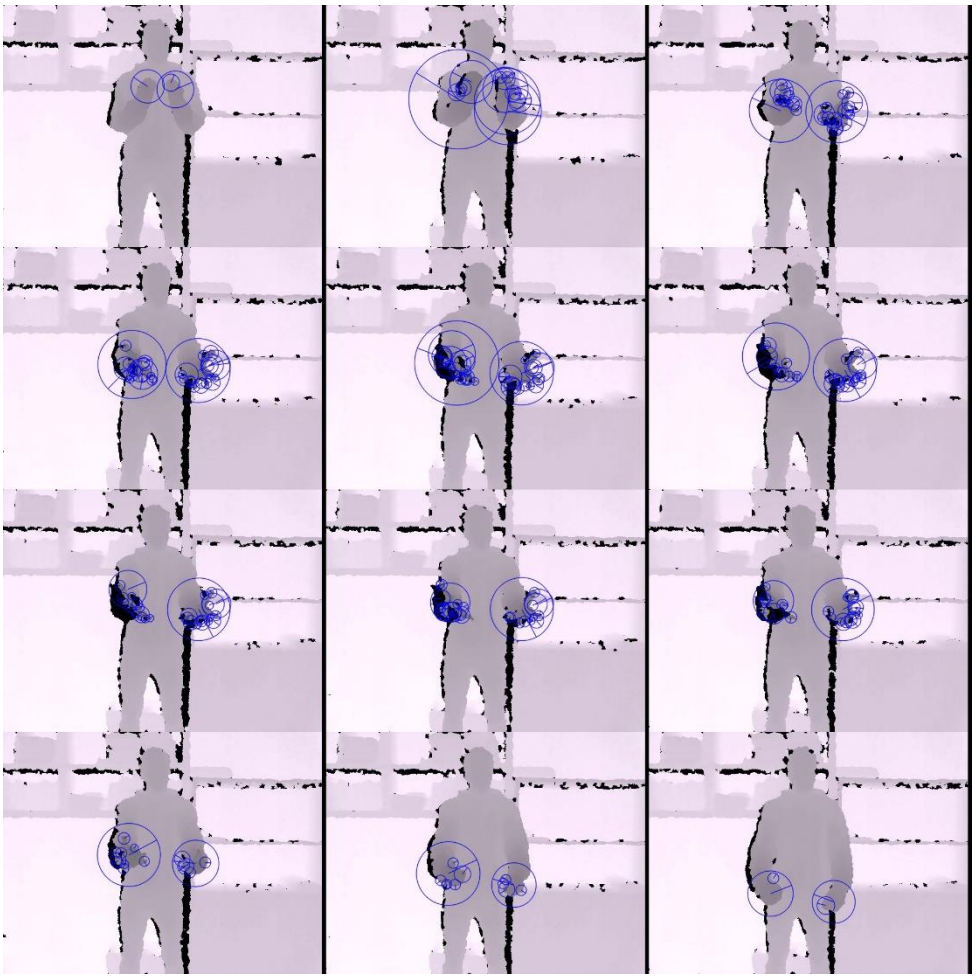


Figura 3.9: Quadros de vídeo com pontos de interesse detectados pelo SURF.

A identificação dos pontos de interesse é feita usando o algoritmo SURF [BAY08]. A configuração do algoritmo SURF foi utilizada com as propriedades padrão conforme a Tabela 3.2. Em cada quadro do vídeo então são identificados os pontos de interesse que armazenam informações como: posição nas coordenadas *x* e *y*, tamanho do vetor, ângulo do ponto, o indicador de qual oitava está o ponto e a sensibilidade do ponto.

Tabela 3.2: Parâmetros de configuração do algoritmo SURF.

Parâmetro	Valor utilizado	Descrição do parâmetro
Hessian threshold	500	Limiar para a matriz Hessian usada para localizar os pontos-chave.
Octaves	4	Número de pirâmides de oitavas utilizada para detecção dos pontos de interesse.
Octave layers	2	Número de camadas nas pirâmides.
Extended	Falso	Indica se o algoritmo utilizará descritores de 128 ou 64 elementos.
Upright	Falso	Indica se o algoritmo os pontos sem orientação ou com o ângulo dos pontos de interesse.

Os pontos de interesse podem ser vistos na Figura 3.10 e armazenam informação de coordenada *x*, coordenada *y*, tamanho e ângulo de cada ponto. A partir dos pontos de interesse o algoritmo descreve cada vetor de características para os pontos utilizando um descritor de tamanho de 64 elementos por ponto.

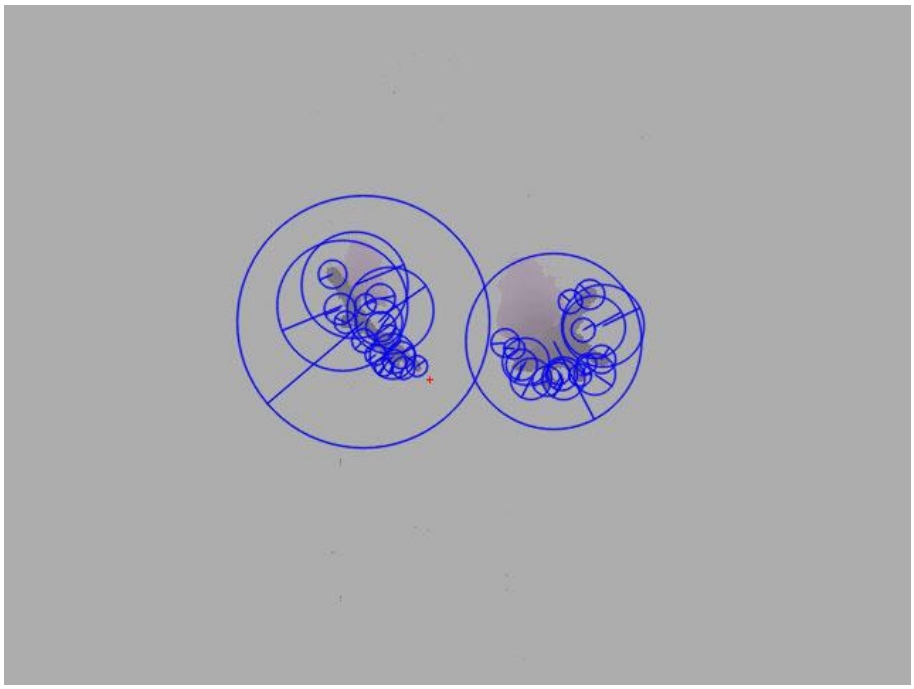


Figura 3.10: Pontos de interesse em azul e destaque do centro de massa em vermelho.

Em conjunto às características SURF foram extraídas também características SIFT para cada quadro utilizando o mesmo processo de máscara. A Figura 3.11 apresenta os quadros do vídeo V_00_1_00_D.avi com a máscara utilizada para a imagem e os pontos de interesse identificados pelo algoritmo SIFT.

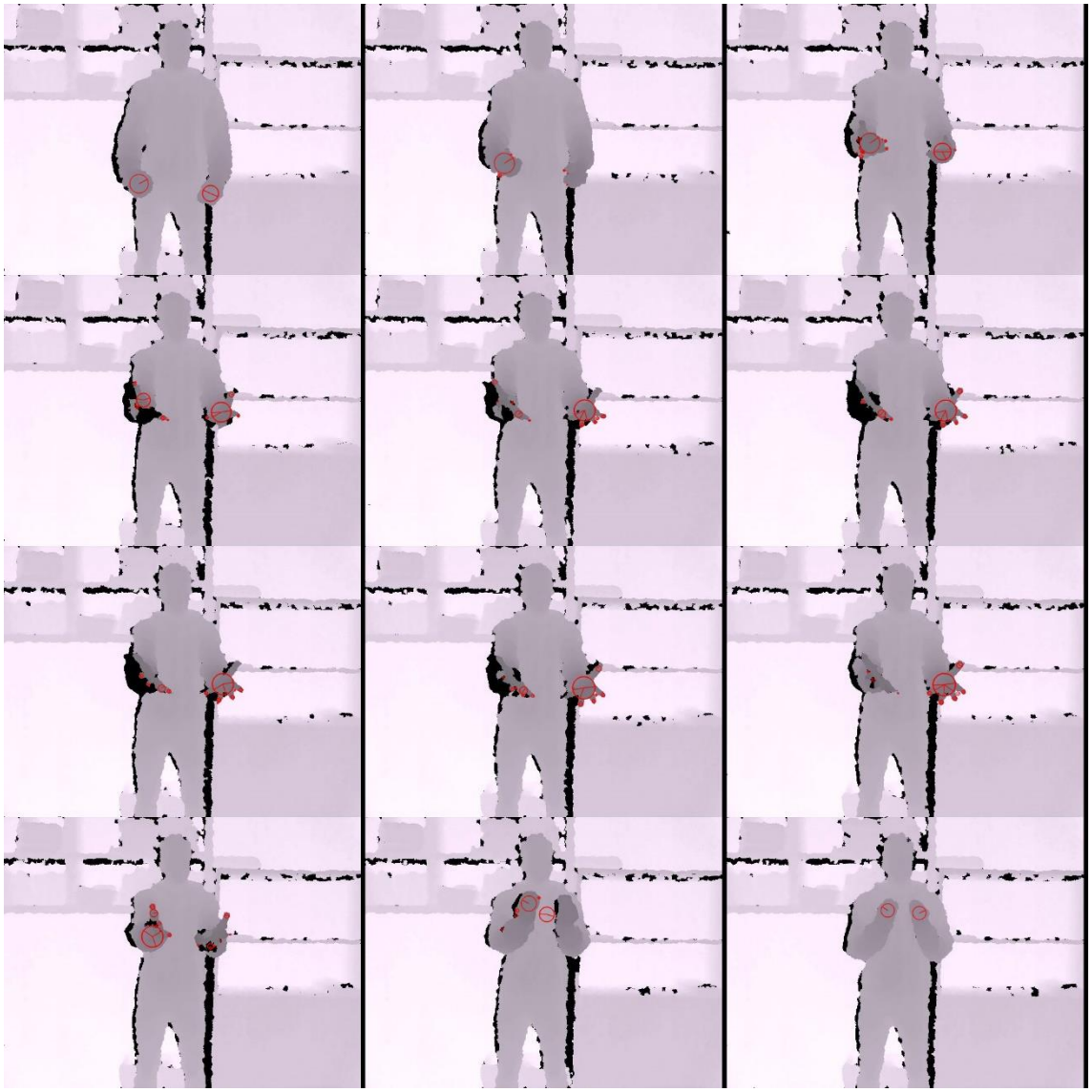


Figura 3.11: Quadros de vídeo com pontos de interesse detectados pelo SIFT

Outra identificação dos pontos de interesse é feita usando o algoritmo SIFT [LOW99]. A configuração do algoritmo SIFT foi utilizada com as propriedades padrão conforme a Tabela 3.2. Em cada quadro do vídeo então são identificados os pontos de interesse que armazenam informações como: posição nas coordenadas *x* e *y*, tamanho do vetor, ângulo do ponto, o indicador de qual oitava está o ponto e a sensibilidade do ponto.

Tabela 3.3: Parâmetros de configuração do algoritmo SIFT.

Parâmetro	Valor utilizado	Descrição do parâmetro
Number of features	20	Número de melhores características definidas por pontuação do algoritmo.
Octave layers	3	Número de camadas em cada oitava.
Threshold Contrast	0.04	Limiar de contraste para reduzir as características menos expressivas em regiões semi-uniformes.
Threshold Edge	10.0	Limiar utilizado para filtrar as características com informação de borda.
Sigma	1.6	Valor do Gaussian aplicado à imagem de entrada.

Os pontos de interesse podem ser vistos na Figura 3.11, no exemplo dos pontos detectados pelo SURF, armazenam informação de coordenada *x*, coordenada *y*, tamanho e ângulo de cada ponto. A partir dos pontos de interesse o algoritmo descreve cada vetor de características para os pontos utilizando um descritor de tamanho de 64 elementos por ponto.

As características são baseadas nos valores normalizados dos pontos de interesse e considerando o centro de massa do ator no primeiro quadro do vídeo e o descritor de cada ponto localizado. Os valores de características são:

- Ponto X de origem: *x*;
- Ponto Y de origem: *y*;
- Coordenada X: $|x - x_0|$ / largura do vídeo;
- Coordenada Y: $|y - y_0|$ / altura do vídeo;
- Coordenada Z: z / 255.0f;
- Descritor: descritor de 64 elementos calculado pelo SURF para o ponto de interesse ou o descritor de 128 elementos calculado pelo SIFT;

Os valores de descritores de cada quadro do vídeo serão armazenados para a geração de um *codebook* de descritores por quadro.

3.2.2. Geração do *codebook* das características

Os descritores detectados pelo algoritmo SURF e SIFT em todos quadros dos vídeos de exemplo são carregados e separados para a geração do *codebook* de características, um *codebook* para cada algoritmo aplicado. Para este processo o método de agrupamento *k-Means* foi selecionado devido a sua velocidade de processamento, facilidade de implementação e robustez na utilização com tamanhos de classes variáveis. O método de agrupamento *k-Means* utiliza um número de classes *K*, a ser determinado em experimento, definindo o número de entradas do *codebook*. Os valores de cada

descritor de treinamento serão armazenados como base para o processamento do agrupamento *k-Means*.

Dado um valor *k* inicial e um vetor dos descritores dois passos são executados nesse processo. O primeiro passo consiste na atribuição do vetor de descritores a um centroide, calculado a partir do valor médio verificado pela distância Euclidiana utilizando Algoritmo 3.2. O cálculo de distância é obtido da subtração de cada propriedade de duas características C_0 e C_1 na exponenciação com expoente 2. O valor de distância é retornado da raiz quadrada do valor calculado.

Algoritmo 3.2 Método para cálculo de distância entre duas características

```
{recebe outra classe de característica C e utiliza os valores da própria classe C0}

valor ← 0
n ← Tamanho(C0.descritor)

valor ← valor + Potencia(C0.x - C1.x, 2)
valor ← valor + Potencia(C0.y - C1.y, 2)
valor ← valor + Potencia(C0.z - C1.z, 2)

para i ← 0 até n faça
    valor ← valor + Potencia(C0.descritor(0) - C1.descritor(0), 2)
fim para

{retorna RaizQuadrada(valor)}
```

A cada característica é atribuído uma identificação de qual centroide ele pertence, considerando o centroide mais próximo. No segundo passo são calculados os novos centroides baseado na média de todas as características atribuídas. Então é a diferença entre a distância Euclidiana do centroide anterior e o novo. Verifica-se a se diferença é maior que uma taxa de erro de 0,003. Caso a diferença seja menor que a taxa de erro o processo é executado novamente, caso contrário o processo é finalizado com a identificação de centroide por descritor. Cada um dos centroides finais é definido como valores de entrada do *codebook*.

Os *codebooks* para SURF e SIFT são então armazenados em arquivo próprio de extensão CBF em formato de texto com cada valor separado por linha para utilização no processo de validação.

3.2.3. Cálculo do histograma dos quadros

Com o *codebook* de características, cada quadro dos vídeos de treinamento, com suas características, é processado para geração dos histogramas do quadro. Para cada quadro são criados quatro histogramas, dois histogramas para cada algoritmo com um para cada lado do vídeo, separado a partir do ponto do centro de massa. Na Figura 3.12 a linha vermelha representa os lados da imagem e os pontos de interesse, detectado pelo algoritmo SURF, utilizados para calcular as características do quadro são destacados em azul.

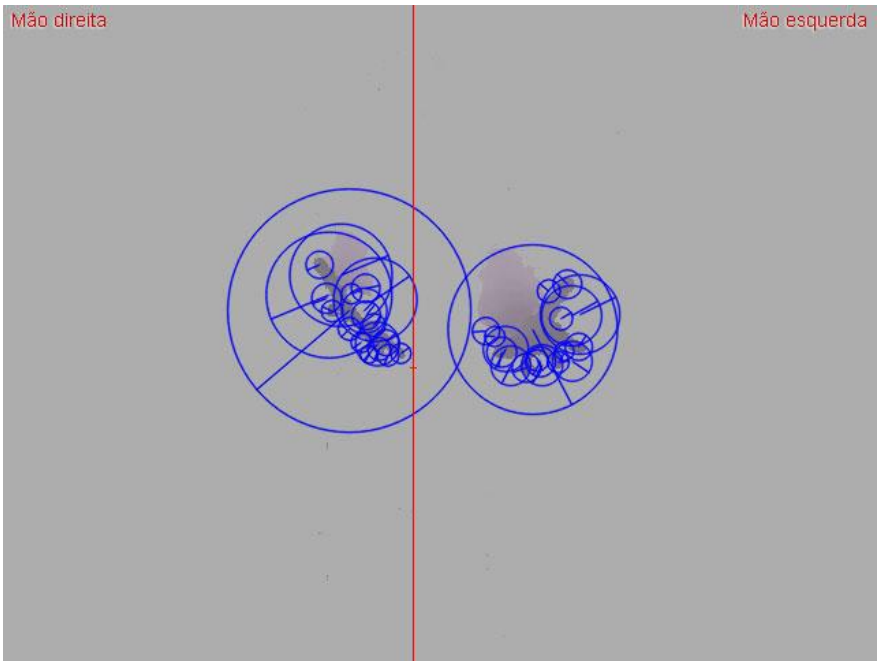


Figura 3.12: Exemplo dos lados e pontos de interesse no quadro.

Cada característica do quadro é selecionada e o ponto calculado utilizando a posição do ponto para definir se pertence ao histograma que representa o lado esquerdo ou direito do quadro do vídeo. A distância da característica é calculada com cada item do *codebook*, o índice do *codebook* que retornou a menor distância é então adicionado ao histograma correspondente.

No vídeo V_00_1_00_D.avi a primeira característica do quadro possui os valores:

- Origem X: 272;
- Origem Y: 215;
- X: 0,425;
- Y: 0,447916;
- Descritores: vetor com as características.

Com o ponto do centro de massa na posição X com o valor 300 e a posição Y com o valor 265, é calculada a qual lado pertence o ponto da característica. Na Figura 3.13 a característica selecionada é destacada em verde dentro do espaço da mão direita.

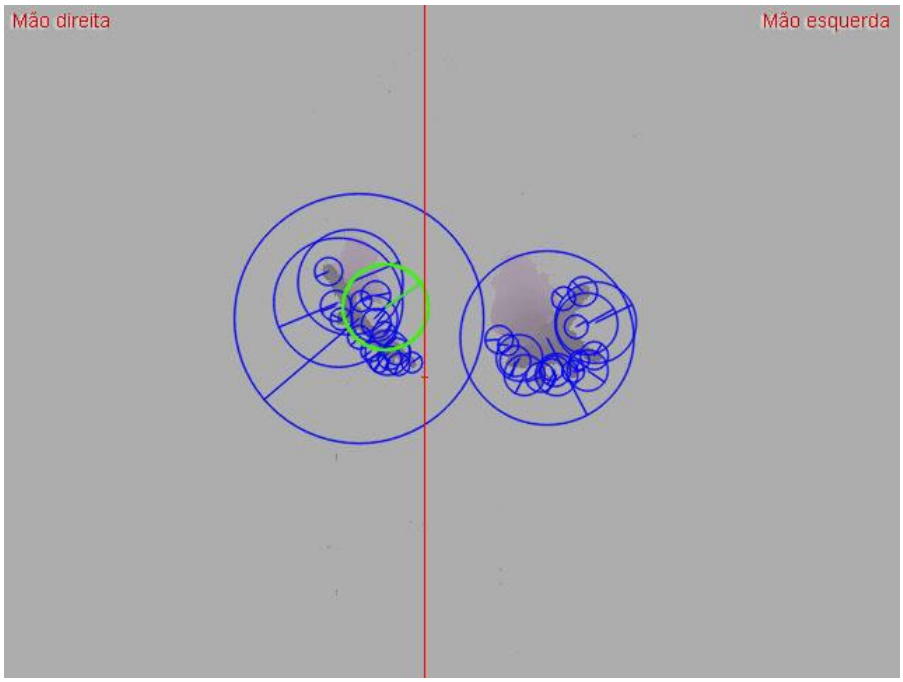


Figura 3.13: Característica destacada no lado da mão direita de um quadro de vídeo.

Calculando a distância de cada característica dos *codebooks* com a característica selecionada obtemos a lista de distâncias sendo o menor valor 0,104 identificado no índice 30. Com esse número a entrada 30 do histograma referente ao lado da mão direita é incrementada.

Os histogramas de quadros dos vídeos de treinamento serão utilizados para a geração de um *codebook* de histogramas utilizado para calcular apenas um histograma resultante para cada vídeo.

3.3. Criação da Cadeia de Caracteres do Vídeo

3.3.1. Geração do *codebook* dos histogramas

O próximo passo utiliza os histogramas gerados para cada quadro dos vídeos para criar novos *codebooks*. O *codebook* de histogramas é usado para gerar dois histogramas por vídeo mantendo as informações de cada característica extraída. Os histogramas de cada quadro dos vídeos de treinamento são carregados e utilizando agrupamento *k-Means*

são identificados *L* classes de histogramas, sendo *L* um valor a ser definido na fase de treinamento. O agrupamento *k-Means* é calculado utilizando o Algoritmo 3.3 no cálculo de distâncias Euclidiana entre os histogramas, retornando a diferença entre cada elemento elevadas ao quadrado e somadas e calculando a raiz quadrada do total somado.

Algoritmo 3.3 Cálculo de distância entre dois histogramas.

```
{recebe outra classe de histograma  H0 e utiliza os valores da própria classe H1}
n ← Tamanho(H0)
valor ← 0

para i ← 0 até n faça
    valor ← valor + Potencia(H0(i) – H1(i))
fim para
{retorna RaizQuadrada(valor)}
```

O *codebook* é então armazenado em arquivo próprio de extensão CBH em formato de texto com cada valor de histograma, para cada um dos lados, separado por linha para utilização no processo de codificação dos vídeos.

3.3.2. Cálculo do histograma dos vídeos

A cada vídeo, os histogramas dos quadros serão comparados ao *codebook* usando o cálculo de distância apresentado no Algoritmo 3.4.

O algoritmo apresenta uma matriz *V* (*n*, *o*) que representa os histogramas por quadro de um vídeo, sendo *n* a quantidade de quadros e *o* representa o tamanho de cada histograma. A matriz do *codebook* é representada por *C*(*m, o)*, sendo *m* a quantidade de entradas do *codebook.* A cadeia de caracteres correspondente é obtida calculando o elemento mais próximo de uma posição da matriz do *codebook* do histograma de cada quadro, obtendo assim o vetor *S*(*m)* de tamanho *m*. O menor valor do vetor *S* representa o elemento mais próximo da matriz. A chave do menor valor de *S* é obtida por uma função *inv*() e atribuída a uma variável *c*. Convertendo o valor de *c* para um caractere ASCII com a função *toASCII()* é obtido o caractere correspondente ao quadro, esse valor é adicionando a uma *string* da assinatura do vídeo representada pela variável *SV*.

Algoritmo 3.4 Calculo de distância de histograma do quadro do vídeo

```
{inicialização}
para i ← 0 até n faça
    para j ← 0 até m faça
        para k ← 0 até o faça
            S(k, j) ← | C(j,k) - V(i, k) |
        fim para
    fim para
    c ← inv(S)
    SV ← toASCII(c)
fim para
```

A classificação é então realizada usando as cadeias de caracteres de cada vídeo de validação comparada às cadeias de caracteres dos vídeos de treinamento.

3.4. Classificação Usando a Distância de *Strings*

A classificação desenvolvida para esse método utiliza o algoritmo de Levenshtein alterando o valor de custo com base na distância dos caracteres presentes nas cadeias. Essa comparação é realizada considerando que o número de histogramas próximos que podem ser confundidos na comparação dos *codebooks* que também é influenciado pela quantidade de quadros disponíveis no vídeo utilizado. O Algoritmo 3.5 apresenta a distância de Levenshtein com a variação de custo destacada. O cálculo é feito a partir do número equivalente à distância entre os caracteres da tabela ASCII identificados através do método *inteiro* e os valores são subtraídos e o valor absoluto é considerado como custo.

Algoritmo 3.5 Cálculo de distância de Levenshtein com custo variável

```
{recebe as strings de parâmetros S1 e S2}
n ← Tamanho(S1)
m ← Tamanho(S2)
D ← Matriz(n, m)
a ← 0
b ← 0

para i ← 0 até n faça
    D(i, 0) ← i
para j ← 0 até m faça
    D(0, j) ← j

para i ← 1 até n faça
    para j ← 1 até m faça
        se (S1(i – 1) == S2(j – 1)) então
```

```
        custo = 0;
    senão
        a ← inteiro(S1(i – 1))
        b ← inteiro(S2(j – 1))
        custo = | a – b |
    fim se

    D(i, j) ← min(D(i – 1, j) + 1, D(i, j – 1) + 1, D(i – 1, j – 1) + custo)
  fim para
fim para

{retorna D(n, m)}
```

Para cada vídeo de validação v são executadas as comparações com a base de *string* de treinamento T de tamanho p e o método de classificação retorna um vetor de proximidade calculado a partir da comparação de distância de edição de cada assinatura da base de treinamento com a assinatura do vídeo de validação.

O vetor de proximidade pode ser ordenado considerando o menor valor de distância como o sinal mais provável de ser o sinal equivalente. Este processo pode ser representado pela Equação 3.1.

$$P(p) = Distancia(T(p), v) \tag{3.1}$$

Um exemplo pode ser apresentado considerando cada assinatura dos vídeos de treinamento. A Tabela 3.4 apresenta os 20 códigos de cada vídeo de treinamento, separados por cada lado, gerado para a palavra “entregar”.

Tabela 3.4: Códigos de vídeo da palavra “entregar”.

Vídeo	Mão direita	Mão esquerda
V_06_1_00_D.avi	011330010000004<	011330010000004<
V_00_2_00_D.avi	M313000000200001	M312000000200002
V_04_2_00_D.avi	133210000110000A	133120000010000B
V_00_3_00_D.avi	837200000000000;	737200000000000<
V_00_1_00_D.avi	641100100030000B	631100100010000E
V_05_2_00_D.avi	223020020000000<	222030010000000=
V_03_2_00_D.avi	351120100000000:	351210100000000:
V_19_1_00_D.avi	221110021000100;	221110021000100;
V_01_2_00_D.avi	221050000010001A	221050100000001A
V_07_1_00_D.avi	3122001102001117	3122001102001117
V_09_1_00_D.avi	123500130000000C	123510120000000C
V_02_1_00_D.avi	034030010000003D	134130010000003B
V_06_3_00_D.avi	222210220000101:	222220120000101:
V_06_2_00_D.avi	020110010000002C	020010010000002D
V_05_1_00_D.avi	1110101000001148	5120101000000027
V_07_2_00_D.avi	321030010010001?	321030010010001?
V_03_1_00_D.avi	052030400000301@	053030400000301?
V_01_1_00_D.avi	301410140100003=	301310140100003>
V_18_1_00_D.avi	502120010000000G	402110010010001G
V_08_1_00_D.avi	351010010010011>	441020000100011>

Usando a assinatura de teste como:

$$v_d = 6622002000010107$$
$$v_e = 6622002000000108$$

Tabela 3.5: Distância Levenshtein para a validação de exemplo direita v_d.

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

Com os valores da base e a cadeia de caracteres do vídeo de validação é possível calcular a proximidade $P(1)$ aplicado à assinatura $T(1)$ com o vídeo v utilizando um cálculo de distância de Levenshtein. A Tabela 3.5 resultado de proximidade para o primeiro vídeo é 9.

Tabela 3.6: Cálculo de proximidade entre cadeias de caracteres.

i	**Treinamento *T(i)***	***v***	***P(i)***	**Posicionamento**
1	011330010000004<	6622002000010107	9	
2	M313000000200001	6622002000010107	8	3
3	303010000000000A	6622002000010107	8	4
4	052520050010000>	6622002000010107	7	1
5	121100441001002>	6622002000010107	9	
6	101010104013001F	6622002000010107	9	
7	333100110000011;	6622002000010107	9	
8	120101001122202<	6622002000010107	9	
9	830300007113003H	6622002000010107	9	
10	202000110000000;	6622002000010107	7	2
11	520=01106713300O	6622002000010107	9	
12	250204511000<00A	6622002000010107	8	5
13	550206312148300;	6622002000010107	11	
14	491802502066001;	6622002000010107	9	
15	100120400000029N	6622002000010107	8	
16	310201:60250091C	6622002000010107	10	
17	A01000110000000=	6622002000010107	8	
18	701000000030003?	6622002000010107	8	

Seguindo o processo de comparação aplicado a toda a base de treinamento é obtido o valor de proximidade para cada combinação de *string* resultando no vetor *P* e para classificação pode-se considerar as 5 *strings* mais próximas conforme Tabela 3.6.

Utilizando os resultados de proximidade para cada teste são então sugeridos os prováveis resultados de classificação para cada sinal.

Capítulo 4

Resultados Experimentais

Neste capítulo são apresentadas as configurações utilizadas nos algoritmos, a metodologia usada para avaliar o desempenho do método proposto e os resultados experimentais realizados para validação do método proposto.

4.1. Configuração dos extratores de características

O método proposto neste trabalho utiliza os algoritmos SURF e SIFT para detecção das características nos quadros dos vídeos. O algoritmo SURF possui três parâmetros de configuração principais: limiar para a matriz Hessian, o número de pirâmides de oitavas e os números de camadas nas pirâmides.

Para o limiar utilizado na matriz Hessian foram comparados valores baseado no valor padrão do algoritmo e algumas variações acima e abaixo. A Figura 4.1 apresenta os pontos detectados dos quadros 11 ao 14 do vídeo V_06_1_00_D.avi da palavra entregar usando a seguinte configuração de limiar: (a) 100, (b) 300, (c) 500 e (d) 700; com o número de 4 pirâmides de oitavas e 2 camadas por pirâmide.

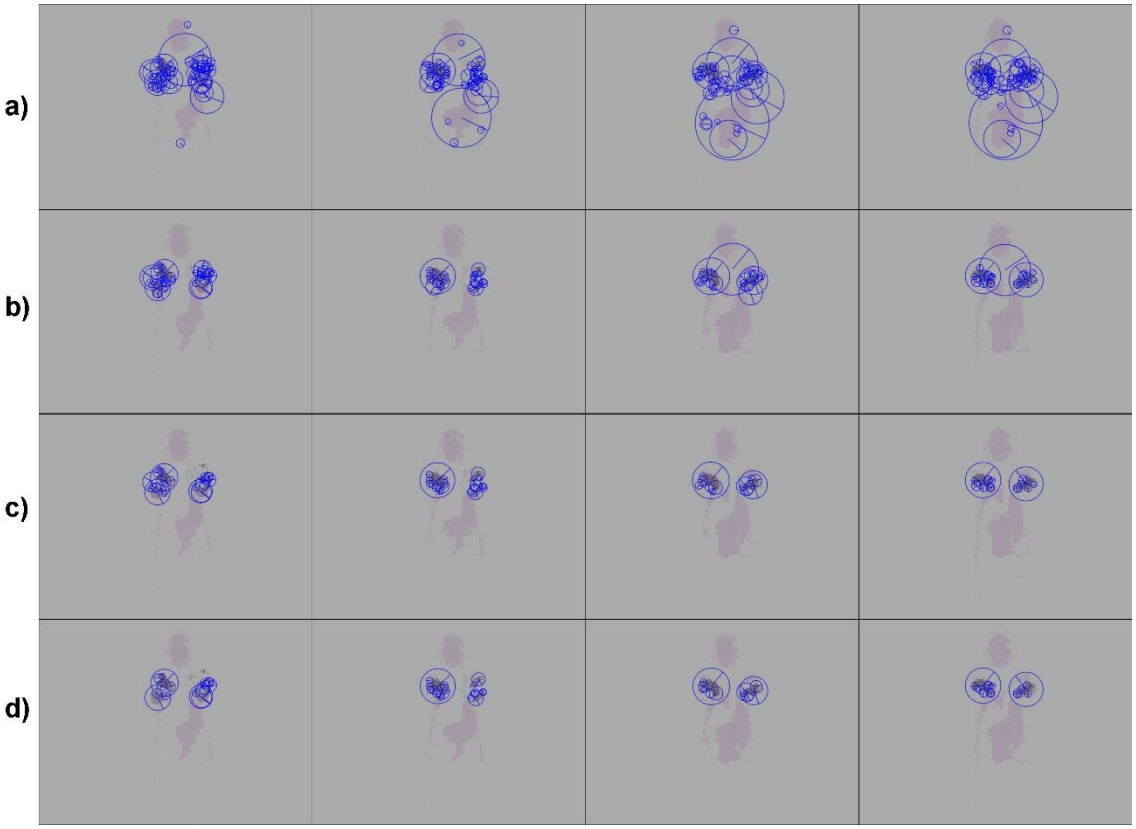


Figura 4.1: Pontos dos quadros com limiar (a) 100, (b) 300, (c) 500 e (d) 700.

A Tabela 4.1 representa os pontos de interesse detectados pelo algoritmo para cada configuração de Hessian e pirâmide de oitavas e as camadas por pirâmide.

Tabela 4.1: Tabela de comparação das configurações do SURF.

Hessian	**4 pirâmides de oitavas e 2 camadas por pirâmide**	**6 pirâmides de oitavas e 3 camadas por pirâmide**	**8 pirâmides de oitavas e 4 camadas por pirâmide**
100	942	1.128	1.252
300	415	494	536
500	275	396	429
700	206	334	365

Baseado nos testes de configuração é possível notar que as configurações utilizando o *Hessian* 100 apresenta mais pontos de interesse, mas captura pontos que estão fora da região de interesse para o processo de identificação do movimento. As outras configurações apresentam resultados melhores mantendo os pontos concentrados no movimento das mãos dos atores. A configuração com o limiar de 500 com 4 pirâmides de oitavas e 2 camadas identifica os pontos de interesse no movimento das mãos para as variadas palavras aplicadas incluindo as palavras com menores alterações de movimentos.

Para a configuração do algoritmo SIFT foram utilizadas comparações com variação de camadas de oitavas e quantidade de características selecionadas. Na Figura 4.2estão destacados os pontos de interesse selecionados para o quadro 17 do vídeo V_00_1_00_D.avi da palavra entregar usando a seguinte configuração: (a) 20 características com 3 oitavas, (b) 20 características com 6 oitavas, (c) 80 características com 3 oitavas e (d) 80 características com 6 oitavas.

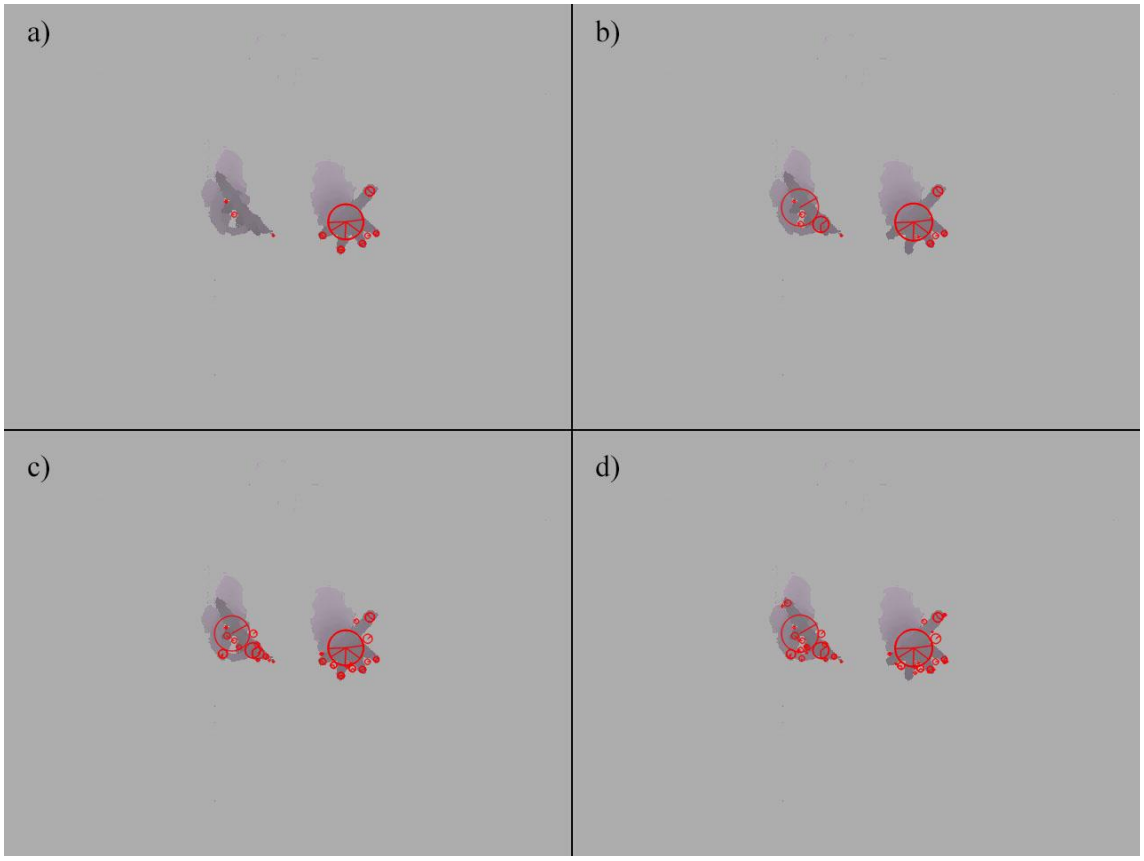


Figura 4.2: Pontos dos quadros com características (a, b) 20 (c, d) 80 e camadas por oitava (a, c) 3 (b, d) 6.

Os pontos detectados pela configuração de 20 características em 6 camadas por oitava atingem os principais pontos em destaque na imagem.

4.2. Comparação dos Algoritmos de Distância de *Strings*

As cadeias de caracteres para cada vídeo são geradas, baseada em um *codebook* onde cada entrada do *codebook* possui o mesmo peso, ou seja, a entrada 13 possui a mesma diferença de características que as entradas 3 e 29. Considerando este aspecto a matriz de similaridade e penalidade aplicada à similaridade não se aplicam ao processo.

Os resultados na comparação entre as cadeias de caracteres apresentam valores diferenciados que dependem do algoritmo utilizado. Uma comparação entre os algoritmos se fez necessária utilizando o mesmo processo de geração das cadeias para cada vídeo e executando o cálculo de distância com os vídeos de exemplares. Na Tabela 4.2 são comparados os resultados método utilizando três tipos de comparação de cadeias de caracteres na classificação dos vídeos, considerando as três primeiras classes selecionadas pelo método, nesta comparação foi executado o método usando o descritor SURF com *codebook* de características com 64 entradas e o *codebook* de histogramas com 32 entradas. O algoritmo de Levenshtein com custo fixo e o algoritmo de DTW apresentam resultados aproximados na primeira identificação e se aproximam do resultado de custo variável na segunda ou terceira opção da classificação. A confusão gerada pelo algoritmo DTW resulta da similaridade entre as cadeias do vídeo que possuem tamanho fixo e quantidade de caracteres limitada. Utilizando uma taxa de 1 até 5 vídeos como exemplares na classificação obtém-se uma taxa de acerto maior comparada aos 18 exemplares mas baixa em comparação ao Levenshtein.

Tabela 4.2: Comparação de acertos usando algoritmos de distância de *strings*.

Classes selecionadas	**Levenshtein**		**Dynamic Time Warping**
	Custo fixo	**Custo variável**	
Top 1	33%	60%	20%
Top 2	53%	67%	33%
Top 3	67%	73%	33%

Em uma comparação mais específica entre as variações do algoritmo de Levenshtein com custo fixo e custo variável por caractere é possível identificar o motivo da variação dos resultados. A Tabela 4.3 apresenta as 10 menores distâncias encontradas entre a cadeia de caracteres do vídeo de validação V_04_1_00_D.avi da palavra Entregar e as cadeias de caracteres de treinamento de todas as palavras. Para cada resultado é apresentada a palavra ao qual o vídeo de treinamento identificado pertence. É possível verificar que o algoritmo com custo variável encontra 5 correspondentes na palavra Entregar e 5 erros, sendo o menor valor em destaque na tabela encontrado na palavra correta. O algoritmo com custo fixo encontra 3 correspondentes na palavra Entregar e 7 erros nas outras palavras, gerando a confusão por identificar uma maior similaridade com a palavra Empurrar que é destacada como a menor distância.

Tabela 4.3: Menores distâncias de Levenshtein com custo fixo e variável.

Palavra	Distância de Levenshtein	
	Custo fixo	Custo variável
Entregar	17	25
Entregar	-	**23**
Entregar	18	25
Entregar	-	**23**
Entregar	18	**23**
Pegar	18	-
Pegar	18	-
Abrir	-	24
Olhar	18	-
Empurrar	18	24
Empurrar	**16**	-
Empurrar	-	25
Empurrar	17	24
Falar	18	-
Trabalhar	-	25

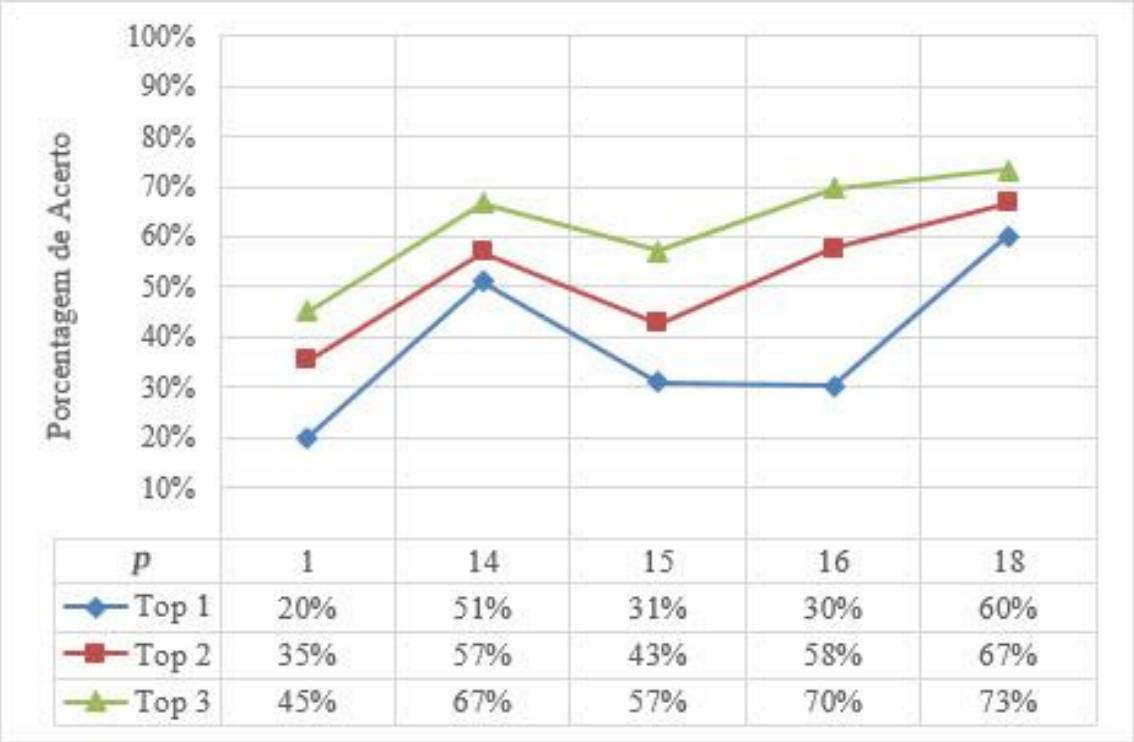
O algoritmo com custo variável identifica 5 acertos da palavra Entregar sendo três deles identificados com a menor distância em comparação com o de custo fixo que identifica a menor distância na palavra Empurrar.

Devido às características do algoritmo Levenshtein com custo variável é possível selecioná-lo como o melhor para a aplicação.

4.3. Análise de Resultados com até 3 Principais Indicações

Esta sessão apresenta a análise de resultados baseada nas 3 principais indicações propostas pelo algoritmo.

A Figura 4.3 apresenta a porcentagem de acertos considerando a primeira, segunda e terceira indicação do algoritmo por cada quantidade de treinamento utilizada. A maior taxa de acerto foi obtida com o valor de *p* para validação com 18 vídeos de treinamento por palavra. Considerando apenas a primeira indicação do algoritmo o melhor resultado foi 60% utilizando 18 vídeos por treinamento.

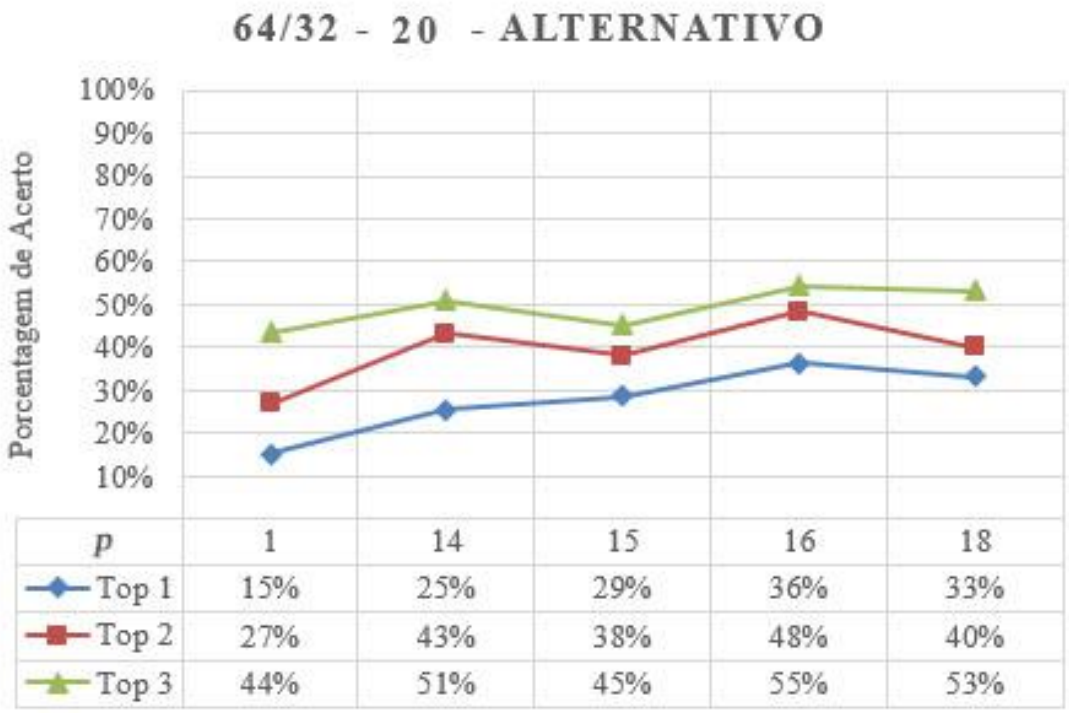


p	1	14	15	16	18
Top 1	20%	51%	31%	30%	60%
Top 2	35%	57%	43%	58%	67%
Top 3	45%	67%	57%	70%	73%

Figura 4.3: Taxa de acerto para cada *p* com até 3 indicações de palavra usando SURF.

Para este teste foi utilizado um *codebook* de 64 características diferentes e um *codebook* de 32 histogramas, o algoritmo SURF foi configurado usando limiar de corte Hessian de 500 com 4 pirâmides de oitavas e 2 camadas por pirâmide. Para cada processo foram selecionados *p* vídeos de treinamento resultando para cada palavra classificada até 3 indicações, considerando em ordem crescente a menor distância entre as cadeias de caracteres.

O teste foi repetido com o descritor SIFT configurado usando um total de 20 características com 3 camadas por pirâmide de oitava e as taxas de acerto ficam abaixo do descritor SURF. A Figura 4.4 apresenta a comparação entre os resultados usando o algoritmo SIFT, as taxas de acerto obtidas utilizando este descritor são menores. É possível observar que a diferença na quantidade de pontos de interesse resulta em maior confusão entre os resultados.



p	1	14	15	16	18
Top 1	15%	25%	29%	36%	33%
Top 2	27%	43%	38%	48%	40%
Top 3	44%	51%	45%	55%	53%

Figura 4.4: Taxa de acerto para cada p com até 3 indicações de palavra usando SIFT.

A variação utilizada para configurar cada um dos descritores influencia diretamente na comparação entre os códigos mas de maneira geral o algoritmo SURF detecta pontos de interesses mais representativos para os vídeos de profundidade utilizados.

4.4. Análise de Erros

No treinamento de 18 vídeos por palavra a matriz de confusão resultante é apresentada na Tabela 4.4. A matriz apresenta o maior índice de erro na palavra Empurrar e Fechar. As palavras Empurrar e Fechar não são identificadas pelo algoritmo gerando conflitos com outras palavras.

Tabela 4.4: Matriz de confusão com top-1 usando 18 vídeos por palavra.

	entregar	**pegar**	**abrir**	**olhar**	**empurrar**	**fechar**	**falar**	**puxar**	**trabalhar**
entregar	50%	0%	0%	25%	0%	0%	0%	25%	0%
pegar	0%	75%	0%	0%	0%	0%	0%	25%	0%
abrir	0%	25%	25%	0%	0%	0%	0%	50%	0%
olhar	0%	0%	0%	100%	0%	0%	0%	0%	0%
empurrar	25%	50%	0%	0%	0%	0%	0%	0%	25%
fechar	0%	0%	0%	0%	100%	0%	0%	0%	0%
falar	0%	25%	0%	0%	0%	0%	75%	0%	0%
puxar	0%	0%	0%	0%	50%	0%	0%	50%	0%
trabalhar	0%	25%	0%	0%	0%	25%	0%	0%	50%

A palavra Fechar possui o menor índice de acerto entre as palavras como identificado na **Erro! Fonte de referência não encontrada.**Tabela 4.4 gerando conflito com a palavra Empurrar. Na Figura 4.5 a comparação das imagens dos vídeos da palavra Fechar e Empurrar observa-se que os movimentos das mãos são realizados em momentos diferentes do vídeo mas possuem proximidade na posição final das mãos do ator. Essas características resultam em histogramas com valores próximos na separação dos quadros resultando em uma cadeia de caracteres semelhantes. Executando a comparação da cadeia de caracteres são identificados 3 dos 5 menores resultados dentro da classe da palavra Fechar e apenas 1 na palavra correta.

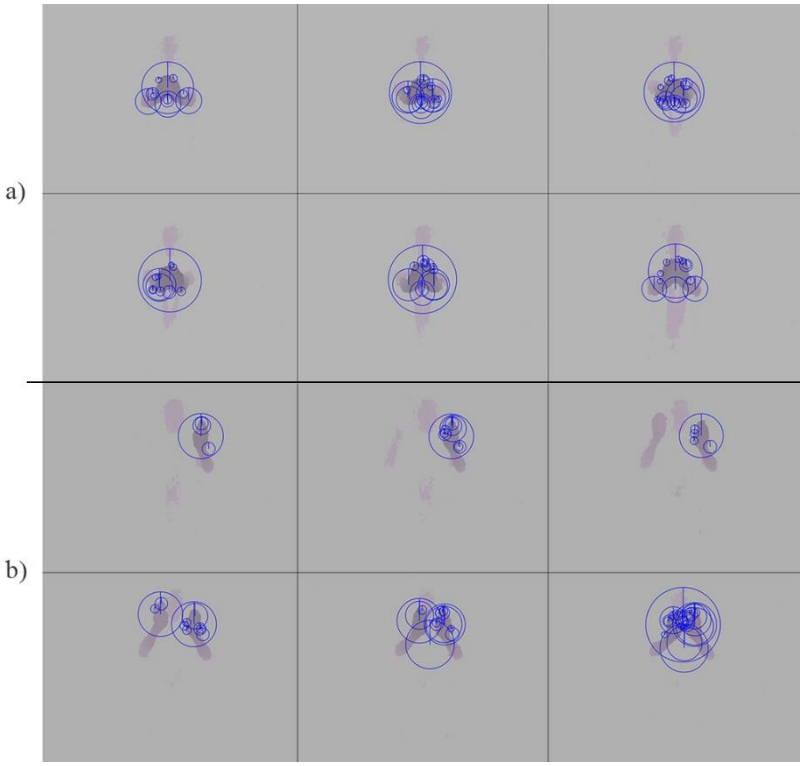


Figura 4.5: Comparação dos movimentos da palavra (a) Empurrar e (b) Fechar.

Observa-se também que a classificação possui maior conflito em duas palavras das 8 encontradas na base de vídeos, portanto, uma maior variação da base e diferenciação de característica pode resultar em melhores resultados.

4.5. Considerações Finais

Os resultados do método proposto demonstram que é possível realizar a classificação utilizando a comparação de cadeias de caractere com taxas de acerto de até 60% em média para primeira indicação. A principal variação dos resultados encontra-se na configuração do extrator de características e nos tipos de características utilizadas.

A taxa de acerto proposto neste trabalho fica abaixo dos resultados apresentados na Tabela 2.6 de comparação de resultados entre bases de dados, referenciada a seguir. Onde a menor taxa de acerto obtido usando um exemplar de treinamento atinge 74,36%.

Tabela 4.5: Comparação de resultados dos trabalhos participantes do ChaLearn

Referência	Base	Extrator de características	Classificador	Taxa de acerto (%)
Malgireddy et. al. [MAL12]	ChaLearn [CHA11]	HOG / HOF	Multi-channel HMM	81,54
Lui [LUI12]	ChaLearn [CHA11]	Least Squares Regression	HOSVD	76,97
Wu, Zhu e Shao [WUZ12]	ChaLearn [CHA11]	Extended-MHI e MSE	Correlation Coefficient Comparison	74,36
Konečný e Hagara [KON13]	ChaLearn [CHA11]	HOG / HOF	Quadratic-Chi distance com Single Model	89,02
			Quadratic-Chi distance com Multiple Model	81,53
Wan et. al. [WAN13]	ChaLearn [CHA11]	3D EMoSIFT	SOMP	87,41

Observa-se que o método proposto para classificação de gestos utilizando a distância de cadeias de descritores é valido conseguindo diferenciar determinadas características de cada gesto. Porém, existe a necessidade de se obter outras informações e características a serem extraídas do vídeo, como o rastreamento do movimento utilizando a informação de esqueleto do vídeo, a identificação da posição das mãos e outras características para agregar a geração da cadeia de caracteres.

Capítulo 5

Conclusão

Este trabalho apresentou um método para classificação de um vocabulário definido de gestos de LIBRAS. O método é capaz de codificar o vídeo do gesto utilizando extração de características e criando um processamento para conversão do vídeo em uma cadeia de caracteres que representa o movimento. A classificação utilizando distância de cadeia de caracteres é utilizada para fornecer uma lista de possibilidades na classificação.

A proposta foi avaliar a extração de características a partir de pontos de interesse identificados através do algoritmo SURF sem a utilização dos dados de esqueleto do interlocutor ou rastreamento das mãos.

O processo de configuração do algoritmo para extração de característica e *codebooks* de características foram testadas com variadas configurações para se obter o maior destaque e especificação para cada gesto. Um método foi desenvolvido para a geração de um histograma do vídeo completo separado pelo lado esquerdo e direito com o resultado codificado em forma de caracteres ASCII.

A comparação dos algoritmos de distância de cadeias de caracteres utilizando métodos como o Levenshtein, Damerau-Levenshtein e o DTW permitiu a identificação do método que consegue diferenciar melhor os códigos de cada vídeo e classificá-los entre as classes de cada palavra. No algoritmo de Levenshtein foi aplicado o método de custo variável que apresentou uma diferenciação em média de 50% melhor na comparação das classes.

Os testes foram realizados utilizando uma adaptação do método de validação cruzada *Leave-p-out* onde *p* para a quantidade de exemplares. Os resultados observados para a primeira opção de classificação (top 1) apresentam taxa de acerto de 51%

considerando 14 vídeos de modelo e 60% com 18 vídeos de modelo. Considerando as três indicações da classificação o resultado atingido é de 73% para 18 vídeos de modelo.

Os erros identificados entre as palavras classificadas concentram-se principalmente entre as palavras Empurrar e Fechar devido a quantidade próxima de características identificadas, tempo de movimento e características do gesto.

O método apresentado tem como vantagem a codificação de vídeo em formato simplificado e com a extração de características sem rastreamento específico de partes da imagem evita-se o uso d o esqueleto calculado pelo Kinect® ou a necessidade de maior resolução para o tratamento das configurações da mão. Entretanto o método apresenta a perda de informações de características importantes do gesto causando confusão entre palavras que possuem duração próxima.

Como sugestão de trabalho futuro há a possibilidade de implementar outros extratores de características para comparação com o SURF para detecção do movimento utilizando o mesmo processo de codificação dos vídeos em cadeia de caracteres. Com esse estudo é possível também adotar a abordagem de combinação dos extratores de características.

Referências

[BAY08] BAY, H.; ESS, A.; TUYTELAARS, T.; GOOL, L. V. SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding (CVIU)*, 2008. vol. 110, no. 3, p. 346-359.

[BRA02] BRASIL. Lei nº 10.436, de 24 de abril de 2002. Dispõe sobre a Língua Brasileira de Sinais - Libras e dá outras providências. *Diário Oficial da União*: Brasília, DF. 20 abr. 2002. Seção 1, p. 23.

[BRA05] BRASIL. Decreto nº 5.626, de 22 de dezembro de 2005. Regulamenta a Lei no 10.436, de 24 de abril de 2002, que dispõe sobre a Língua Brasileira de Sinais - Libras, e o art. 18 da Lei no 10.098, de 19 de dezembro de 2000. *Diário Oficial da União*: Brasília, DF. 23 dez. 2005. Seção 1, p. 28.

[CAL08] CALHAU, Ana; et. al. Alinhamento de sequências. Instituto Superior Técnico: Lisboa, 2008.

[CHA11] ChaLearn. *ChaLearn Gesture Dataset (CGD 2011)*. California, 2011.

[DAM64] DAMERAU, Fred J. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 1964.

[EDE97] EDELMAN, S.; INTRATOR, N.; POGGIO, T. Complex cells and object recognition. *Neural Information Processing Systems*, 1997. Não publicado.

[FAL13] FALAHATI, S. *OpenNI Cookbook*. Birmingham: Packt Publishing, 2013.

[GIO08] GIORGINO, Toni. Computing and Visualizing Dynamic Time Warping Alignments in R: The DTW Package. Journal of Statistical Software, ago. 2008. vol. 31. ed. 7. p. 1-24.

[GUY12] GUYON, I; et al. ChaLearn Gesture Challenge: Design and First Results. *In: Workshop on Gesture Recognition and Kinect Demonstration Competition*, jun. 2012. p. 4321-4326.

[GUY13] GUYON, I; et al. Results and Analysis of the ChaLearn Gesture Challenge 2012. *In: Proc. WDIA*, 2012, p.186-204.

[HAD09] HADJIKHANI, N.; KVERAGA, K.; NAIK, P.; AHLFORS, S. P. Early (N170) Activation of Face-Specific Cortex by Face-like Objects. *Neuroreport*, 2009. p. 403-407.

[HAY02] HAYKIN, S. *Redes neurais: princípios e prática*. 2 ed. Porto Alegre: Artes Médicas, 2002. p. 27-29.

[IBG10] IBGE – Instituto Brasileiro de Geografia e Estatística. *Censo Demográfico 2010: Características gerais da população, religião e pessoas com deficiência*. Rio de Janeiro: 2010. p.114.

[JAR89] JARO, M. A. Advances in record linkage methodology as applied to the 1985 census of Tampa Florida. *Journal of the American Statistical Society*, 1989. vol. 84, p. 414–420.

[KIM09] KIM, Tae-Kyun; CIPOLLA, Roberto. Canonical correlation analysis of video volume tensors for action categorization and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2009. vol. 31, p. 1253-1278.

[KON13] KONEČNÝ, Jakub; HAGARA, Michal. One-Shot-Learning Gesture Recognition using HOG-HOF Features. *In: Journal of Machine Learning Research*, 2013. vol. 1, p. 1-48.

[KUE11] KUEHNE, H.; JHUANG, H.; GARROTE, E.; POGGIO, T.; SERRE, T. HMDB: a large video database for human motion recognition. *In: Proccedings of the International Conference on Computer Vision (ICCV)*, 2011.

[LEV66] LEVENSHTEIN, Vladmir I. Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii Nauk SSSR*, 1965. vol. 163, n. 4, p. 845-848. Tradução do Russo In: Soviet Physics Doklady, 1966. vol. 10, no. 8, p. 707-710.

[LIN09] LIN, Zhe; JIANG, Zhuolin; DAVIS, Larry S. Recognizing actions by shape-motion prototype trees. *In: Proccedings of the International Conference on Computer Vision (ICCV)*, 2009.

[LIS00] LISETTI, C. L.; SCHIANO, D. J. Automatic classification of single facial images. *In: Pragmatics Cogn.*, 2000. vol. 8, p. 185–23.

[LOW99] LOWE, David. Object recognition from local scale-invariant features. *In: Proceedings of the International Conference on Computer Vision (ICCV)*, 1999. vol. 2, p. 1150-1157.

[LUI12] LUI, Yui Man. A Least Squares Regression Framework on Manifolds and its Application to Gesture Recognition. *In: CVPR2012 Workshop on Gesture Recognition*, 2012.

[MAL12] MALGIREDDY, M. R.; INWOGU, I.; GOVINDARAJU, V. A Temporal Bayesian Model for Classifying Detecting and Localizing Activities in Video Sequences. *In: Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2012. IEEE Computer Society Conference on, 2012. p. 43-48.

[MAT10] MATTAR, João. *Game em educação: como os nativos digitais aprendem*. Pearson Prentice Hall: São Paulo, 2010.

[MEN14] MENDONÇA, Vinícius G. *Método para Classificação de um Conjunto de Gestos usando Kinect*. PUC-PR: Curitiba, 2013.

[MIK05] MIKOLAJCZYK, Krystian; SCHMID, Cordelia. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005. vol. 27, n. 10.

[MIN12] MING, Y. et. al. Activity recognition from rgb-d camera with 3d local spatio-temporal features. *In: Proceedings of IEEE International Conference on Multimedia and Expo*, 2012. p. 344–349.

[MIT07] MITRA, Sushmita; ACHARYA, Tinku. Gesture Recognition: A Survey. *In: Systems, Man, and Cybernetics - Part C: Applications and Reviews*, maio 2007. vol. 37, n. 3. IEEE Transactions on, 2007. p. 311-324.

[NED70] NEEDLEMAN, Saul B.; WUNSCH, Christian D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 1970. v. 48, p. 443–453.

[OPE14] OpenCV. 2014. Disponível em: <http://opencv.org/>. Acesso em: 13 maio 2014.

[PER08] PEREIRA, Maria Cristina da C. Orientações Curriculares e Proposição de Expectativas de Aprendizagem para a Educação Infantil e Ensino Fundamental. *São Paulo: Secretaria da Educação do Estado de São Paulo*, 2008. p.22.

[SHA12] WU|, Di; SHAO, Ling; ZHU, Fan. One Shot Learning Gesture Recognition from RGBD Images. *In: Computer Vision and Pattern Recognition Workshops (CVPRW)*, Providence, 16 a 21 de jun. 2012. IEEE Computer Society Conference on, 2012. p. 7-12.

[SCH04] SCHÜLDT, C.; LAPTEV, I.; CAPUTO, B. Recognizing human actions: A local SVM approach. In: ICPR, 2004. p. 32-36.

[SMI81] SMITH, Temple F.; WATERMAN, Michael S. Identification of Common Molecular Subsequences. *Journal of Molecular Biology*, 1981. vol. 147, p. 195-197.

[WAN13] WAN, Jun; et. al. One-shot Learning Gesture Recognition from RGB-D Data Using Bag of Features. *In: Journal of Machine Learning Research*, 2013. vol. 14, p. 2549-2582.