Pós-Graduação em Ciência da Computação

Uma Proposta de Sistema para Tradução entre Linguagens de Sinais

Por

Davi Hirafuji Neiva

Dissertação de Mestrado

Universidade Federal de Pernambuco

posgraduacao@cin.ufpe.br

www.cin.ufpe.br/~posgraduacao

Recife, 2015

Universidade Federal de Pernambuco
Centro de Informática

Davi Hirafuji Neiva

Uma Proposta de Sistema para Tradução entre Linguagens de Sinais

Trabalho apresentado ao Programa de Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: *Cleber Zanchettin*

Recife
2015

Catalogação na fonte
Bibliotecária Jane Souto Maior, CRB4-571

N417p Neiva, Davi Hirafuji
Uma proposta de sistema para tradução entre linguagens de sinais / Davi Hirafuji Neiva. – Recife: O Autor, 2015.
119 f.: il., fig., tab.

Orientador: Cleber Zanchettin.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da computação, 2015.
Inclui referências.

1. Inteligência artificial - Aprendizagem. 2. Visão computacional. I. Zanchettin, Cleber (orientador). II. Título.

006.31 CDD (23. ed.) UFPE- MEI 2015-164

Dissertação de Mestrado apresentada por **Davi Hirafuji Neiva** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "**Uma Proposta de Sistema para Tradução entre Linguagens de Sinais**" orientada pelo **Prof. Cleber Zanchettin** e aprovada pela Banca Examinadora formada pelos professores:

_____

Prof. Leandro Maciel Almeida
Centro de Informática/UFPE

_____

Prof. Byron Leite Dantas Bezerra
Departamento de Engenharia de Produção / UFPB

_____

Prof. Cleber Zanchettin
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 28 de agosto de 2015.

_____

Profa. Edna Natividade da Silva Barros
Coordenadora da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

A Taciana e a minha família.

Agradecimentos

A realização deste trabalho foi um longo caminho, cheio de altos e baixos e momentos difíceis, mas no final deu tudo certo. Gostaria de agradecer a Taciana que estava lá em todos os momentos me apoiando e incentivando a continuar, acreditando até o último segundo. Dedico este trabalho a ela pois sei que sem ela, o caminho teria sido bem mais complicado. Dedico também este trabalho a minha família, ao seu suporte incondicional e entendimento da minha reclusão. Gostaria de agradecer ao meu orientador pelo seu conhecimento e prontidão a cada versão entregue. E por fim, dedico este trabalho a Miú, minha gata, que me fez companhia noites e mais noites adentro, dormindo loucamente na frente do monitor.

Agradeço imensamente a tudo e a todos. *It´s been a hell of a ride*.

“Taciana, meu computador quebrou”…“Tens backup?”...“Não”...(Silêncio)

Davi Hirafuji Neiva

Resumo

Na sociedade contemporânea existe uma forte campanha pela acessibilidade e pela inclusão de pessoas com deficiências. No Brasil há em torno de 9 milhões de surdos que usam a Linguagem Brasileira de Sinais (LIBRAS) e enfrentam diversas barreiras de comunicação. Várias iniciativas de aplicativos móveis proveem tradução entre LIBRAS e Português, porém a comunicação entre surdos de diferentes nacionalidades permanece um desafio. Este trabalho tem como objetivo propor um sistema que permita a pessoas surdas de diferentes nacionalidades se comunicarem através de sinais. Para isso, um aplicativo móvel captura gestos pela câmera e se comunica com um servidor web, que identifica o sinal através de uma rede neural artificial ELM (*Extreme Learning Machine*). O pré-processamento das imagens, para permitir que o sinal realizado pela mão seja isolado, combina técnicas de visão computacional como detecção de bordas, algoritmo de balde e uso de máscara para remoção do plano de fundo. O sistema também provê uma interface web para cadastro de sinais pelos usuários, permitindo a construção colaborativa da base de sinais internacional. Assim, o presente trabalho propõe uma abordagem inovadora para comunicação entre surdos de diferentes nacionalidades, aliada a uma plataforma colaborativa para construção da base de sinais, que pode também ajudar os surdos a aprenderem outras linguagens de sinais diferentes da sua.

Palavras-chave: Reconhecimento de gestos. ELM. Tradução entre sinais. Visão computacional. Redes Neurais Artificiais.

Abstract

Nowadays much has been done for accessibility and inclusion of people with disabilities using technology. In Brazil, there are approximately 9 million of deaf people that uses the Brazilian Sign Language (LIBRAS) e go through various communication barriers. There are some applications that provide a translation between LIBRAS and Portuguese, but communication between deaf people among different nationalities, using different sign languages, is still a great challenge. This research aims to provide a system that allows deaf people from different nationalities communicate with each other using their own sign language. In order to accomplish this, a mobile application captures gestures through the use of its camera and communicates with a web server, who identifies o sign using an artificial neural network ELM – Extreme Learning Machine. Images are processed in order to isolate the hand by combining computer vision technics like edge detection, bucket fill algorithm and masks to remove the background information. The system also provides a web interface to register new gestures as well as upload new images for a specific gesture, thus creating a collaborative gesture database. The present work purposes a novel approach for communication between deaf people with different nationalities combined with a collaborative platform to create the gestures database that can not only translate between different sign languages, but motivate deaf people to learn a different sign language.

Keywords: Gesture recognition. ELM. Translation between gestures. Computer vision. Artificial neural networks.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

3D	Três Dimensões
ASL	*American Sign Language*
CLCS	*Convexity Local Contour Sequences*
DTW	*Dynamic Time Wrapping*
ELM	*Extreme Learning Machine*
HSV	*Hue Saturation Value*
IHC	Interação Humano-Computador
IOS	*iPhone Operating System*
IIS	*Internet Information Service*
FSM	*Finite State Machine*
GPS	*Global Positioning System*
HMM	*Hidden Markov Models*
LIBRAS	Linguagem Brasileira de Sinais
LBP	*Local Binary Patterns*
LSTM	*Long Short-Term Memory*
MDRNN	*Multi-Dimensional Recurrent Neural Network*
PCA	*Principal Component Analysis*
RGB	*Red Green Blue*
SMS	*Short Message Service*
SVM	*Support Vector Machines*
TDD	*Telecommunications Device For The Deaf*

USB	*Universal Serial Bus*
VOIP	*Voice Over Internet Protocol*

Sumário

Capítulo 1
Introdução

Na área da computação, há constantes mudanças, constantes inovações e novas formas de pensar. Em meados de 1965, Gordon Moore lançou uma teoria (MOORE, 1965), que se tornou norteadora para a indústria, segundo a qual a cada 18 meses, o número de transistores iria duplicar juntamente com o poder de processamento, sem representar nenhum custo a mais na fabricação, e sem comprometer o espaço em uso e o consumo de energia (MOORE, 1965). Para se ter uma ideia desse crescimento, o primeiro processador Intel 8008 [1] tinha 3.500 transistores. Dez anos depois, por volta de 1982, o Intel 80286[2] tinha 134.000 e o Pentium 4 tinha 500 milhões[3]. Este crescente poder de processamento, juntamente com o poder de armazenamento em memória, tem possibilitado o surgimento e acoplamento de novos dispositivos de *hardware*, novas tecnologias e novas formas de aproveitar este potencial.

Por outro lado, a área de Interação Humano-Computador (IHC) vem investigando, nas últimas décadas, como prover uma melhor interação entre humanos e máquinas. Hoje em dia existem tecnologias que fogem do paradigma de teclado e *mouse* e proveem uma dinâmica diferente usando algo mais natural ao homem: o movimento do corpo, capturando os gestos que os humanos fazem tão naturalmente. Cada vez mais, surgem aprimoramentos e novas formas de captura de gestos pelas máquinas. Por exemplo, os videogames captam o movimento do jogador usando câmeras, sensores e luzes, e interagem de acordo, como no caso dos consoles

[1] http://petsd.net/8008.php

[2] http://www.computermuseum.li/Testpage/Chip-Intel80286.htm

[3] http://www.intel.com/p/pt_BR/support/highlights/processors/pentium4

XBOX[4], PLAYSTATION[5] e Wii[6]. Já existe *hardware* capaz de detectar o movimento das mãos, como o *LeapMotion*[7], e até mesmo um anel capaz de detectar o movimento do dedo chamado de *RING*[8]. Ambos são dispositivos de *hardware* que por meio de sensores e giroscópio conseguem mapear e capturar o movimento das mãos e consequentemente dos dedos, e transmitem via *Bluetooth* (no caso do *RING*) ou USB (no caso do *LeapMotion*).

Novos tipos de tecnologia estão sendo embarcados também nos aparelhos celulares. Novas funcionalidades incluem: detecção de movimento da íris, através da qual um vídeo é pausado quando o usuário não está mais olhando para o celular; e detecção de movimento da mão, que permite realizar uma determinada ação como parar um vídeo, mudar uma página de leitura, fazer uma ligação, entre outros, através de gestos específicos. Cada vez mais, é possível mapear a diversidade de gestos humanos em ações correspondentes em dispositivos móveis.

Atualmente, um celular é um dispositivo de fácil mobilidade podendo ser levado para qualquer lugar a qualquer instante. Estas características fomentaram o uso deste dispositivo no desenvolvimento desta dissertação de mestrado, a qual objetiva investigar o reconhecimento de gestos em dispositivos móveis, no contexto de linguagens de sinais utilizadas por pessoas surdas para se comunicarem. Além dessa facilidade, o celular possui recursos de câmera, permitindo a captura dos movimentos, e acesso à Internet que viabilizará a comunicação com um servidor para enviar sinais a serem classificados. O objetivo principal desta pesquisa é propor um tradutor de linguagens de sinais, permitindo que uma pessoa surda, ao ter em mãos este sistema e estando em um país que possui uma linguagem de sinais diferente da sua, possa saber o significado de um sinal que uma outra pessoa está fazendo na sua linguagem e vice-versa.

Entretanto, as necessidades técnicas que este sistema demanda requerem um alto poder de processamento computacional, e muito espaço de memória e armazenamento. Devido ao tamanho de um dispositivo móvel, seu poder

---

[4] http://www.xbox.com/en-US/

[5] https://www.playstation.com/

[6] http://www.nintendo.com/

[7] https://www.leapmotion.com/

[8] http://logbar.jp/ring/en

computacional é limitado em relação a um computador de mesa e principalmente um servidor de processamento de informações. Estas limitações dificultam a viabilização da classificação de sinais no celular. A relação de poder computacional e memória com tempo é inversamente proporcional: treinar um classificador em um celular iria afetar seu desempenho, consumir muita memória e limitar muito seu uso, afetando diretamente o consumo da bateria. Com isto em mente, unimos o poder computacional de um servidor com a mobilidade de um celular, resultando assim em uma divisão do desenvolvimento em um módulo servidor e um módulo móvel, cada um com suas funções e particularidades.

1.1 Justificativa

Na sociedade atual, existe uma forte campanha pela inclusão de pessoas com deficiências diversas, buscando prover acessibilidade para todos, sejam eles deficientes físicos ou intelectuais. Com os novos recursos computacionais, abriram-se muitas possibilidades para aumentar o acesso de pessoas com deficiências tanto para usar estas tecnologias quanto para ter uma maior facilidade em sua vida cotidiana. Um exemplo de uso da tecnologia por deficientes auditivos é o telefone para surdos (*Telecommunications Device For The Deaf* – TDD[9]). Trata-se de um telefone público comum, que é acoplado a um aparelho de teletexto em que um atendente faz o serviço de intermediação entre um deficiente auditivo e uma pessoa ouvinte. O surdo interage com o dispositivo digitando e lendo mensagens de texto no teletexto. Apesar de exemplos como este, a comunicação é algo que, na maioria das vezes, é um desafio no cotidiano do deficiente auditivo, pois nem sempre ele vai ter um intérprete ao lado, e nem sempre seu interlocutor vai entender a sua linguagem de sinais. A repetição deste fato pode acabar frustrando ambas as partes.

Com o advento dos *smartphones*, realizar uma ligação tornou-se apenas uma pequena funcionalidade dentro da gama de possibilidades do celular. O *smartphone* é um dispositivo que se tornou, para muitas pessoas, um item indispensável no dia a dia, e com o avanço da tecnologia, tem-se pensado em novas formas de interação com o usuário, como telas sensíveis ao toque, realidade aumentada, uso de câmeras, voz, sensores (acelerômetro, giroscópio) e GPS. O *smartphone* também provê novas

[9] http://atwiki.assistivetech.net/index.php/TDD_(Telecommunications_Device_for_the_Deaf)

formas de comunicação como mensagens de voz, videoconferência, *Voice Over IP* (VOIP), bate-papos, redes sociais, entre outros. Mas o que tornou o *smartphone* tão usado foi sua praticidade, facilidade de interação e o fato de executar uma gama de aplicativos que atende às mais diversas necessidades dos usuários.

Porém, hoje em dia, ainda há poucas aplicações móveis direcionadas a pessoas com deficiência auditiva. O deficiente auditivo usa o celular principalmente se comunicando através de mensagens de texto (SMS e aplicativos de mensagens). Isto pode ser problemático, pois muitos surdos não têm facilidade de utilizar uma língua escrita, já que a sua língua materna geralmente é a linguagem de sinais. Uma alternativa para isto se dá através de videoconferência, mas esta também tem limitações, pois só funcionaria entre pessoas que saibam a mesma linguagem de sinais.

Existem alguns exemplos de aplicativos especificamente desenvolvidos para deficientes auditivos. O *SignLanguage!*[10], para o sistema operacional IOS, ensina a linguagem de sinais americana (*American Sign Language* - ASL) através de vídeos e imagens, contendo alguns exemplos e exercícios. Na mesma linha, há o *LVI Libras*[11], para o sistema operacional Android, que ensina a linguagem de sinais brasileira também por vídeos e imagens. Para a plataforma *Windows Phone*, tem-se o *ProDeaf*[12], aplicativo criado por alunos da Universidade Federal de Pernambuco que se diferencia por permitir a comunicação entre surdos brasileiros, cuja língua materna é LIBRAS, e pessoas ouvintes que falam Português brasileiro. O aplicativo utiliza a câmera do celular e uma luva de tecido colorido para permitir o reconhecimento dos gestos, e traduz LIBRAS para áudio em português e vice-versa, utilizando um avatar 3D. O *ProDeaf* foi o primeiro aplicativo de tradução instantânea de LIBRAS. Porém, como existem muitos regionalismos nas linguagens de sinais, a equipe do *ProDeaf* reduziu o conjunto de informações suportadas no desenvolvimento do seu protótipo.

Apesar de já existirem algumas soluções para reconhecimento de gestos em aparelhos de celular, em particular no contexto de linguagem de sinais, o presente trabalho possui dois diferenciais principais: a proposta de tradução instantânea entre diferentes linguagens de sinais; e o envolvimento ativo dos usuários, de maneira que

---

[10] https://itunes.apple.com/us/app/sign-language!/id321202730?mt=8

[11] http://getplayapps.com/lvi-libras/

[12] https://www.microsoft.com/en-us/store/apps/prodeaf/9nblggh09frl

eles possam alimentar o conteúdo das bases de sinais usadas para as traduções. Isso é viabilizado a partir de uma página *web*, que apresenta os sinais que já existem no dicionário, e permite ao usuário cadastrar novos sinais ou adicionar novas imagens relacionadas a um sinal já cadastrado.

1.2 Objetivos

Este trabalho tem como principal objetivo realizar o reconhecimento de gestos em dispositivos móveis para traduzir o sinal entre diferentes linguagens de sinais. Os objetivos específicos descritos a seguir contribuem para se alcançar tal objetivo geral:

- Capturar as imagens de sinais a partir de um dispositivo móvel;
- Tratar as imagens a fim de deixar apenas a mão, na posição de um sinal, em destaque;
- Escolher um classificador e realizar seu treinamento a cada nova imagem recebida na base;
- Centralizar as imagens dos sinais em um único repositório, facilitando a atualização da base para todos os usuários a qualquer instante;
- Permitir que os usuários construam colaborativamente a base de sinais em diversas linguagens;
- Prover um aplicativo para a interação do usuário e a tradução dos sinais.

1.3 Estrutura desta Dissertação

O presente trabalho está dividido em 8 capítulos. No capítulo 2 é apresentada uma breve explicação do que são gestos e linguagens de sinais.

O capítulo 3 fornece a fundamentação teórica, apresentando as duas técnicas de inteligência artificial utilizadas na pesquisa e as técnicas de processamento de imagens.

No capítulo 4, é apresentada uma revisão de literatura, mencionando alguns trabalhos na área de pesquisa de reconhecimento de gestos.

O capítulo 5 descreve a abordagem seguida nesta pesquisa, apresentando uma visão geral do sistema proposto, seguida de um detalhamento dos módulos implementados e o fluxo geral incluindo interação do usuário e comunicação entre os

módulos. Este capítulo também expõe a aplicação dos métodos de processamento e classificação de imagens no contexto deste trabalho.

O capítulo 6 descreve os métodos utilizados para o funcionamento deste sistema tais como a base de dados criada e os dispositivos usados para a realização dos experimentos.

O capítulo 7 apresenta os resultados desses experimentos, relacionados especificamente à remoção do plano de fundo das imagens, classificação e tradução de sinais.

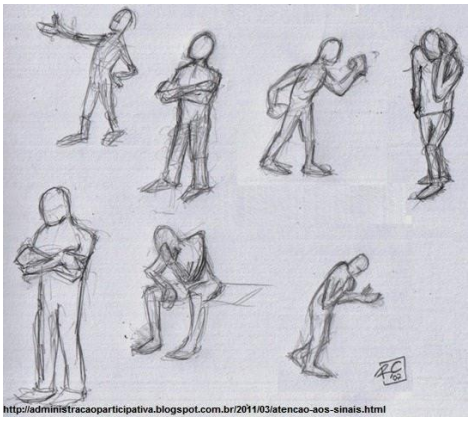
Por fim, o capítulo 8 expõe as considerações finais, destacando as principais contribuições desta pesquisa, e indicando algumas limitações do sistema desenvolvido, seguidas de trabalhos futuros para aprimorar o mesmo.

Capítulo 2
Gestos e Linguagens de Sinais

Gestos são movimentos do corpo que podem envolver dedos, mãos, braços, cabeça e face, podendo ser estáticos ou dinâmicos ou uma combinação dos dois, e possuem o intuito de transmitir algum tipo de informação (MITRA & ACHARYA, 2003), (DEUSDADO, 2002). Os gestos possuem várias finalidades (LISETTI, 2000), entre elas destacam-se a interação com e entre surdos, expressão de emoções, alerta em certas situações, e, com as novas tecnologias, navegação em ambientes virtuais e interação entre humanos e máquinas (MITRA & ACHARYA, 2003), (DEUSDADO, 2002).

As linguagens de sinais utilizam os movimentos das mãos e expressões faciais e corporais, em vez de sons, para permitir uma comunicação entre surdos ou pessoas com deficiência auditiva. A maior parte da comunicação se dá através do movimento das mãos, através do qual são representadas as palavras e letras (BRECAILO, 2012). As expressões coporais e facias acompanham para dar ênfase a uma sentença, embora elas possam expressar uma palavra, ideia ou sentido sem precisar usar as mãos, conforme os exemplos na Figura 1 (a). A Figura 1 (b) mostra algumas expressões corporais que podem representar: preocupação, angústia, raiva, indiferença.

(a)[13]



(b)[14]

Figura 1: Expressões faciais e seus significados

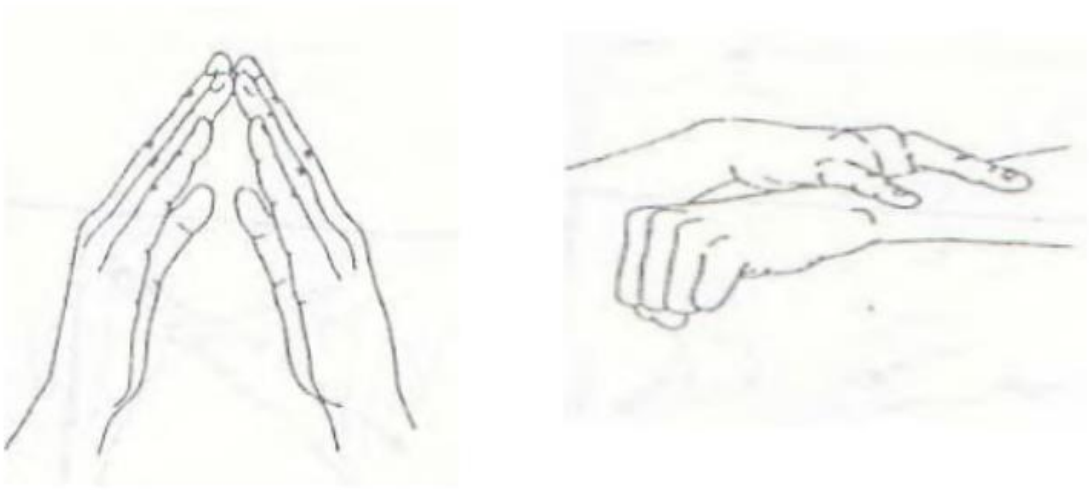
As palavras ou itens lexicais das linguagens de sinais são formados por combinações da forma e do movimento das mãos e do ponto no corpo ou no espaço onde esses sinais são feitos. Como toda linguagem, as línguas de sinais possuem fonologia, morfologia, sintaxe e semântica, e não são universais. Cada país possui a sua própria língua de sinais, que sofre influências da cultura nacional. Como qualquer outra língua, elas também possuem expressões que diferem de região para região, o que as legitima ainda mais como ferramentas de comunicação. A linguagem de sinais brasileira, conhecida como LIBRAS[15], teve sua origem na Língua de sinais Francesa (ALMEIDA & ALMEIDA, 2012).

Os sinais em LIBRAS podem representar letras ou palavras. Usualmente, a comunicação é feita através de sinais que representam palavras, como na Figura 2.

[13] Fonte: http://paulaleonny.no.comunidades.net/fotos-de-expressao-facial-e-seus-significados

[14] Fonte: http://arteculturasurda.blogspot.com.br/2014/04/a-importancia-da-linguagem-gestual-e.html

[15] http://www.libras.org.br/libras.php



(a) (b)

Figura 2. Exemplos de sinais em LIBRAS que representam palavras
(a) Casa e (b) banheiro[16]

Porém, quando uma palavra não tem um sinal, como no caso de um nome próprio, a palavra precisa ser soletrada. Nesse caso usa-se a técnica de datilologia, representando cada letra por um sinal, como exemplificado na Figura 3.

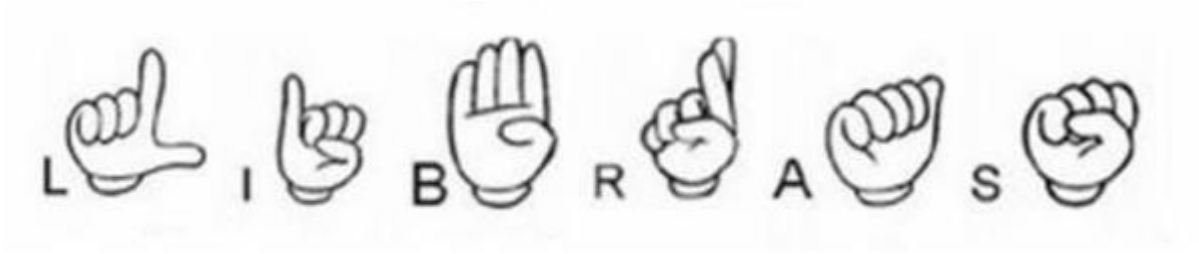


Figura 3: Exemplos de sinais de letras em LIBRAS[17]

Além disso, os sinais também podem ser estáticos ou dinâmicos. Sinais estáticos são aqueles representados por uma posição da mão, enquanto que sinais dinâmicos são representados por uma combinação de movimentos e posição da mão. Temos alguns casos no alfabeto brasileiro como as letras exemplificadas na Figura 4.

[16] http://pt.slideshare.net/aucisantos/formao-na-sre-colatina-1-parte-sinais-em-libras

[17] Universidade Estadual do Ceará, Faculdade de Educação, Ciências e Letras de Igatu (www.uece.br/fecli)

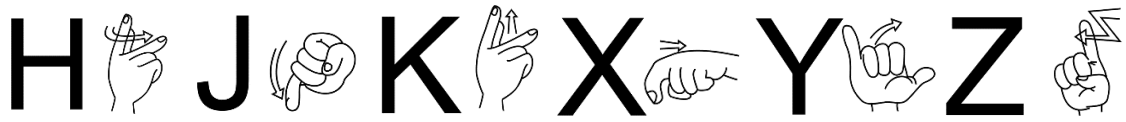


Figura 4: Exemplos de sinais dinâmicos em LIBRAS[18]

A técnica de datilologia é mais usada para soletrar o nome de pessoas, lugares e outras palavras que ainda não possuem sinal[19]. Entretanto, as palavras soletradas são, muitas vezes, substituídasao longo do tempo por sinais criados pelos surdos, ou seja, no momento em que o conceito de um sinal é entendido por seus interlocutores, substitui-se a datilologia da palavra (SOARES, 2014). Por isso, em LIBRAS, é permitida a criação de um sinal completamente novo que pode ser atrelado a uma pessoa, substituindo o seu nome, para evitar a datilologia de um nome toda vez que ele é mencionado. Cada pessoa pode inventar seu próprio sinal e usá-lo em uma conversa ou para se apresentar. Normalmente, isto ocorre quando se trata de nomes próprios, que não têm significado, não existindo, assim, no dicionário de LIBRAS.

As expressões faciais (BRECAILO, 2012) têm um papel importante, como mostrado na Figura 5, similar à entonação dada em uma palavra falada. A expressão facial substitui a entonação e assim pode-se mudar o sentido de uma palavra, dar uma afeição à frase e estabelecer qual tipo de frase está sendo sinalizada: afirmativa, imperativa, negativa, exclamativa ou interrogativa. Por exemplo, ao fazer uma pergunta, as sobrancelhas ficam franzidas e um ligeiro movimento da cabeça para cima é feito. Para fazer uma frase exclamativa, as sobrancelhas ficam levantadas, a cabeça inclina-se para cima e para baixo fechando a boca para intensificar.

[18] Ideia Criativa (www.ideiacriativa.org)

[19] http://www.girafamania.com.br/girafas/lingua_sinais.html

Figura 5: Exemplos de expressões faciais que acompanham sinais
Fonte: (FELIPE, 2013)

A linguagem de sinais passou por várias intervenções e até chegou a ser abolida por um tempo (STROBEL, 2009), até ser o que ela é hoje em dia. Em meados de 1750, os surdos viviam em condição de miséria por não conseguirem se comunicar e se expressar com outras pessoas e eram excluídos da sociedade (FALCÃO L. A., 2007) (STROBEL, 2009). No Brasil, a linguagem de sinais começou a surgir em 1857 quando o professor francês Hernest Huet fundou o Instituto Nacional dos Surdos-Mudos no Rio de Janeiro e em 2005 foi regulamentada através do Decreto 5626 (BRASIL, 2005), que diz que a linguagem brasileira de sinais deve ser falada em todos os ambientes escolares, empresariais, universidades e espaços públicos. Há muito trabalho sendo feito para expandir o ensino de LIBRAS no Brasil. Livros publicados mostram métodos e novas abordagens de como lidar com surdos quando diagnosticados cedo (FALCÃO L. A., 2010), e didática para o ensino de LIBRAS (FALCÃO L. A.) (FALCÃO L. A., 2012).

Assim como a língua portuguesa ou qualquer língua falada, LIBRAS varia de região para região. Além disso, uma pessoa que se comunica por LIBRAS não poderia usar esta linguagem em outro país, que por sua vez possui a sua própria linguagem de sinais. Para permitir esta comunicação, nesta dissertação propõe-se um sistema de tradução entre sinais a fim de facilitar a comunicação com e/ou entre deficientes auditivos. Para implementar essa tradução, são necessárias técnicas de reconhecimento de gestos, que fundamentam a parte teórica deste trabalho, descrita no próximo capítulo.

Capítulo 3
Fundamentação Teórica

Desde o trabalho realizado por (GUNNAR, 1973), em que foi proposto que os gestos humanos poderiam ser reconhecidos através da informação do movimento, os perfis de movimento e suas trajetórias têm sido investigados para fins de reconhecimento de gestos humanos por máquinas. O reconhecimento de gestos é um exemplo de pesquisa interdisciplinar, situada na ampla área da Inteligência Artificial (IA).

Dentre as principais técnicas desta área, serão abordadas as técnicas investigadas no desenvolvimento desta dissertação: Máquinas de Vetores de Suporte (SVM – do inglês *Support Vector Machines*) e Máquinas de Aprendizado Extremo (ELM – do inglês *Extreme Learning Machines*). Ambas são técnicas de classificação que determinam que solução deve ser tomada diante de informação desconhecida, com base na representação do conhecimento previamente existente. Para este trabalho, elas foram usadas no contexto de classificação de imagens (dos sinais). Cada sinal neste trabalho é representado por várias imagens, para aumentar as chances de uma classificação correta. Assim, dados conjuntos de imagens já conhecidas / classificadas, o objetivo é descobrir em qual conjunto uma nova imagem poderia se enquadrar. Ao final deste processo de classificação, é possível saber a qual sinal corresponde a nova imagem. Os dados aqui utilizados são contínuos, englobando um problema de classificação de multiclasses.

Porém, antes da classificação, há um passo importante realizado neste trabalho que é o processamento das imagens. A fim de remover informações desnecessárias que não fazem parte de um sinal (plano de fundo), as imagens passam por um "tratamento" para destacar apenas a mão na imagem.

As próximas seções explicam as técnicas de Processamento de imagens e Aprendizagem de Máquina utilizadas.

3.1 Processamento de Imagens

Neste trabalho, o processamento de imagens tem o objetivo de remover o plano de fundo das imagens e deixar somente a mão em destaque. Isto é feito porque informações do plano de fundo não são relevantes para um sinal e retirá-las da imagem, diminuindo a quantidade de informações, melhora o resultado da etapa de classificação dos sinais. Algumas técnicas foram utilizadas para atingir este objetivo.

A primeira delas foi a detecção de pele, baseada no trabalho de (OLIVEIRA & CONCI, 2009), ilustrado na Figura 6.

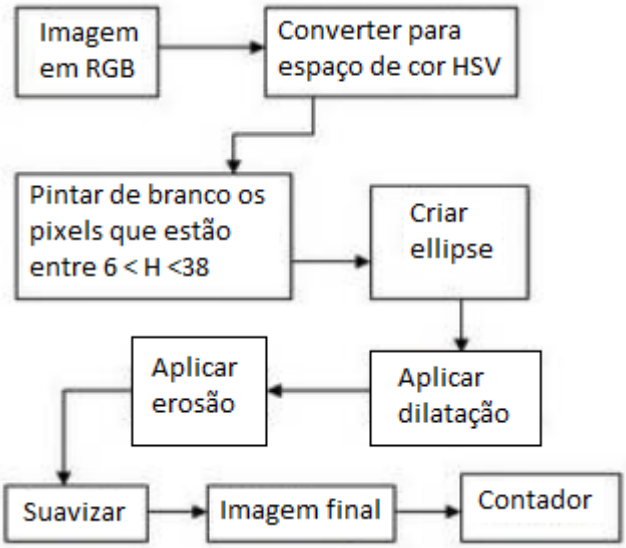


Figura 6: Fluxo de detecção de pele
Adaptada de: (OLIVEIRA & CONCI, 2009), tradução nossa

Como a figura mostra, os autores propõem a conversão da imagem do espaço RGB para o espaço de cor HSV, que é mais relacionado à percepção humana das cores. Além disto, os autores definem que a pele no canal H tem um limiar entre 0 e 50 e no canal S entre 0.23 e 0.68, para as etnias asiáticas e caucasianas. Os autores utilizam apenas o canal H com limiar entre 6 e 38 e foram aplicados alguns filtros para obter o resultado mostrado na Figura 7.

Figura 7: Resultado final da detecção de pele proposta no artigo

Fonte: (OLIVEIRA & CONCI, 2009)

A técnica de detecção de pele foi investigada neste trabalho para destacar apenas a mão na imagem.

Uma outra técnica utilizada foi a detecção de bordas, para detectar o contorno da mão. Esta técnica foi aplicada através do algoritmo do filtro de *Canny*, escolhido devido a sua baixa taxa de erro e boa localização (SUCHARITA, JYOTHI, & MAMATHA, 2013). Boa localização quer dizer que as bordas detectadas devem ser as mais próximas possíveis das bordas reais.

Este filtro tem como finalidade encontrar os contornos de uma imagem utilizando os seguintes passos:

1. Filtrar ruídos da imagem através do filtro gaussiano:

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

2. Encontrar o gradiente de intensidade da imagem:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} \quad G = \sqrt{G_x^2 + G_y^2}$$

$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

3. Remover pixels que não fazem parte de uma linha;
4. Aplicar dois limiares (máximo e mínimo) definidos ao chamar a função.
    a. Se o gradiente de um pixel for maior do que o limiar máximo, o pixel é aceito como parte de uma borda;
    b. Se o gradiente for menor que o mínimo, ele é rejeitado;

c. Se o gradiente estiver entre os dois limiares, ele vai ser aceito somente se estiver conectado a um pixel que for maior que o limiar máximo.

Há algumas maneiras de definir os limiares máximo e mínimo. Uma delas pode ser por tentativa e erro, método que pode tomar tempo em achar os valores ideais e pode funcionar para apenas uma imagem. Outra maneira é calcular a média dos pixels da imagem e fornecendo uma porcentagem (*sigma)* para ser calculada em cima da média (ROSEBROCK, 2015). No exemplo visto o valor de sigma foi de 0.33. Os valores são achados através da seguinte fórmula:

mínimo = int(max(0, (1.0 - sigma) * v))

máximo = int(min(255, (1.0 + sigma) * v))

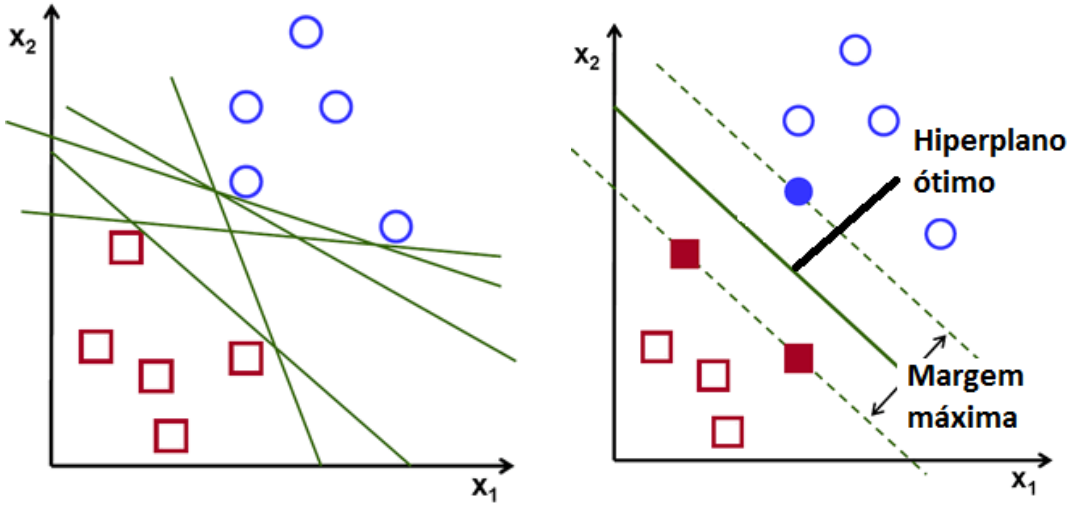
onde v é a média dos pixels da imagem, e sigma corresponde ao valor percentual a ser usado.

Em combinação com a técnica de detecção de bordas, é usado um algoritmo de balde para preencher um espaço que tem uma determinada cor. O algoritmo de balde é um algoritmo recursivo que visa preencher os 4 (acima, abaixo, à esquerda, à direita) ou 8 (contornando o pixel inicial) pixels vizinhos dado um ponto inicial. A cada pixel preenchido, o algoritmo chama novamente ele mesmo para cada pixel vizinho, parando quando não acha mais nenhum pixel com a cor inicial. Considerando-se que os contornos da mão já foram detectados, o algoritmo de balde é usado para deixar o espaço interno da mão uniforme, eliminando ruídos na mão. Na prática, este algoritmo objetiva preencher de branco todos os espaços pretos dentro do contorno da mão.

3.2 Support Vector Machines

As máquinas de vetores de suporte, mais conhecidas como SVM (do inglês *Support Vector Machine*) constituem uma técnica de aprendizado de máquina que vem crescendo bastante em uso. Ela é utilizada em diversas áreas da computação, tais como reconhecimento de textos, análise de imagens e bioinformática. A teoria do SVM estabelece uma série de princípios que devem ser seguidos na obtenção de classificadores com boa generalização, definida como a sua capacidade de prever corretamente a classe de novos dados do mesmo domínio em que o aprendizado ocorreu.

O SVM é baseado no conceito de planos de decisão que definem as regiões de decisão. O hiperplano separa os conjuntos de objetos que pertencem a classes diferentes. A Figura 8 mostra um conjunto de objetos que podem pertencer à classe vermelha (representada por quadrados), à esquerda ou à classe azul (representada por círculos), à direita. Podemos observar na Figura 8 (a) que existem vários classificadores lineares que podem separar as duas classes, mas apenas um que maximiza a margem de separação (distância da instância mais próxima ao hiperplano de separação das duas classes em questão). O hiperplano com margem máxima é chamado de hiperplano ótimo, que será o objeto de busca do treinamento do classificador. A linha no gráfico da Figura 8 (b) separa as duas classes. Os exemplos mais próximos da linha de separação são chamados de vetores de suporte. Qualquer nova instância que vier a ser classificada, se for posicionada do lado esquerdo da linha, é classificada como sendo da classe azul, e caso se localize no lado direito da linha, pertence à classe vermelha (VAPNIK & CORTES, 1995).



(a) (b)

Figura 8: Classificação SVM linear

Adaptada de: ((d) Opencv Dev Team, 2015), tradução nossa

A Figura 8 (a) mostra que há várias linhas que podem ser traçadas a fim de separar as duas classes. O exemplo da Figura 8 (b) mostra a essência do SVM, que é a construção de um hiperplano ótimo, de modo que ele possa separar diferentes classes de dados com a maior margem possível entre elas.

O SVM foi inicialmente projetado em 1962 para trabalho com problemas lineares (VAPNIK & CORTES, 1995), ou seja, de fácil separabilidade, entre duas classes conforme vimos na Figura 8. Em 1992, foi proposto o uso do SVM para

classificação de dados não lineares. Dados não lineares são dados que não podem ser separados de maneira que as classes fiquem cada uma de um lado do hiperplano como podemos ver na Figura 9. Nesses casos, o SVM usa um recurso chamado *kernel trick* (funções de kernel).

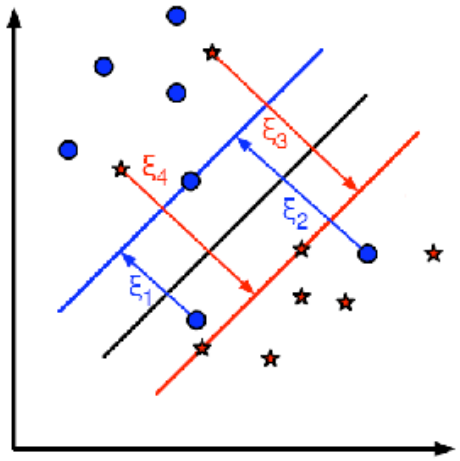


Figura 9: Hiperplano para dados não lineares
FONTE: ((f) Opencv Dev Team, 2015)

As funções de kernel têm a finalidade de projetar os vetores de características de entrada em um espaço de características de alta dimensão para classificação de problemas que se encontram em espaços não linearmente separáveis. A probabilidade do problema se tornar linearmente separável aumenta à medida que se aumenta o espaço da dimensão do problema. A Figura 10 mostra alguns dados não lineares e como eles podem se apresentar aumentando o espaço de dimensão do problema.

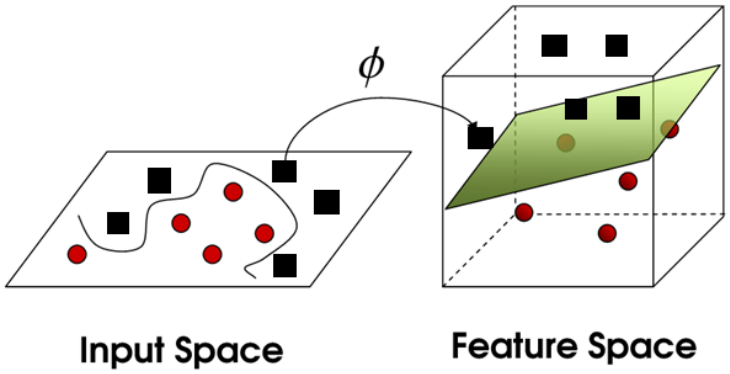


Figura 10: Representação dos dados em uma alta dimensão
Imagem a esquerda representa o espaço de entrada e a da direita novo espaço de características em uma alta dimensionalidade[20]

Entretanto, para obter uma boa distribuição para esse tipo de problema é necessário um conjunto de treinamento com um elevado número de instâncias (GONÇALVES, 2010).

A técnica de SVM foi investigada neste trabalho devido à sua popularidade, sendo experimentado com frequência nas pesquisas da área, inclusive para efeito de comparação.

3.3 Extreme Learning Machines

As Máquinas de Aprendizado Extremo (ELM, do inglês *Extreme Learning Machines*) são uma variação das *single-hidden layer feedforward networks* (SLFNs). Em redes do tipo *feedforward*, as informações são propagadas linearmente, não tendo nenhum tipo de ciclo interno ou realimentação da rede. Podemos ver na Figura 11 uma representação deste tipo de rede. A camada de entrada (primeira coluna) recebe as informações do ambiente externo e estas são passadas para a camada escondida ou intermediária (coluna do meio).

[20] http://www.imtech.res.in/raghava/rbpred/svm.jpg

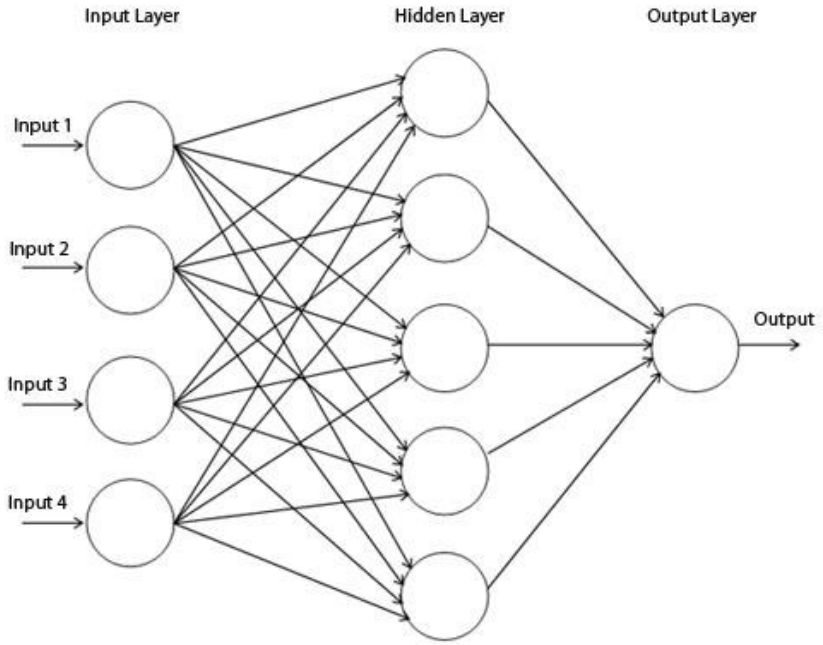


Figura 11: Rede neural do tipo SLFN - *Single Layer Feed Forward*[21]

Há três principais abordagens comumente usadas para treinar uma rede *feedforward*:

- Baseada em gradiente descendente, por exemplo, *backpropagation*. O gradiente é a primeira força dos pesos, no caso uma função linear de pesos. O algoritmo de declive descendente é baseado numa correção proporcional ao gradiente. Assim, a correção é proporcional a uma função linear de pesos. Tal abordagem só é passível de ocorrer em redes com retroalimentação.
- Baseada em métodos de otimizações padrões. É o caso do SVM para um tipo de SLFNs, chamadas de *Support Vector Network*.
- Baseada em técnicas de quadrados mínimos. Por exemplo, *radial basis function* (RBF) *network learning* (LOWE, 1989).

As redes do tipo SLFN têm como desvantagens: necessidade de definição dos valores de parâmetros ideais *a priori*; tempo de treinamento; ocorrência de parada em um mínimo local; e ocorrência de *overfitting* e *underfitting*. O termo *underfitting* ocorre quando uma rede não consegue realizar o mapeamento desejado por sua pequena

[21] http://www.codeproject.com/KB/dotnet/predictor/network.jpg

quantidade de neurônios na camada escondida ou porque o treinamento foi interrompido de forma prematura, não treinando o suficiente. Já o *overfitting* acontece quando a rede memoriza os padrões de treino e perde assim a capacidade de generalizar, deixando de prever corretamente as saídas relativas a pontos que não se encontram no conjunto de treino (ruídos). O *overfitting* pode ocorrer quando há um número excessivo de neurônios ou camadas para o problema que está sendo abordado.

A ELM, proposta por (HUANG, ZHU, & SIEW, 2006), consiste em uma rede *feedforward* com uma única camada oculta, na qual os pesos das entradas para os neurônios ocultos são escolhidos aleatoriamente, enquanto que apenas os pesos dos neurônios ocultos para a saída são determinados pelo método da matriz pseudo-inversa usando o conjunto de treinamento. Consequentemente, ELM oferece vantagens significativas tais como velocidade rápida de aprendizagem, facilidade de implementação, e menos interferência humana quando comparada a SLFNs mais tradicionais, tais como as redes *Multi-Layer Perceptron* (MLP) e RBF, já que não é preciso determinar *a priori* o valor de pesos ideais.

A técnica de ELM foi investigada neste trabalho devido ao seu rápido treinamento e por obter altas taxas de acerto, como visto em vários artigos (HUANG, 2014) (NETO, LUDERMIR, & ZANCHETTIN, 2015) que usaram esta rede.

Capítulo 4
Trabalhos Relacionados

Há muitas pesquisas na área de reconhecimento de gestos, baseadas no crescente poder computacional do *hardware* e/ou em várias técnicas como modelagem estatística, computação visual, reconhecimento de padrões, processamento de imagens e *soft computing* (MITRA & ACHARYA, 2003).

A maioria das pesquisas tem usado técnicas de modelagem estatística como análise de componentes principais (*Principal Component Analysis* – PCA), modelos escondidos de Markov (*Hidden Markov Models* - HMMs), sistemas avançados de filtragem de partículas (ARULAPALAM, MASKELL, GORDON, & CLAPP, 2001) e algoritmos condensadores (ISARD & BLAKE, 1998). HMMs têm sido usados extensivamente no reconhecimento de gestos (YAMATO, OHYA, & ISHII, 1992), utilizando programação dinâmica a partir de dados da amostragem da trajetória de um gesto. As máquinas de estado finito (*Finite State Machines* - FSM) também têm sido um método efetivo para modelar gestos humanos (DAVIS & SHAH, 2002).

Dentre os trabalhos após (GUNNAR, 1973), em que foi dito que os gestos humanos poderiam ser reconhecidos através da informação do movimento, perfis de movimento e suas trajetórias, pode-se citar (BOBICK & WILSON, 2002), que propuseram uma técnica baseada em estados para a representação e reconhecimento dos gestos. Em seu estudo, um gesto é definido como uma sequência de estados dentro de uma configuração de espaço.

Davis e Shah (2002) propuseram um método diferente usando uma máquina de estado finita para modelar qualitativamente quatro fases distintas de um gesto genérico. Na primeira fase, a mão deve estar na posição que dá início ao movimento (que no contexto do trabalho referenciado corresponde à mão com todos os dedos levantados). A segunda fase é a movimentação dos dedos até a posição final do sinal. A terceira fase é o repouso desta posição e por fim a mão volta para a posição da

primeira fase para simbolizar que o gesto acabou. A partir disto, utilizando uma luva, são detectadas as pontas dos dedos para enfim detectar o sinal que está sendo feito. Uma desvantagem desta abordagem é a utilização de uma luva para detecção das pontas dos dedos. A pesquisa proposta nesta dissertação procura não utilizar nenhum acessório para detecção da mão. A proposta de usar estados finitos que representam o começo e o fim de um sinal é bastante interessante quando analisando sinais em um vídeo. Isto pode reduzir o custo computacional já que não estaríamos analisando todos os quadros do vídeo, e sim, somente os quadros após o gesto de início até o gesto final. Esta abordagem pode ser adotada em um trabalho futuro que engloba gestos dinâmicos.

No trabalho de (MCKENNA & GONG, 1999), os gestos foram modelados os gestos como uma sequência de eventos visuais. Dadas trajetórias segmentadas temporalmente, cada segmento é associado a um evento e representado por um modelo probabilístico de trajetórias. Os autores usaram como reconhecimento a trajetória feita em um dado conjunto de dados relacionados a tempo e espaço e não o sinal propriamente dito. Esta abordagem pode trazer um problema entre reconhecimento de gestos que fazem ou têm uma trajetória parecida já que a mão não é analisada em detalhe. Os gestos reconhecidos no trabalho dos autores foram: apontar para esquerda, para a direita, acenar acima e acenar abaixo.

No trabalho de (ISARD & BLAKE, 1998), um algoritmo de condensação foi proposto como um meio probabilístico para monitorar curvas nas cenas visuais. Este estudo visa acompanhar um objeto em um ambiente com muita informação. Este mesmo estudo foi estendido por (BLACK & JEPSON, 1998) para reconhecer gestos e expressões faciais, modelando movimentos humanos como trajetórias temporais de alguns parâmetros estimados que descrevem o estado de um gesto em relação ao tempo. O intuito deste trabalho foi analisar a trajetória que a mão faz para reconhecer um sinal. Parecido com a proposta de (MCKENNA & GONG, 1999), este trabalho traz o mesmo problema entre reconhecimento de gestos que fazem ou têm uma trajetória parecida já que a mão não é analisada em detalhe.

Existem ainda alguns métodos de reconhecimento de gestos como *top-down* (O'ROURKE & BADLER, 1980) e *bottom-up* (OSCARSSON, 1982). O primeiro é baseado na reconstrução geométrica do corpo, enquanto o segundo se baseia na extração de características de baixo nível de imagens, como possíveis áreas que contêm uma pessoa (usadas neste caso para contar a quantidade de pedestres em

uma imagem) (YAMATO, OHYA, & ISHII, 1992). Muitos dos sistemas baseados no método *top-down* definem a reconstrução do corpo em cilindros e outros formatos chamados de "*super-quadrics*". Esta representação é tirada através de imagens onde cada parte do corpo simboliza um parâmetro do modelo. Porém, um dos problemas desta técnica se dá ao fato de uma imagem poder ter oclusões de alguma parte do corpo. Esta abordagem analisa todo o corpo identificando extremidades e junções. Esta abordagem é mais complexa pois requer um maior nível de detalhamento da imagem e as junções são muito próximas às extremidades, podendo haver problemas de oclusão e constantes mudanças de posição. As abordagens *top-down* e *bottom-up* não utilizam informações detalhadas da imagem e sim informações globais e no nosso trabalho, faz-se necessário o detalhamento da mão para diferenciação entre os sinais.

Para auxiliar na utilização de alguns dos artigos anteriormente citados, o reconhecimento de gestos requer o uso de algum dispositivo de captação de imagem ou uso de aparelhos que possam captar o movimento do corpo, a direção do olhar e expressões faciais, como luvas e/ou roupas especialmente adaptadas para este fim, e marcadores para mapeamento óptico. A Figura 12 mostra alguns tipos de roupa de captura de movimento usadas atualmente.

Como podemos observar na Figura 12, as roupas usam tecnologias diferentes como luzes, cores e sensores. Tanto a primeira luva (Figura 12 (a)) como a roupa do meio (Figura 12 (b)), utilizam a mesma tecnologia de sensores giroscópios espalhados pela peça. Cada sensor está localizado em uma junção do corpo e é ligado a um transmissor de rede sem fio. A roupa com refletores de luz infravermelha (Figura 12 (d)) utilizada pela *Virginia Commonwealth University*, é um equipamento mais caro pois utiliza várias câmeras, porém é mais preciso. Neste sistema são utilizadas dez câmeras que disparam luzes infravermelhas que ao serem refletidas pela roupa, são captadas pelas câmeras. A partir disto monta-se o esqueleto do corpo que é usado em um modelo 3D. A luva colorida (Figura 12 (c)) é usada por (WANG, 2008) para rastrear a mão com base na sua pose levando em consideração as cores. Este trabalho utiliza um banco de poses com mais de 18 mil posições de dedos e cada pose é uma imagem 40x40 pixels.

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

Figura 12: Exemplos de acessórios de captura de movimento

(a) Luva para captura de movimento[22]; (b) Luva para reconhecimento por cores[23] (WANG, 2008); (c) Roupa para captura de movimento[24]; (d) Roupa utilizando refletores de luz[25], créditos da foto: Ali Jones

A luva, ao ser capturada pelo sistema, passa por um processo de tratamento de imagens como mostra a Figura 13, até ficar com tamanho 40x40.

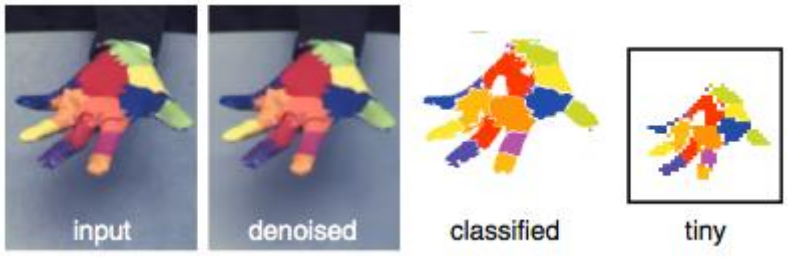


Figura 13: Tratamento das cores da luva

Fonte: (WANG, 2008)

Os autores utilizaram uma luva com dez cores, devido à câmera captar melhor essas cores mesmo com sombreamento. No primeiro passo, a imagem sofre um

[22] https://www.inition.co.uk/product/synertial-igs-glove/

[23] http://www.gizmag.com/mit-cheap-easy-gesture-recognition-system/15229/

[24] http://www.metamotion.com/gypsy/IGS-190-hybrid-motion-capture-system.html

[25] http://www.commonwealthtimes.org/2015/02/16/motion-capture-class-gets-a-taste-of-3d-animation-2/

processo de remoção de ruídos, suavizando-a utilizando um filtro bilateral. Em seguida, as cores da imagem são classificadas como sendo uma das dez cores presentes na luva ou não, utilizando um modelo de mistura gaussiana; por fim a imagem é redimensionada para 40 x 40. Esta última imagem é então utilizada para achar a imagem correspondente no banco utilizando o algoritmo de distância de *Hausdorff-like*.

A maioria das tecnologias apresentadas requer uma comunicação com um *software* de manipulação 3D como o *Autodesk Maya*[26], *Autodesk 3DS Max*[27], *Blender*[28], entre outros, que contêm um modelo que representa o tipo de roupa utilizado. Cada articulação do corpo que está sob a roupa é associada a um dispositivo, seja um giroscópio, uma luz ou até mesmo uma cor, que aciona o mesmo movimento no modelo que está sendo representado no *software* 3D. Isto dá um movimento fluido a um personagem, se assemelhando assim a um movimento mais natural de uma pessoa.

Tecnologias baseadas em sensores para reconhecimento de gestos variam de acordo com várias dimensões como: precisão, resolução, latência, abrangência do movimento, conforto do usuário e custo. No caso de sensores acoplados a luvas, o usuário normalmente precisa carregar um aparelho grande e muitos fios que o ligam a um computador. Isto diminui a facilidade e a naturalidade da interação do usuário. Técnicas baseadas em visão computacional, apesar de não terem que lidar com problemas de fios e carregamento de equipamentos pesados, podem ter problemas de oclusão de determinadas partes do corpo. Porém, tais técnicas também têm como vantagem a possibilidade de lidar com propriedades como textura e cor. A técnica de rastreamento pode detectar movimentos rápidos e sutis dos dedos enquanto a mão do usuário se move, enquanto que sistemas baseados em visão computacional vão dar, no melhor caso, uma ideia geral do tipo de movimento dos dedos (MITRA & ACHARYA, 2007).

Muitos fatores podem atrapalhar o reconhecimento de gestos como: variações nas execuções humanas de um mesmo gesto, tanto em relação ao tempo de execução quanto em relação à configuração espacial; erros de mapeamento;

---

[26] http://www.autodesk.com/products/maya/overview

[27] http://www.autodesk.com/products/3ds-max/overview

[28] https://www.blender.org/

variações na captura da imagem; frequência de amostragem, entre outros. Para realizar este reconhecimento existem diferentes metodologias de segmentação, extração de características e classificação. Muitos dos sistemas de reconhecimento precisam de um vasto trabalho de segmentação e alinhamento dos dados, e devem ser rápidos para suportar comportamento interativo (HONG, TURK, & HUANG, 2000).

Na etapa de segmentação, o computador deve captar uma sequência de imagens e reconhecer somente o objeto de interesse, ou seja, para o escopo deste trabalho, as mãos da pessoa (já que a expressão facial não será reconhecida). Para isto, deve-se usar um algoritmo para extrair o plano de fundo da imagem e identificar o objeto em questão. No caso das mãos da pessoa, pode-se usar o algoritmo de detecção de pele (*skin detection*). Como as pessoas têm tonalidades de pele diferentes, este algoritmo atribui a um pixel a cor 255 caso seja pele e 0 caso não seja pele, ou vice versa, binarizando a imagem e deixando-a em preto e branco.

Para a definição de quais pixels representam pele, há várias abordagens que levam em consideração o espaço de cor a ser analisado. Alguns adotam a utilização do formato *YCbCr*; outros o formato RGB (*Red*, *Green*, *Blue*), mais conhecido por ter uma melhor representação das cores utilizando apenas três cores (vermelho, verde e azul). O espaço de cor *YCbCr* representa a imagem em termos de luminância e a diferença entre os outros sinais em relação à luminância. O canal "*Y*" representa luminância (brilho da imagem); o canal "*Cb*" é a diferença entre o sinal azul e a luminância; e o "*Cr*" é a diferença entre o sinal vermelho e a luminância. O RGB é um dos espaços de cores mais utilizados para processamento e armazenamento de dados digitais de imagens (FIBIGER, 2004). Outros usam o RGB normalizado e por fim o modelo HSV (*Hue Saturation Value*).

Os trabalhos de (CAI & GOSHTASBY, 1999) e (OLIVEIRA & CONCI, 2009) utilizam o sistema de HSV. Ambos mencionam que a tonalidade da cor da pele varia entre 6 e 38 no canal H e entre 0.23 e 0.68 no canal S, porém eles só utilizam o canal H. Utilizando o espaço de cor *YCbCr*, o trabalho de (SORIA, SCHAEFFER, & VILLAREAL, 2013) afirma que o canal Y terá um limiar maior que 80, o canal *Cb* entre 85 e 135 e o canal Cr entre 135 e 180. Em qualquer dos espaços de cor adotado, esta abordagem faz uma análise pixel a pixel para transformar a imagem em preto e branco de acordo com o limiar adotado. Após o processo de binarização, são aplicadas algumas técnicas morfológicas para suavizar esta nova imagem. O processo descrito

anteriormente traz muito ruído na imagem e aplicando técnicas como dilatação, erosão e suavização, diminuem os ruídos indesejáveis.

No trabalho de (SORIA, SCHAEFFER, & VILLAREAL, 2013), a imagem binarizada é usada para calcular o "*convexhull*". Este processo consiste em achar o menor polígono que contém determinados pontos dentro de um espaço euclidiano; em outras palavras, é o formato que é formado quando se coloca um elástico em volta de um determinado objeto, como ilustrado na Figura 14 Figura 14.

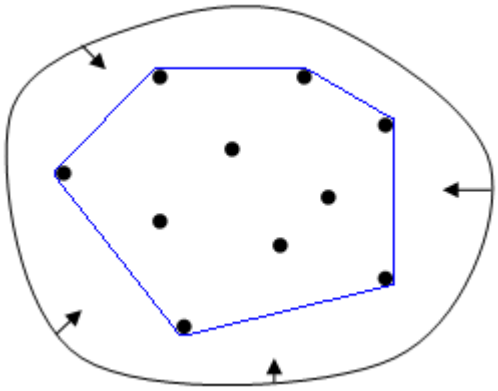


Figura 14: Ideia geral do *convexhull*

A Figura 15 mostra a imagem binarizada da mão sendo contornada pelo *convexhull* (linha exterior).

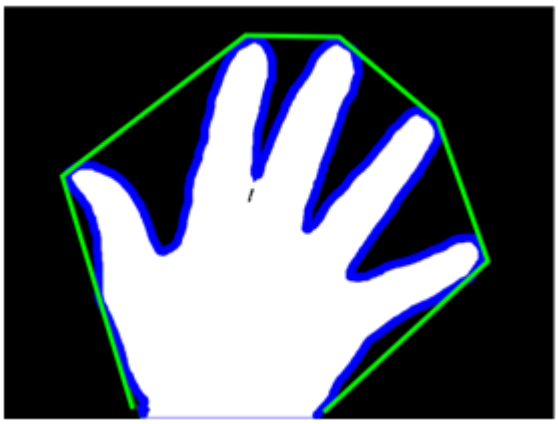


Figura 15: Determinação do *convexhull* na imagem binarizada
Fonte: (SORIA, SCHAEFFER, & VILLAREAL, 2013)

Após este processo, são computados os vértices do *convexhull* onde há uma mudança de direção, indicando assim o ponto da mudança (Figura 16). A partir destes vértices, é calculado o pixel do contorno da mão que maximiza a distância do vértice,

resultando assim nos pontos chamados de "*convexity defects*". O resultado é usado para indicar a quantidade de dedos que estão levantados.

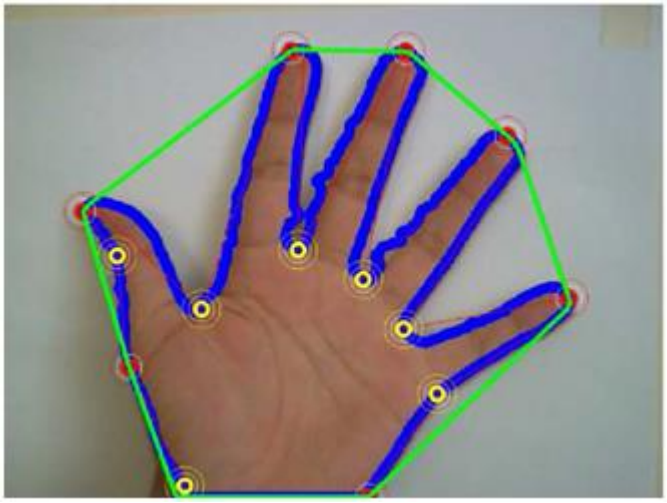


Figura 16: Computação dos vértices do *convexhull* (*convexity defects*)
Fonte: (SORIA, SCHAEFFER, & VILLAREAL, 2013)

O uso do *convexhull* dá informações a respeito do contorno da mão e nesta pesquisa, é utilizada toda informação da mão a fim de obter mais detalhes e detectar sinais que possam usar cruzamento dos dedos ou sobreposição das mãos.

Há outros estudos que englobam técnicas de visão computacional e redes neurais, como (NETO, LUDERMIR, & ZANCHETTIN, 2015). Nele, foram utilizadas técnicas de binarização e detecção de bordas que servem de entrada para uma rede neural do tipo ELM, obtendo uma classificação de 95% de acerto no reconhecimento de sinais em LIBRAS. A imagem da mão foi binarizada e foram utilizados os filtros de *canny* e *sobel* para extrair características como contorno da mão da imagem binarizada. Esses dois filtros são usados para detectar as bordas de uma imagem. Enquanto o filtro de *sobel* tem a propriedade de realçar linhas verticais e horizontais mais escuras que o fundo, calculando o gradiente de uma imagem, o filtro de *canny* utiliza um algoritmo de multi-estágios descrito na seção 6.1. Após o uso destes filtros, a imagem foi dividida em blocos uniformes, e para cada bloco, foi computada a quantidade de pontos brancos. A imagem foi então normalizada para se obter um percentual de pontos brancos por bloco que serve de entrada para a rede neural. O uso de ELM na nossa pesquisa foi motivado pelos bons resultados alcançados por este trabalho para reconhecer sinais em LIBRAS, porém com abordagens diferentes

para o tratamento de imagens e os dados de alimentação da rede como descrito no capítulo sobre a abordagem da pesquisa (Capítulo 4).

Outro trabalho, realizado por (JÚNIOR, BARROS, FERNANDES, BEZERRA, & FERNANDES, 2013), propôs a predição de gestos incompletos. Desta forma, o sistema pode prever um gesto mesmo que ele não tenha sido totalmente capturado. O sistema funciona primeiramente extraindo um vetor de características de cada quadro através de um módulo de extração de características. Cada novo vetor de características serve como entrada para um módulo de predição de gestos que a partir disto prevê um possível gesto. Foram usadas duas técnicas no módulo de predição: *Hidden Markov Model* (HMM), que aprende gestos completos e reconhece gestos incompletos, e *Dynamic Time Wraping* (DTW) para calcular as distâncias entre os gestos de entrada e achar o sinal previsto. O sistema de predição que utiliza HMM utiliza o clusterização *K-means* para achar a melhor aproximação inicial e é treinado utilizando o algoritmo de *Baum-Welch*. A Figura 17 mostra a arquitetura deste sistema utilizando HMM.

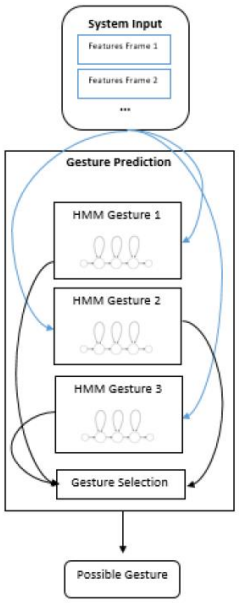


Figura 17: Arquitetura HMM de predição
Fonte: (JÚNIOR, BARROS, FERNANDES, BEZERRA, & FERNANDES, 2013)

O sistema de predição que utiliza DTW usa um conjunto de exemplos para cada gesto para compor a representação completa do gesto. Ele calcula a distância entre cada entrada e o conjunto de exemplos, e o gesto que tiver a menor distância é tido como o gesto final. A Figura 18 mostra a arquitetura deste sistema utilizando DTW.

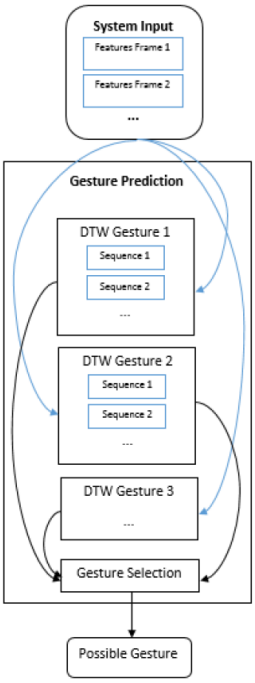


Figura 18: Arquitetura usando DTW
Fonte: (JÚNIOR, BARROS, FERNANDES, BEZERRA, & FERNANDES, 2013)

O módulo de extração de características foi proposto pelos mesmos autores em um outro artigo (BARROS, et al., 2013), chamado de *Convexity local contour sequences for gesture recognition* (CLCS). Este módulo consiste em binarizar a imagem utilizando o filtro de Otsu, filtro de média para suavizar, aplicar filtro de erosão e subtrair da imagem binarizada originalmente encontrada pelo filtro de *Otsu*. Pontos de interesse que satisfazem uma determinada tolerância são achados aplicando o algoritmo de Douglas-Peucker e são em seguida conectados fechando o contorno da mão, para depois serem achados outros pontos mais significativos dentro desta área utilizando o algoritmo de Sklansky. Tal pesquisa é muito interessante em prever qual gesto está sendo feito a partir de um gesto incompleto, porém vai além do contexto do nosso trabalho já que lida com gestos dinâmicos e incompletos, e pode ser usada como referência para futuro suporte a gestos dinâmicos.

O trabalho de (BARROS, et al., 2013) procurou diminuir a dimensionalidade das características encontradas pelos métodos de *Speed Up Robust Feature* (SURF) (GUPTA & MA, 2001) e *Local Contour Sequence* (LCS) (BAY, TUYTELLAARS, & GOOL, 2008) utilizando uma abordagem chamada de *Convexity approach. Convexity approach* recebe como entrada os pontos que representam a posição da mão achados pelos algoritmos de SURF e LCS. A primeira parte utiliza o algoritmo de *Douglas-Peucker* para criar um polígono que contorna a posição da mão usando os pontos

mais externos detectados. Na segunda parte, o *convex hull* é encontrado, e utilizando o algoritmo de *Sklankys* os pontos mais expressivos são achados baseados na distância entre pontos. Esta abordagem trouxe uma taxa de 91% tanto para SURF quanto para LCS quando utilizando o classificador HMM. Utilizando uma rede neural recorrente de *Elman,* foi obtida uma taxa de 90% utilizando LCS e 92% para SURF. A Figura 19 ilustra as etapas de execução desta abordagem.

Figura 19: Fluxo de execução da *Convexity Approach*

Utilizando o algoritmo *Convexity Invariant Position Based on Ransac* (CIPBR) para extração de características, o trabalho de (SANTOS D. G., NETO, FERNANDES, & BEZERRA, 2014) utiliza dois classificadores: o DTW e o HMM, ambos vistos no trabalho citado anteriormente. O intuito deste algoritmo CIPBR é reduzir a posição da mão nas imagens em em dois conjuntos de informações. Este algoritmo é composto de quatro módulos: o primeiro módulo realiza a binarização de uma imagem utilizando o filtro de *Otsu* (NOBUYUKI, 1979)*;* o segundo módulo consiste em achar a linha de pulso utilizando uma regressão linear simples, traçar o contorno da mão usando o *Hu Invariant moments.* O terceiro módulo traça uma circunferência utilizando esta distância. Por último, os pontos de assinatura do CIPBR são encontrados calculando o *convex hull* utilizando o algoritmo de *Andrew´s monotone chain* (DAY, 1988)*.* Utilizando o classificador HMM (CIPBR-HMM), foi obtida uma taxa de 94% e com o DTW (CIPBR-DTW), uma taxa de 99%. A Figura 20 mostra o fluxo entre os módulos utilizados no algoritmo de CIPBR.

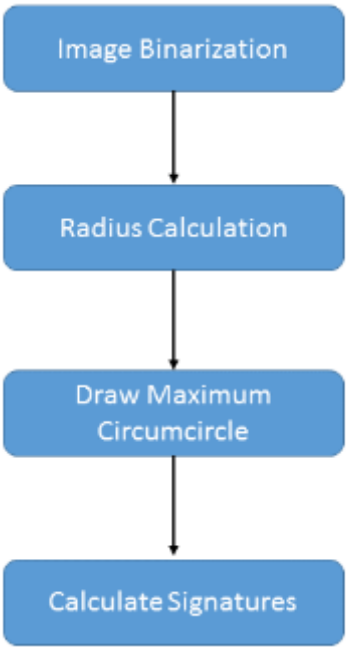


Figura 20: Os quatro módulos usado no CIPBR
(SANTOS D. G., NETO, FERNANDES, & BEZERRA, 2014)

Por fim, o trabalho de (SANTOS D. G., NETO, FERNANDES, & BEZERRA, 2015) utiliza o mesmo algoritmo de extração de características acima descrito (CIPBR) e seleciona o melhor tamanho para os vetores de características usando a função de *fitness* baseada na função de distância euclidiana para reconhecer gestos dinâmicos. Os vetores de características são reduzidos usando o algoritmo *selector*. A escolha do tamanho dos vetores se dá através do *Binary Particle Swarm Optimization* (BPSO). BPSO otimiza um problema tendo um conjunto de soluções canditatas chamados de partículas que são usados para explorar um espaço de solução para achar a melhor solução. Cada partícula tem sua posição e velocidade, e a função de *fitness* avalia o quão melhor cada partícula apresenta em cada iteração. Este método tem o nome *Binary* devido ao fato de que as posições e velocidades são representadas com 0s e 1s. Quando uma partícula se move e sua posição tem um melhor índice de *fitness* que anterior, sua posição é salva. A função de distância euclidiana mede a distância de cada partícula para a mesma partícula em todos os vetores binários para o mesmo gesto. A redução de cada vetor se dá através da normalização do CIPBR. Este trabalho obteve uma taxa de 99% de acerto.

Os trabalhos de (BARROS, et al., 2013), (SANTOS D. G., NETO, FERNANDES, & BEZERRA, 2014) e (SANTOS D. G., NETO, FERNANDES, &

BEZERRA, 2015) são trabalhos que apresentaram uma ótima taxa de acerto na classificação juntamente com técnicas de classicação interessantes que devem ser estudadas mais a fundo futuramente, porém todos são utilizados para classificar sinais dinâmicos e imagens com plano de fundo homogêneos. O presente trabalho procura apenas classificar sinais estáticos conforme já explicado anteriormente.

Os trabalhos aqui apresentados possuem abordagens interessantes, porém o contexto desta pesquisa difere das demais por estar tratando com planos de fundo tanto homogêneos como complexos e utilizando exemplos de contextos reais de uso de um celular convencional. Alguns dos trabalhos citados simulam um ambiente de desenvolvimento com restrições similares a um celular, com pouca memória e processamento. Este trabalho optou por utilizar um celular de fato, para ser mais fiel ao contexto de uso do sistema proposto e avaliar o seu comportamento concorrendo com outros processos que utilizam os mesmos recursos de memória, espaço interno e processamento.

Na forma de lidar com uma imagem, este trabalho ao invés de utilizar imagens binarizadas ou com informações do contorno da mão, procura utilizar toda a informação dos pixels da mão a fim de usar o máximo de detalhe para identificar os vários tipos de sinais que compõem as linguagens de sinais, pois alguns deles precisam de informações frontais da mão (informações que seriam perdidas na binarização ou utilizando somente seu contorno).

Capítulo 5
Abordagem

Esta dissertação propõe um sistema de reconhecimento e tradução entre sinais de linguagens usadas por deficientes auditivos, com o objetivo de auxiliá-los a se comunicarem com pessoas que usem uma linguagem de sinais de outro país. O sistema é composto por um aplicativo móvel (que se comunica com um servidor *web*) e uma interface *web*.

5.1 Visão Geral do Sistema Proposto

Para se comunicar com uma pessoa que fala outra língua de sinais, primeiramente, o usuário do sistema proposto deve, no seu aplicativo móvel, escolher duas linguagens: a sua língua (língua mãe) e a língua para a qual os sinais devem ser traduzidos (língua estrangeira).

Utilizando recursos do celular, a câmera será o meio principal para viabilizar a tradução. O usuário deve apontar a câmera traseira do celular para a pessoa que irá se comunicar com ele (seu interlocutor, que se comunica em uma língua de sinais diferente da sua). A câmera captura os gestos da pessoa e traduz para a linguagem do usuário. Considerando um cenário ideal, onde ambos tenham o sistema instalado, a tradução será feita da mesma forma para a outra pessoa.

Para realizar esta tradução o sistema analisa cada quadro do vídeo que está sendo captado e procura classificar este quadro de acordo com os sinais previamente cadastrados. Uma questão que surge é: como popular o sistema com todos os sinais das diversas linguagens do mundo para viabilizar esta tradução entre duas linguagens quaisquer de maneira que o usuário não dependa de uma atualização na base de sinais?

A linguagem de LIBRAS ou qualquer outra linguagem de sinais, é baseada na posição da mão. Do ponto de vista computacional, cada sinal é representado por uma imagem (no caso de um sinal estático) ou um conjunto de imagens (no caso de um sinal dinâmico). É muito complicado saber de todos os sinais existentes, popular o sistema e disponibilizá-los para a sociedade. Isto representaria um grande trabalho que consumiria bastante tempo e não necessariamente levaria a um resultado satisfatório para todos. Além disto, podemos ter um sinal que não esteja com classificação satisfatória (ou seja a tradução não está sendo feita corretamente) e bastaria que novas imagens fossem adicionadas neste novo modelo para obter um melhor resultado.

Pensando nisto, foi desenvolvida uma página web disponibilizada para o usuário que mostra os sinais, organizados por país, e as suas imagens já existentes.

Figura 21: Tela da página web listando um país e seus gestos

A Figura 21 mostra a página web que lista os sinais cadastrados para um determinado país, podendo o usuário ainda selecionar outros países e criar novos sinais. Na Figura 22, o usuário pode ver as imagens associadas a um sinal no banco de sinais no servidor.

Figura 22: Imagens associadas a um sinal

Novo sinais podem ser cadastrados pelo usuário e todos os sinais podem ser atualizados, adicionando novas imagens a fim de melhorar a sua classificação.

Assim, o sistema proposto funciona em duas plataformas: web e móvel. De maneira geral, um usuário de posse do aplicativo móvel pode acessar a página web, fazer seu *login*, e ter acesso aos sinais já existentes e às imagens que estão sendo usadas para cada sinal. Ainda nesta página, ele pode criar novos sinais, adicionar novos países e navegar entre os sinais. No aplicativo, o usuário pode escolher a sua linguagem de sinais materna e uma linguagem a ser traduzida, conforme explicado anteriormente. Para isso, o aplicativo se comunica com o servidor para a realização do reconhecimento e tradução. A Figura 23 mostra a tela de seleção entre duas linguagens no aplicativo móvel.

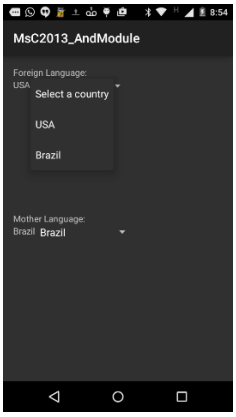


Figura 23: Seleção dos países para fazer a tradução entre as linguagens de sinais

5.2 Módulos

Como descrito na seção anterior, o sistema possui dois módulos principais: servidor e móvel. O módulo servidor é subdividido em cinco submódulos para facilitar o seu desenvolvimento, que foi realizado nas linguagens C# e C++, e para disponibilizar o *website* foi utilizado o *Internet Information Service*[29] (IIS) da Microsoft. A organização dos módulos é mostrada na Figura 24. Esta seção detalha cada módulo.

[29] https://www.iis.net/

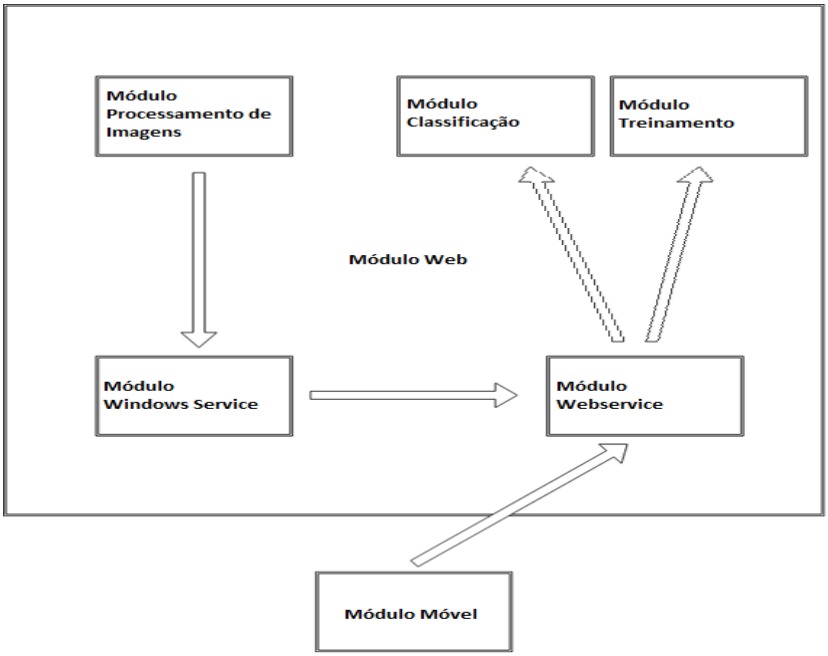


Figura 24: Organização dos módulos

5.2.1 Módulo Servidor / WEB

No módulo servidor, são armazenadas todas as imagens que são submetidas pelos usuários como sendo representações de um sinal, assim como o arquivo de mapeamento que é gerado na criação de um sinal. Ele contém ainda a página web e o servidor IIS que disponibiliza esta página na Internet. O servidor também contém os arquivos relacionados ao treinamento e classificação de sinais e tratamento das imagens.

Os arquivos de mapeamento e treinamento, necessários para classificação e tradução, são disponibilizados para *download* pelo aplicativo móvel através deste módulo, pelo *webservice*. O módulo servidor é dividido em: *web*, *webservice*, *Windows service*, processamento de imagens, treinamento e classificação.

5.2.1.1 Módulo Web

O módulo *web* disponibiliza uma interface para o usuário que mostra os sinais cadastrados e as imagens correspondentes a cada sinal, organizadas em pastas por país (Figura 25 e Figura 26).

Figura 25: Interface *web* com os sinais organizados por país

Figura 26: Interface *web* mostrando todas as imagens de um mesmo sinal

A interface *web* permite a criação de novos sinais e países, e o *upload* de novas imagens para um determinado sinal. Quando um novo sinal é criado, é criada também uma pasta com seu nome, dentro da pasta do país correspondente, no servidor. O sinal recebe também um índice, utilizado para atualizar o arquivo de mapeamento, que associa os nomes das pastas dos sinais aos seus índices. Cada imagem submetida ao sistema será salva dentro da pasta do sinal correspondente. Na etapa de tradução, o sistema, ao capturar uma imagem da câmera e classificá-la como sendo de um determinado sinal já existente, retorna o índice da pasta do sinal. O arquivo de mapeamento é então consultado para que seja retornado o nome do sinal e seja exibida a imagem correspondente na língua do usuário. No caso de haver diferentes imagens associadas ao mesmo sinal, é retornada a primeira da lista. A Figura 27 e Figura 28 mostram a criação de um novo sinal e a tela inicial de um sinal assim que foi criado.

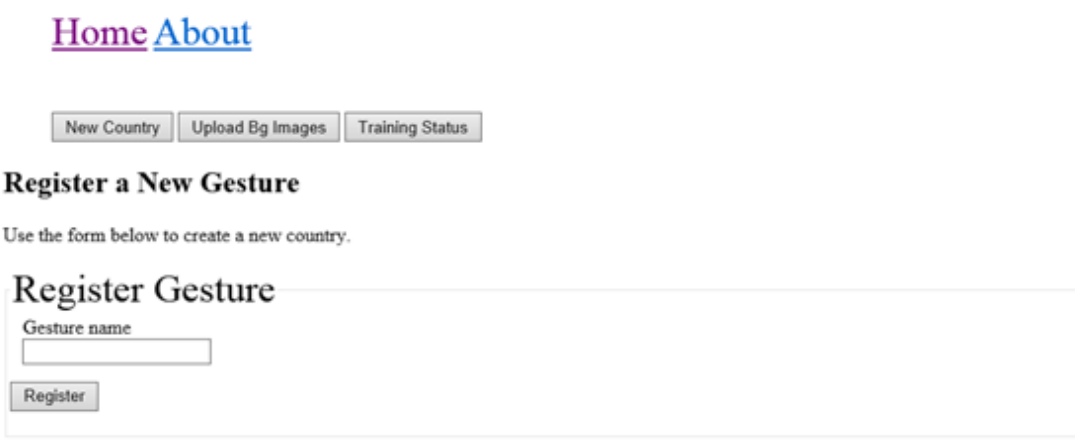


Figura 27: Criação de um sinal na interface *web*

Figura 28: Sinal sem nenhuma imagem

5.2.1.2 Módulo Webservice

O módulo *webservice* consiste em um serviço para informar ao módulo *web* se a base de imagens de um determinado país está em treinamento ou não, e chamar o módulo treinamento quando houver uma mudança no sistema de pastas (monitorado pelo módulo *Windows service*).

5.2.1.3 Módulo Processamento de Imagens

Este módulo faz o tratamento das imagens enviadas pelos usuários, para que sejam usadas no treinamento. Este tratamento consiste na remoção do plano de fundo (conforme será explicado na seção 5.3.3).

5.2.1.4 Módulo Treinamento

O módulo de treinamento é responsável por criar um arquivo de treinamento do classificador para cada país. Para construir o arquivo de treinamento, este módulo usa o arquivo de mapeamento gerado pelo módulo *web*, para pegar o caminho de cada sinal no sistema de arquivos do servidor. De posse do caminho, ele transforma cada imagem em um vetor e salva uma tupla (índice, vetor) no arquivo de treinamento. Este arquivo é utilizado para treinar a rede. O método de treinamento é descrito na seção 5.3.4.

5.2.1.5 Módulo Classificação

Este módulo é responsável por realizar a classificação ao receber as imagens capturadas pelo celular. A classificação ocorre analisando 10 imagens (quadros de um vídeo) capturadas sequencialmente sem nenhum intervalo entre elas, e o resultado final da classificação é o sinal (classe) mais frequente.

5.2.2 Módulo Móvel

Este módulo foi desenvolvido na plataforma Android utilizando OpenCV, e se comunica com o módulo servidor através do *webservice* para solicitar a tradução. Como o treinamento e a classificação são feitos no servidor, o aplicativo é basicamente responsável pela interação com o usuário, captura e exibição das imagens dos sinais no processo de tradução. O tamanho das imagens capturadas no dispositivo móvel pode variar de dispositivo para dispositivo. Para fins deste trabalho, a resolução foi de 480 de largura por 640 de altura. Procurou-se utilizar uma iluminação constante, porém com planos de fundo diferentes, homogêneo para o servidor e complexo para o dispositivo móvel. Mais detalhes do dispositivo utilizado são descritos na seção 6.1 na página 68.

5.3 Fluxo Geral

A seção anterior detalhou a função de cada módulo que compõe o sistema proposto. Resumidamente, a interação entre os módulos no servidor ocorre da

seguinte forma: quando o módulo *Windows service* detecta que houve alguma mudança no sistema de arquivos, que no caso seria a adição de uma nova imagem para um determinado sinal (após ser tratada pelo módulo de processamento de imagens), ou remoção de uma imagem (que fosse julgada inadequada por um usuário, por exemplo), ele chama o módulo *webservice* informando qual pasta foi modificada. Por sua vez, o *webservice* chama o módulo de treinamento informando que país ele deve treinar e altera o *status* do país em questão para “em treinamento”, mudando este status quando o treinamento acaba. Quando o módulo *web* lista os países, ele verifica que determinado país está em treinamento e mostra o *status* corrente para o usuário. Ao final do treinamento, um arquivo é gerado com todas as informações das imagens, a ser usado na classificação.

A interação entre servidor e celular ocorre por meio do *webservice*. O *webservice* lista para o aplicativo os países existentes para a tradução e disponibiliza os arquivos de mapeamento para serem baixados. Estes arquivos são baixados assim que o usuário seleciona os países cujas linguagens ele deseja usar, no processo de tradução.

Como já foi mencionado, uma das propostas deste trabalho é dar controle aos usuários para que eles possam manipular os sinais, ou seja, criar, visualizar e excluir, estabelecendo assim um banco de sinais criado colaborativamente. No sistema proposto, uma ação gera um encadeamento de outras ações cujo objetivo maior é de viabilizar a tradução de sinais. Por exemplo, quando o usuário cria um sinal, não apenas é adicionado o nome do sinal ao banco de dados, mas é também criado um sistema de arquivos para aquele sinal, que será usado para o treinamento. Já o envio de uma imagem para um determinado sinal já cadastrado, não só representa a adição de uma imagem na base, mas também inicia todo o processo de treinamento. A remoção segue um processo semelhante. Os fluxos da criação de sinais, do envio de novas imagens e da tradução em si são detalhados a seguir.

5.3.1 Criação de um Sinal

O primeiro encadeamento acontece quando um sinal é criado. Na criação de um sinal, é criado também um sistema de pastas para conter o nome do sinal e uma pasta de imagens processadas. A raiz da pasta do sinal contém o arquivo de

mapeamento, descrito anteriormente, e os arquivos executáveis para o processamento de imagens e treinamento da rede neural. A pasta de “imagens processadas” é populada depois que o módulo de processamento de imagens é concluído.

O usuário pode ainda criar um país - afinal todo sinal obrigatoriamente pertence a um país - mas esta parte não encadeia nenhuma sequência de ações, trata-se apenas de uma hierarquia na organização dos sinais.

5.3.2 Submissão de uma Imagem

Esta ação ocorre quando o usuário submete uma nova imagem para um sinal de gestos. Nesta parte, o módulo de processamento de imagens é acionado. Ao ser completado, o resultado do tratamento da imagem é mostrado e o usuário pode aceitar ou não aquela imagem para ser incluída no treinamento (Figura 29). Caso o usuário aceite, a imagem é movida para a pasta de imagens processadas e inicia-se então o treinamento acionado pelo módulo *webservice*. Caso contrário, a imagem é descartada e o usuário pode submeter uma nova imagem.

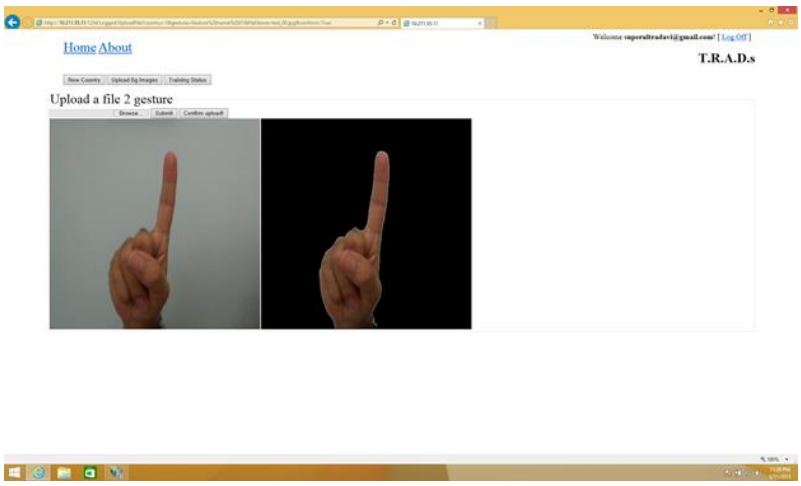


Figura 29: Tratamento de imagem realizado após o envio de uma imagem para o servidor

Todo este encadeamento resulta em um conjunto de arquivos que ficam disponíveis na pasta do sinal e são disponibilizados para *download* através do módulo *webservice*. Qualquer alteração no sistema de arquivos de um sinal, seja adição ou remoção, resulta em um novo treinamento.

Várias técnicas foram testadas para atingir o objetivo desta pesquisa de reconhecer e traduzir gestos. Para tratar as imagens enviadas pelos usuários, inicialmente procuramos usar a técnica de binarização baseada em detecção de pele, porém essa técnica não permite reconhecer gestos que sejam feitos com os dedos cruzados e/ou sobreposição de mãos e dedos, nem distinguir entre o dorso e a palma da mão. Além disso, foi percebido que qualquer mudança no plano de fundo interferia no resultado. Foi então desenvolvida uma combinação de técnicas de visão computacional utilizando máscaras de filtro, assumindo-se que as imagens teriam um fundo homogêneo. Dessa forma, obtivemos um resultado capaz de destacar somente a mão na imagem, deixando todo o *background* da imagem preto. Com isto podemos ter somente a visualização da mão nas imagens enviadas pelos usuários.

5.3.3 Tratamento de Imagem

Um desafio para o reconhecimento de gestos neste trabalho diz respeito ao plano de fundo das imagens dos sinais. As imagens recebidas dos usuários, que são usadas no treinamento do classificador, necessitam de um tratamento para remover o plano de fundo e deixar a mão em destaque. Duas abordagens foram experimentadas para realizar esta separação: detecção de pele e uma combinação de métodos de visão computacional utilizando máscaras de filtro.

5.3.3.1 Detecção de Pele

A técnica de detecção de pele testada foi baseada no trabalho de (OLIVEIRA & CONCI, 2009), conforme explicado na seção 3.1, página 26. Essa técnica é baseada em binarização da imagem, na tentativa de remover o plano de fundo e destacar apenas a imagem da mão. Porém, o resultado obtido para o propósito deste trabalho não foi suficiente para separar a mão do plano de fundo. Mesmo ajustando os limiares do canal H, os resultados tiveram muitos pixels que não eram pele classificados assim como ruído na imagem, o que dificulta a detecção da mão. Para obter um resultado que possa ser aplicado no reconhecimento de gestos seria necessário definir um novo limiar para cada foto utilizada, necessitando assim de muitos ajustes manuais. Os melhores resultados obtidos através desta técnica, depois de algumas tentativas para achar o melhor limiar, são apresentados na Figura 30.

Figura 30: Resultados obtidos para detectar pele para fins deste trabalho

Além disso, alguns sinais de LIBRAS podem utilizar tanto a palma como o dorso da mão, cruzar os dedos e até mesmo sobrepor as mãos uma na outra, o que dificultaria o reconhecimento da área da mão na imagem binarizada. Podemos tomar como exemplo a letra “R” que possui o dedo indicador sobrepondo o dedo médio; a letra I usando a palma da mão; e a letra J usando o dorso da mão. A Figura 31 mostra as suas imagens binarizadas.

R I J

Figura 31: Binarização de dedos cruzados, dorso e palma da mão
Utilização de dedos cruzados na imagem R; Utilização da palma da mão na imagem I; Utilização do dorso da mão na imagem J

Os dedos cruzados ao se juntarem podem levar ao reconhecimento como um só dedo, consequentemente podendo haver uma confusão entre sinais. Da mesma forma no uso da palma e do dorso da mão, é impossível saber qual lado está sendo usado.

5.3.3.2 Remoção do Plano de Fundo

Face aos problemas enfrentados com a técnica de detecção de pele, optou-se por uma abordagem que combina alguns métodos de visão computacional utilizando máscaras de filtro com o objetivo de separar a mão do plano de fundo da imagem. Podemos ver o fluxo dos algoritmos de processamento de imagens utilizados na remoção de plano de fundo na Figura 32. Esta seção descreve cada um desses passos.

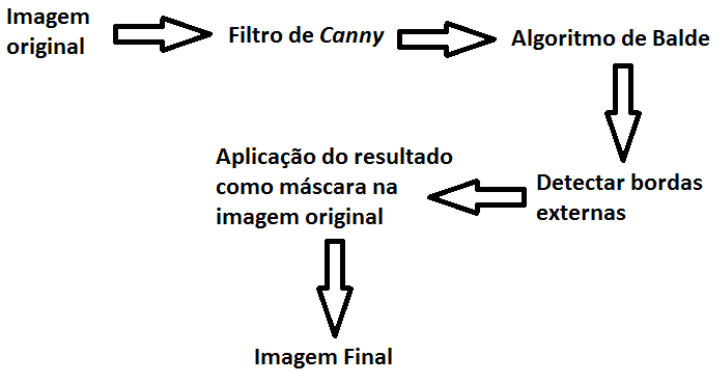


Figura 32: Fluxo do processamento de imagem utilizado

Esse processo foi feito considerando-se dois cenários: servidor e móvel. É no servidor que é realizado o treinamento do classificador. As imagens enviadas através da interface web, especialmente para o treinamento, devem ter um plano de fundo homogêneo, luminosidade constante e pouca sombra para conseguirmos uma remoção completa do plano de fundo. No caso de um plano de fundo que não seja homogêneo, os resultados não serão ideais. Entende-se como resultado ideal a remoção do plano de fundo pintando-o de preto, dando assim destaque à mão. Este cenário configura-se como um ambiente controlado, em que podem ser utilizadas imagens capturadas em situações ideais. Por outro lado, o módulo móvel está inserido em um contexto em que o usuário pode usar o aplicativo em qualquer lugar para realizar uma tradução, e consequentemente é mais difícil que as imagens capturadas possuam as características ideais de plano de fundo e iluminação. O processo de remoção do plano de fundo realizado no servidor é descrito a seguir.

O primeiro tratamento é detectar os contornos da mão. Para isto o filtro de *Canny* ((b) OpenCV Dev Team, 2015), descrito na seção 3.1, página 26, foi utilizado. Dentre as duas formas propostas para encontrar os limiares máximo e mínimo - tentativa e erro e média dos pixels da imagem - foi adotada a primeira forma para achar os valores dos limiares devido aos resultados obtidos utilizando a fórmula da média dos pixels terem apresentado muitos ruídos. A Figura 33 mostra o resultado da detecção de contornos da mão utilizando o filtro de *Canny* na imagem original.

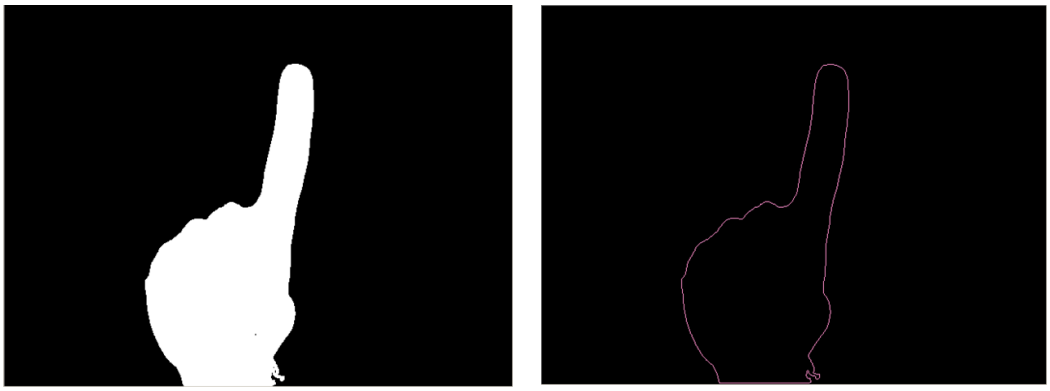
(a) (b)

Figura 33: Resultado do filtro de *Canny*

Imagem original (a) e detecção do contorno da mão após aplicação do filtro de *Canny* (b)

O algoritmo de balde ((e) Opencv Dev Team, 2015) é utilizado logo em seguida para preencher a área delimitada pelo contorno detectado. O algoritmo de balde irá

preencher todos os pixels de uma determinada região com a cor desejada (Figura 34 (a)). Com isto, obtemos uma máscara que vai servir de molde para ser retirada da imagem original (Figura 34 (b)).



(a) (b)

Figura 34: Resultado do algoritmo de balde

Imagem após a aplicação do algoritmo de balde (a) e detecção das bordas (b)

A máscara obtida (Figura 34 (b)) é utilizada na imagem original para retirar o plano de fundo, deixando apenas a mão em destaque. A Figura 35 mostra um exemplo da aplicação da máscara em alguns sinais.

Figura 35: Resultado final após a aplicação da máscara na imagem original

Esta abordagem foi então portada para o módulo móvel, porém o resultado não foi satisfatório como apresentado na seção de resultados.

No módulo móvel, por estarmos lidando com ambientes não controlados, já que o usuário pode usar o sistema na rua, ou em qualquer outro ambiente onde queira comunicar-se com outra pessoa, a abordagem ocorre de uma forma diferente. Foi utilizado o algoritmo *BackgroundSubtractorMOG2* ((a) Opencv Dev Team, 2015) do *OpenCV* para remoção de plano de fundo.

BackgroundSubtractorMOG2 é um algoritmo de segmentação baseado em misturas gaussianas (ZIVKOVIC, Improved Adaptive Gaussian Mixture Model for

Background Subtraction, 2004), (ZIVKOVIC & HEIJDEN, 2006), que usa de 3 a 5 distribuições gaussianas para cada pixel do plano de fundo. Os pesos resultantes da distribuição gaussiana indicam se o pixel faz parte do plano de fundo ou não (dependendo da duração de tempo de cada cor de pixel na cena). Em uma sequência de quadros de entrada que representam um vídeo, os pixels de plano de fundo são mais estáticos e permanecem com a mesma cor na imagem por mais tempo. Este algoritmo provê uma melhor adaptabilidade em variações de cena devidas a iluminação (MORDVINTSEV & ABID, 2014). A Figura 36 mostra o resultado deste algoritmo no celular.

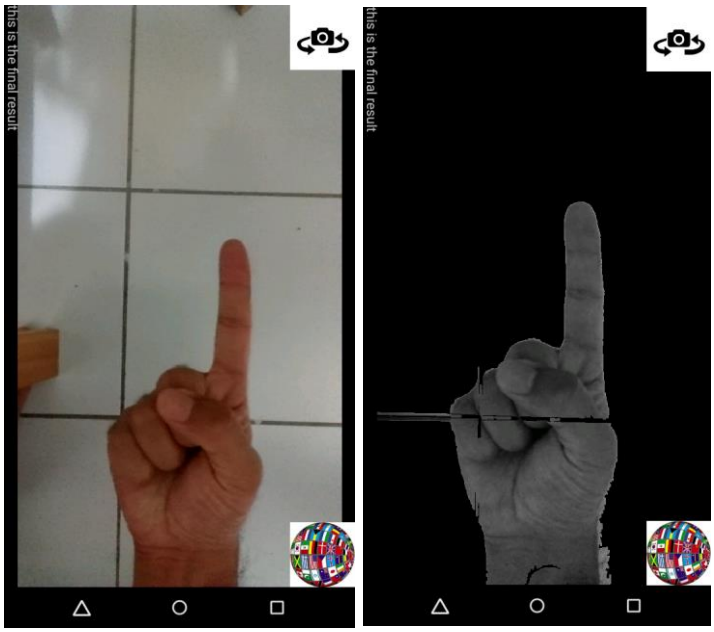


Figura 36: Resultado do *BackgroundSubtractorMOG2* no celular

5.3.4 Treinamento e Classificação

A etapa de tratamento das imagens provê as entradas para o treinamento e ajuste dos classificadores. A cada nova imagem tratada, o sistema realiza um novo treinamento dos classificadores para adicionar o novo conhecimento. Quando o aplicativo móvel é utilizado, o sistema precisa classificar as imagens capturadas pela câmera, com base nesse treinamento prévio.

5.3.4.1 Classificador *Cascade*

A primeira abordagem adotada para a classificação de um sinal em LIBRAS foi utilizar o algoritmo de classificação *Cascade*. Este classificador é normalmente utilizado para detecção de objetos e a ideia foi testar como ele iria se comportar neste contexto de reconhecimento de sinais ((c) Opencv Dev Team, 2015). Esta abordagem utiliza imagens positivas e negativas no treinamento do classificador. As imagens positivas contêm apenas o objeto que estamos querendo encontrar e as imagens negativas podem ser qualquer imagem desde que não contenham o objeto pretendido.

A base de treinamento consistiu de 5 sinais, cada sinal com aproximadamente 200 imagens. As imagens foram capturadas de forma sequencial sem nenhum intervalo entre cada quadro do vídeo capturado pela câmera do celular e tratadas para retirar o plano de fundo conforme descrito anteriormente.

Para utilizar este algoritmo, foi necessário criar duas pastas: uma com todas as imagens positivas e outro com as imagens negativas. As imagens negativas usadas neste teste consistiram apenas do plano de fundo do ambiente sem a presença da mão. Dois arquivos foram criados, um para as imagens positivas e outro para as imagens negativas, contendo todos os caminhos destas imagens. O *OpenCV* cria um arquivo *.vec* que contém as imagens positivas e é usado para treinar o classificador. Ao executar o comando abaixo, deve ser passado como argumento o arquivo das imagens positivas citado anteriormente:

- *opencv_createsamples -info [arquivoImagensPos] -vec posHands*

Podemos verificar o conteúdo deste arquivo executando o seguinte comando:

- *opencv_createsamples -vec posHands*

Isto feito, podemos treinar o classificador utilizando o comando abaixo:

- *opencv_traincascade -data [pastaDestino] -vec [arquivo .vec] -bg [arquivoImagensNeg] -featureType [Haar/LBP]*

O tipo de treinamento pode ser Haar ou LBP (*Local Binary Patterns*) ((c) Opencv Dev Team, 2015). A diferença entre estes dois tipos de treinamento se dá através dos cálculos realizados. O treinamento do tipo *Haar* realiza estes cálculos utilizando números decimais, enquanto que o treinamento do tipo LBP utiliza apenas números inteiros. A base de cálculo deles os diferencia e faz com que o LBP seja mais rápido

do que o *Haar* porém menos preciso, enquanto o Haar por ser mais demorado, tem uma melhor precisão. A parte de teste foi executada 3 vezes, cada tentativa com novas imagens capturadas, porém utilizando a mesma base. Porém, nas 3 tentativas e ao testarmos os dois tipos de treinamento, na prática não foi notada muita diferença, ambos ficaram abaixo dos 10% de acerto, mas a diferença no tempo de treinamento foi significante. Em uma base com aproximadamente 1000 imagens positivas e 500 negativas, o LBP levou algumas horas enquanto o *Haar* levou dias e apresentou alguns problemas como falta de memória, já que o arquivo .xml que estava sendo gerado era maior que 1GB.

Além da baixa taxa de acerto, um mesmo sinal apresentava resultados diferentes na classificação de acordo com a posição da mão em relação à câmera. Por exemplo, acontecia a seguinte situação: o usuário faz o sinal “1”; o sinal é reconhecido pelo sistema; o usuário, ainda fazendo o sinal “1”, move a mão de forma que o dedo coincide com a suposta posição do outro dedo que formaria o sinal “2”; o sistema reconhece o sinal “2”, apesar da mão estar fazendo o sinal “1”, e vice-versa. Assim, seria suficiente a pessoa “achar a posição certa” na câmera, fazendo qualquer sinal, para obter um determinado resultado, o que leva a resultados inconsistentes. Devido a esses resultados, esta abordagem foi descartada.

5.3.4.2 SVM

Partindo para uma abordagem de classificação utilizando inteligência artificial, primeiramente experimentou-se usar SVM. O módulo de treinamento foi desenvolvido usando OpenCV na linguagem C++. O treinamento do SVM consiste na utilização de imagens positivas e negativas, com um rótulo correspondendo a cada uma das imagens. Assim como no classificador *Cascade*, as imagens positivas são imagens que contêm apenas o objeto que estamos querendo classificar, e as imagens negativas são quaisquer imagens desde que não contenham o objeto pretendido, ou seja, podem ser qualquer imagem de plano de fundo.

O SVM experimentado neste trabalho é dividido em três módulos: captura das imagens, treinamento e classificação. A seção 5.3.3.2, página 58, discutiu como o tratamento das imagens foi feito para elas serem usadas no treinamento. A imagem binarizada não foi usada por haver gestos que podem usar a palma da mão ou uma parte do corpo, como o sinal ocupado ou esperar (Figura 37).

Figura 37: Cruzamento de mãos que pode dificultar a binarização

A abordagem do SVM neste trabalho inicia-se com o algoritmo percorrendo todas as pastas mapeadas que estão no arquivo de mapeamento. Cada imagem contida nas pastas corresponde a uma linha em uma matriz, que é usada para o treinamento. Ao mesmo tempo que esta matriz é preenchida, uma outra matriz de rótulos é preenchida com os índices das pastas correspondentes aos sinais. Estas matrizes são então usadas no treinamento.

Por se tratar de um problema multiclasse, o tipo de SVM utilizado foi *SVM::C_SVC* e os tipos de *kernel* experimentados foram: linear (*LINEAR*), polinomial (*POLY*), radial (*RBF*) e sigmoide, na tentativa de encontrar qual poderia ter a melhor taxa de classificação.

Por o SVM ser um classificador de duas classes foram realizados dois tipos de experimentos: utilizando o método de treinamento um contra todos, e treinamento de duas em duas classes.

No primeiro experimento, a cada rodada do treinamento, é atribuído o rótulo 1 a uma única letra, e o rótulo 0 às demais. Em seguida, cada imagem do conjunto de teste é classificada como 1 (ou seja, a letra que foi treinada) ou 0 (ou seja, qualquer outra letra da base). A classificação indica se cada imagem do conjunto de teste é da letra treinada (1) ou não (0). O resultado é dado de acordo com o rótulo que aparece com maior frequência para o conjunto de teste. Por exemplo, para a letra “A”, a rede é primeiramente treinada com A possuindo o rótulo 1, e as demais letras possuindo o rótulo 0. Em seguida, é dado como entrada um conjunto de teste com 10 imagens da letra A, e a rede retorna 1 (ou seja, a imagem é da letra A) ou 0 (ou seja, a imagem não é da letra A) para cada imagem. O procedimento é repetido para todas as letras

da base e a letra que apresentar o rótulo 1 com maior frequência é o resultado final da classificação.

Já no segundo experimento, com treinamento de duas em duas classes, cada letra é treinada contra uma única outra letra por vez. Ou seja, o treinamento para a letra A é feito entre A (rótulo 1) e B (rótulo 0), entre A (rótulo 1) e C (rótulo 0), e assim sucessivamente. O conjunto de teste será então classificado de acordo com cada conjunto de treinamento. Por exemplo, um conjunto de teste com imagens da letra A será classificado de acordo com os conjuntos de treinamento A com B, A com C e assim por diante. A primeira classificação indicará se cada imagem de teste é da letra A (rótulo 1) ou B (rótulo 0), e da mesma forma para todos os outros conjuntos de treinamento. O resultado final será a letra classificada com maior frequência.

Estes experimentos, com entradas conhecidas, foram de caráter investigativo, a fim de verificar as taxas de acerto do SVM. O segundo método foi então utilizado com entradas desconhecidas, sendo classificadas considerando todos os arranjos das letras da base, duas a duas. Os resultados desses três experimentos são apresentados no Capítulo 7.

5.3.4.3 ELM

O algoritmo ELM se desenvolve em duas etapas: treinamento e classificação. Os algoritmos de treinamento e classificação utilizados neste trabalho foram baseados no artigo de (HUANG, ZHOU, DING, & ZHANG, 2012) e foram desenvolvidos inicialmente em Java, e migrados para *MatLab*, conforme explicado nesta seção.

A etapa de treinamento da ELM consiste em transformar todas as imagens em um vetor e criar uma matriz de treinamento. Esta etapa gera um arquivo que contém as informações de: quantidade de neurônios, tipo de função da camada intermediária, quantidade de neurônios de saída juntamente com seus rótulos correspondentes, e por fim as matrizes de *bias* e pesos de entrada e saída.

A etapa de classificação consiste em achar o maior valor na tabela de neurônios de saída. Cada linha nesta tabela corresponde a uma entrada na rede neural e cada coluna é um neurônio de saída da rede. Cada neurônio tem o seu valor de ativação para cada entrada e o neurônio com maior valor de ativação é o resultado final para uma entrada. Para calcular a matriz de resultados, os dados de teste são multiplicados pelos pesos da entrada e somados com a matriz de *bias*, calculadas na etapa de

treinamento. A função de ativação é então calculada dando um resultado entre 0 e 1 para então ser multiplicada pela matriz de pesos de saída.

Os parâmetros a serem ajustados em redes ELM são as funções de ativação e a quantidade de neurônios na camada intermediária, já que os pesos da camada escondida são dados aleatoriamente e os pesos da camada de saída são determinados pela matriz pseudo-inversa usando o conjunto de treinamento. A função de ativação escolhida foi do tipo sigmoide, devido a sua popularidade e facilidade computacional. Como a ideia inicial era de realizar o treinamento no servidor e a classificação no próprio celular, pois isso permitiria que o usuário pudesse solicitar a tradução entre linguagens de sinais sem a necessidade de uma conexão com o servidor (que seria necessária apenas para baixar o arquivo de treinamento ou cadastrar novas imagens), a escolha da quantidade de neurônios na camada intermediária levou em conta aspectos como: o tamanho do arquivo gerado na etapa de treinamento (que seria utilizado no dispositivo móvel), tempo de download, leitura das matrizes e armazenamento das mesmas no dispositivo móvel. Por isso, a quantidade de neurônios na camada intermediária foi inicialmente de 50, uma quantidade muito inferior em relação ao artigo de (NETO, LUDERMIR, & ZANCHETTIN, 2015), por exemplo.

O tamanho do arquivo .xml resultante do treinamento, gerado nesta configuração, foi de 45MB e o tempo de leitura e carregamento das matrizes no celular foi em torno de 2 a 3 minutos. Porém, a taxa de classificação foi de 0% de acerto. Com isto, a quantidade de neurônios teve que ser aumentada para a quantidade usada no artigo de (NETO, LUDERMIR, & ZANCHETTIN, 2015), de 5000 neurônios. A quantidade de neurônios na camada intermediária aumenta o tamanho do arquivo .xml gerado pelo treinamento, já que aumenta o tamanho das matrizes. Esta quantidade de neurônios gerou um arquivo de 4GB e matrizes muito grandes, inviabilizando o download deste arquivo e o armazenamento das matrizes no celular (para realizar a classificação) por falta de memória. O celular usado neste teste foi o modelo de entrada do Motorola Moto E[30], que possui um processador dual core de 1.7 e 1 GB de memória. Dispositivos Android (Android, 2015) têm um limite de uso da memória RAM por aplicativo a fim de manter o sistema funcional *multi-threaded*

[30] http://www.gsmarena.com/motorola_moto_e_dual_sim-6323.php

(StackOverflow, 2015). Este limite varia de acordo com a quantidade de memória, mas no geral, varia entre 16 e 68 MB e ao atingir este limite, o aplicativo é fechado com o erro de *OutOfMemory Exception*, que acontecia quando o aplicativo carregava as matrizes. Com o constante uso do sistema, futuramente a quantidade de neurônios pode a vir diminuir, viabilizando a classificação *off-line*.

Assim, a primeira versão da ELM, que foi implementada na linguagem Java para Android, utilizando uma biblioteca externa para manipulação de matrizes chamada de *Matrix Toolkits For Java*[31], precisou ser migrada para *MatLab*[32], uma ferramenta de cálculo mais robusta que traz facilidades para operações com matrizes. Além disso, a classificação passou a ser feita no servidor, e não mais no celular.

5.3.5 Tradução

Para realizar a tradução, o usuário deve primeiramente selecionar dois países diferentes no aplicativo móvel (e consequentemente a língua mãe e a língua estrangeira). Em seguida, o aplicativo captura, através da câmera, as imagens do sinal que o interlocutor faz (na língua estrangeira). O aplicativo, ao capturar cada imagem, salva em um arquivo os valores dos pixels da imagem em um vetor e envia este arquivo para o servidor para dar início à classificação. A tradução é feita com o resultado da classificação e apresentada ao usuário.

---

[31] http://mvnrepository.com/artifact/com.googlecode.matrix-toolkits-java/mtj/1.0.1

[32] http://www.mathworks.com/products/matlab/

Capítulo 6
Métodos

Esta seção detalha a operacionalização dos experimentos, incluindo a base de dados construída e utilizada e os dispositivos de hardware empregados.

6.1 Base de Dados dos Sinais

Para a realização do treinamento da rede, procuramos bases de dados de sinais em LIBRAS. Porém, as poucas que existem não possuem sinais suficientes ou possuem imagens binarizadas. Devido à forma como as imagens são tratadas neste trabalho, as bases de dados binarizadas não poderiam ser usadas pois o valor da informação dos pixels, são 0 ou 1 (255) e neste trabalho, os valores dos pixels variam de 0 a 255. Portanto, criamos uma base inicial para treinamento. Conforme dito anteriormente, este trabalho trata apenas imagens estáticas.

A base criada consiste de imagens de sinais estáticos do alfabeto LIBRAS que passam pelo tratamento de imagens descrito na seção 5.3.3, página 56, para remoção do plano de fundo antes de serem usadas para treinamento. Há aproximadamente 30 imagens de 150 x 150 pixels para cada sinal. O tamanho 150 x 150 pixels foi escolhido inicialmente sem a preocupação com a razão de aspecto (*aspect ratio*) da imagem após o tratamento para remoção do plano de fundo, mas levando em consideração o custo computacional ao analisar todos os pixels de uma imagem. A razão de aspecto é a proporcionalidade entre a largura e altura de uma imagem.

Porém, alguns problemas foram encontrados durante o desenvolvimento desta base. Inicialmente, foram utilizadas imagens das mãos de quatro pessoas para popular a base com aproximadamente 50 imagens por sinal. A base foi criada com imagens com a resolução nativa da câmera do celular utilizado para tirar as fotos (2448 x 3264 pixels), redimensionadas para 150 x 150 pixels - para diminuir o custo

computacional. Por exemplo, a Figura 38 mostra o sinal da letra G quando redimensionado. Pode-se perceber que a forma da mão não sofre tanta alteração em suas proporções.

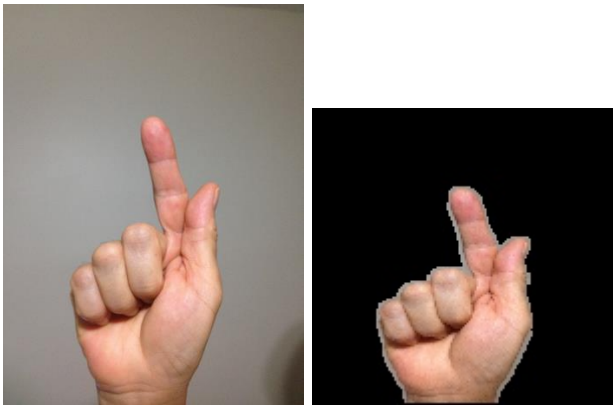


Figura 38: Imagem 2448 x 3264 pixels redimensionada para 150 x 150 pixels

Porém, durante a etapa de classificação, utilizando OpenCV para capturar as imagens da câmera, foi verificado que o tamanho máximo que o OpenCV salva uma imagem não é o mesmo tamanho de uma imagem capturada pela câmera. Por exemplo, no computador de mesa, a maior imagem que foi salva foi de 480 x 640 pixels, enquanto a resolução nativa da câmera é de 1280 x 720 pixels. No celular, a imagem salva pelo OpenCV tem o mesmo tamanho da tela do dispositivo. Neste trabalho, a tela do dispositivo utilizado durante a classificação tinha tamanho de 480 x 864 pixels apesar da resolução da câmera ser de 1440 x 2560 pixels.

Como há vários tipos de dispositivos com os mais variados tamanhos de tela, e como foi visto que o *OpenCV* salva as imagens com o tamanho da tela do dispositivo, o redimensionamento, para manter a razão de aspecto, iria variar de dispositivo para dispositivo. Tomando como exemplo as imagens utilizadas para compor a base de dados e redimensionando-as a fim de manter a razão de aspecto, a imagem 2448 x 3264 (tamanho salvo pela câmera do celular), para se adequar à janela escolhida (150 x 150) passaria a ter um tamanho de 112 x 200, e a imagem 480 x 864 (tamanho salvo pelo OpenCV no celular utilizado nesta pesquisa) teria um tamanho de 83 x 270. Alguns celulares possuem tela de 1920 x 1080 pixels, o que resultaria em uma imagem de 266 x 84. Estas diferenças de tamanho iriam inviabilizar a classificação, já que ao utilizar um classificador para imagens, elas devem ter o mesmo tamanho.

Assim, as imagens capturadas para classificação tinham tamanho original de 480 x 640 pixels e foram redimensionadas para 150 x 150, sem manter a razão de

aspecto. Por exemplo, a Figura 39 mostra o sinal da letra D, inicialmente capturado no tamanho 480 x 864 e redimensionado para 150 x 150 pixels. Percebe-se que neste caso, o redimensionamento causa uma deformação bastante acentuada no formato da mão.

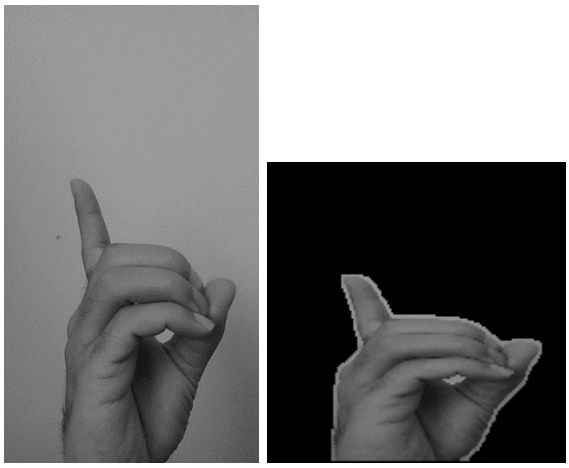
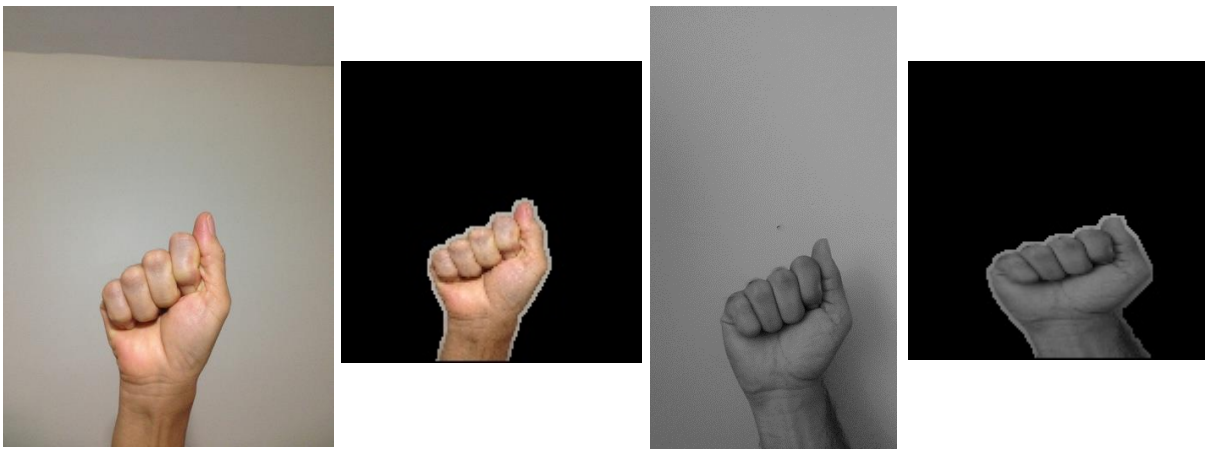


Figura 39: Imagem 480 x 864 pixels redimensionada para 150 x 150 pixels

Na Figura 40, analisamos imagens do sinal da letra “A” advindas das duas condições de redimensionamento (criação da base para treinamento e classificação). Podemos observar que a diferença entre as imagens (a) e (b) é muito pouca, a imagem ficou apenas levemente deformada, enquanto que em (c) e (d) notamos que a imagem sofre uma deformação considerável. Utilizar imagens do tipo (b) para treinamento, e do tipo (d) para classificação, apesar da semelhança, inviabilizou o reconhecimento dos sinais, pois para o classificador são imagens de sinais diferentes.



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

Figura 40: Comparação entre redimensionamentos

Por esta razão, a base de sinais foi refeita utilizando imagens do tamanho da tela do celular (480 x 864 pixels) e redimensionando-as para o tamanho 150 x 150, sem manter a razão de aspecto, obtendo assim uma melhora significativa na classificação conforme veremos na seção seguinte.

6.2 Dispositivos Utilizados

A implementação deste trabalho foi realizada utilizando *OpenCV*[33], uma ferramenta muito poderosa vastamente usada na área de visão computacional e que pode ser usada em várias linguagens e plataformas diferentes (Windows[34], Android[35], Maemo[36], FreeBSD[37], OpenBSD[38], iOS[39], BlackBerry[40], Linux[41] e OS X[42]). Algumas características de *OpenCV* foram fundamentais para o desenvolvimento deste trabalho, tais como: módulos de Processamento de Imagens e Vídeo I/O; recursos de álgebra linear; e mais de 350 algoritmos de visão computacional (filtros de imagem, calibração de câmera, reconhecimento de objetos, análise estrutural).

O dispositivo móvel utilizado para testes foi um celular Android Moto E[43] que possui um processador dual core de 1.7 GHz e 1 GB de memória. Este celular é um modelo de entrada da Motorola não possuindo recursos computacionais avançados, mas para fins deste trabalho, foi um recurso importante que permitiu avaliar o comportamento de aparelhos deste tipo diante do processamento envolvido.

O servidor foi um Intel i7-860 quad-core de 2.8 GHz com 4 GB de memória. O sistema operacional foi o Windows 8.1 e o servidor web foi IIS já instalado no

---

[33] http://opencv.org/

[34] https://www.microsoft.com/en-us/windows/

[35] https://www.android.com/intl/en_us/

[36] http://maemo.org/

[37] https://www.freebsd.org/

[38] http://www.openbsd.org/

[39] https://www.apple.com/ios/

[40] http://us.blackberry.com/

[41] http://www.ubuntu.com/

[42] https://www.apple.com/osx/

[43] http://www.gsmarena.com/motorola_moto_e_dual_sim-6323.php

Windows. O servidor não precisa ser uma máquina de última geração. Os componentes utilizados são de 2009 possuindo um preço acessível hoje em dia.

Capítulo 7
Resultados

Os resultados obtidos neste trabalho estão divididos em duas partes: remoção do plano de fundo e classificação dos sinais.

7.1 Remoção do Plano de Fundo

Conforme descrito anteriormente, a remoção do plano de fundo ocorre em dois momentos distintos: o primeiro quando o usuário faz o *upload* de uma imagem no servidor para ser adicionada à base de um determinado sinal; e o segundo no celular ao capturar as imagens da câmera para classificação. Foi adotado o seguinte critério para avaliar o resultado da remoção de forma subjetiva:

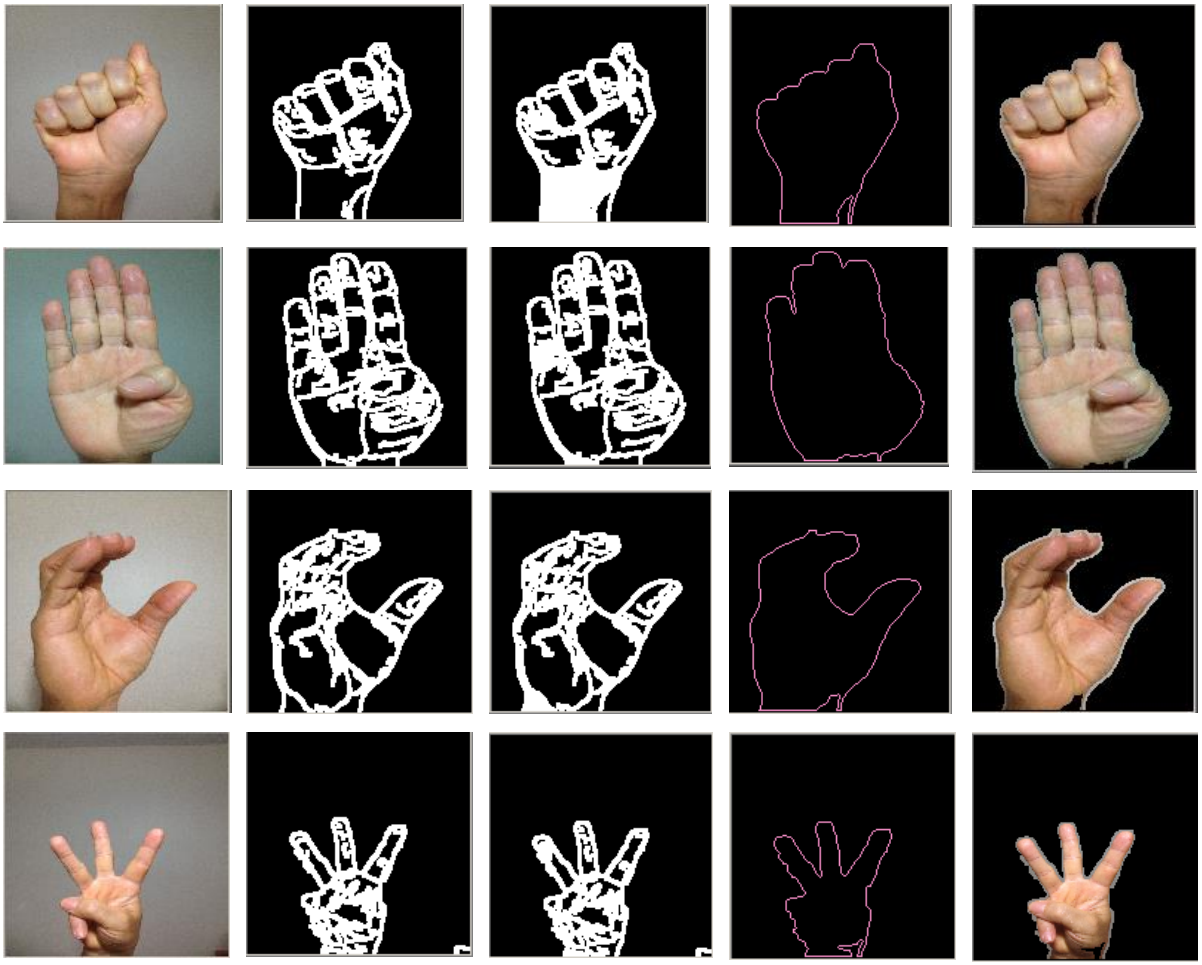
- OK - Plano de fundo removido com sucesso deixando somente a mão em destaque;
- Parcial - Plano de fundo removido, porém mão não está completa por apresentar áreas pretas como se fossem parte do plano de fundo;
- Erro - Resquícios do plano de fundo ainda presentes.

A Tabela 1 apresenta os resultados da remoção no contexto do *upload* para o servidor. Por se tratar de um ambiente controlado, os resultados apresentam uma alta incidência da remoção completa do plano de fundo e correta identificação da mão. Para fins dos experimentos deste trabalho, foi utilizada apenas uma parte do alfabeto de sinais (de A – L) a fim de demonstrar a viabilidade do sistema.

	OK	Parcial	Erro	Taxa de Remoção
A	39	1	0	97,7% (39/40)
B	37	3	0	92,5% (37/40)
C	30	2	8	75%(30/40)
D	35	5	0	87,5% (35/40)
E	38	2	0	95% (38/40)
F	36	1	3	90% (36/40)
G	35	3	2	87,5% (35/40)
H	43	1	0	97,7% (43/44)
I	43	3	1	91,4% (43/47)
J	39	0	1	97,7% (39/40)
K	55	2	0	96,4% (55/57)
L	39	1	0	97,7% (39/40)

Tabela 1: Resultados da remoção do plano de fundo

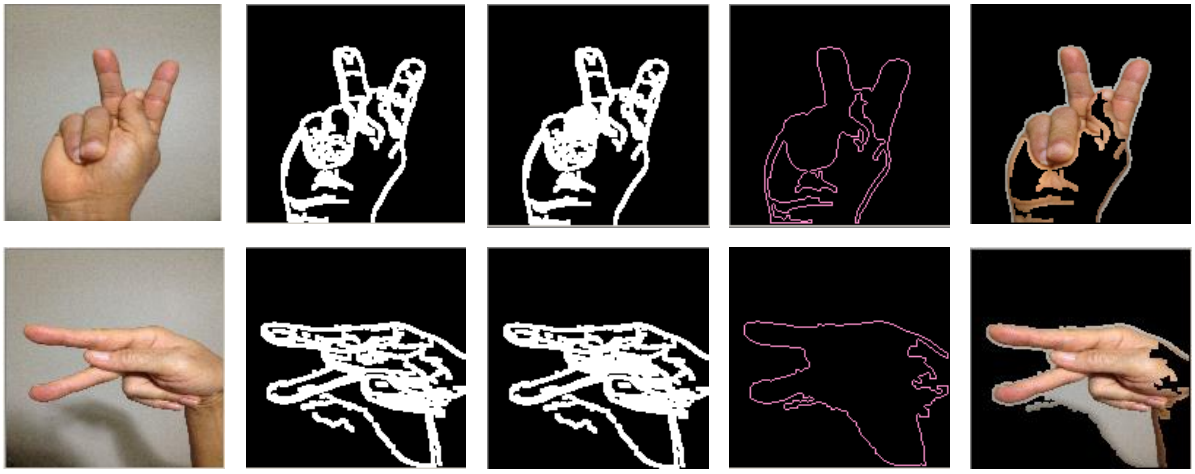
A Figura 41 demonstra as etapas pelas quais uma imagem passa a partir do momento em que um usuário faz o envio de imagem ao servidor, até o resultado final usando o algoritmo de remoção de plano de fundo apresentado na seção 5.3.3.2, página 58, para algumas letras do alfabeto LIBRAS.



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

Figura 41: Processo de remoção do plano de fundo
(a) Imagem original; (b) Imagem após o filtro de C*anny*; (c) Aplicação do algoritmo de balde; (d) Detecção da maior borda; (e) Resultado final

Algumas imagens não tiveram o mesmo resultado, como podemos ver na Figura 42. Este resultado não esperado pode ser explicado pelo funcionamento de preenchimento dos espaços pretos adotado. O algoritmo tenta preencher de branco os espaços pretos que estão entre dois pontos brancos. Para não percorrer a imagem toda, apenas uma região da imagem é percorrida e ao encontrar um pixel preto, aplica-se o algoritmo de balde. Podemos observar que há muitos contornos internos que fecham uma área. Isto prejudica o preenchimento do balde pois acaba preenchendo apenas uma pequena área como podemos ver nas duas imagens da Figura 42 (c). Trata-se de uma limitação de algoritmo de balde quando aplicado neste contexto, pois ele foi pensado em um caso em que não há contornos internos. Não podemos alterar a janela de detecção de borda somente para a área interna da mão - mudar os parâmetros de detecção influencia todas as bordas.



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

Figura 42: Resultados não esperados

Por exemplo, vejamos o caso específico da letra “Y” cujo sinal em LIBRAS foi testado com duas pessoas como mostra a Figura 43. Pequenas variações nas posições dos dedos e luminosidade levaram a resultados muito diferentes. Nas imagens da parte inferior da Figura 43, o algoritmo identificou o primeiro pixel preto após um pixel branco (correspondente ao final de dedo mínimo), salvando assim sua posição. Ao continuar percorrendo a região entre o dedo mínimo e o médio, ele identificou um pixel branco (correspondente ao início do dedo médio) dando a ideia de que o pixel preto achado anteriormente, estaria dentro da região da mão, pintando assim todos os pixels pretos conectados a ele de branco. Isto provocou o efeito visto na terceira imagem da parte inferior da Figura 43 em que o plano de fundo foi todo preenchido de branco.

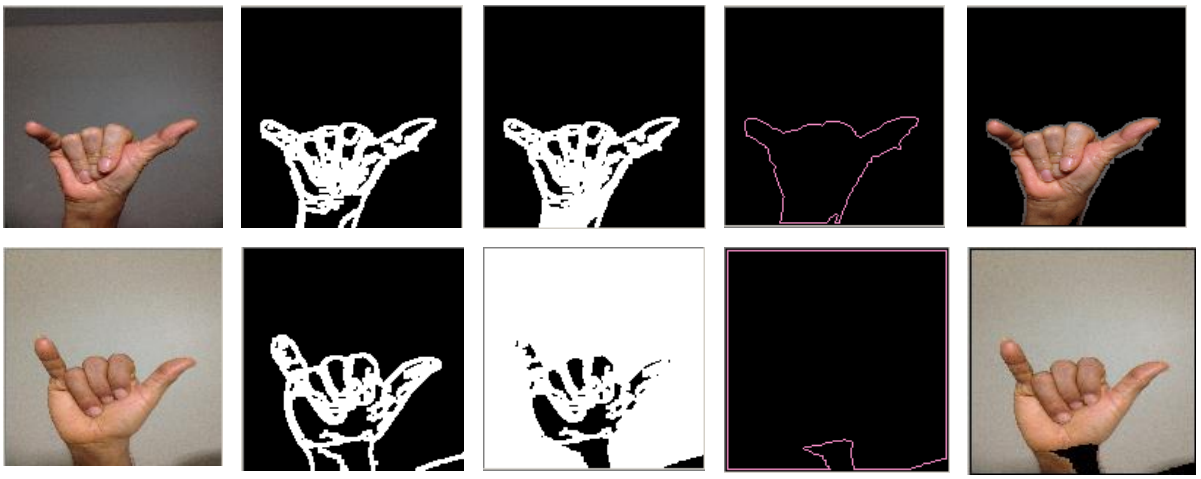


Figura 43: Comparação entre variações do sinal da letra Y

Já no dispositivo móvel, a maioria das imagens obteve uma remoção parcial do plano de fundo. O mesmo algoritmo de remoção de plano de fundo, quando executado no dispositivo móvel, apresentou uma maior sensibilidade, apresentando assim muito ruído no resultado final. Nos casos de identificação parcial da mão e de erro, pode ter havido variações na iluminação do ambiente, presença de sombra e dificuldades de distinção entre cores pelo sensor da câmera do celular (em caso de planos de fundo de cor parecida com a cor da mão) e/ou instabilidade no manuseio do dispositivo durante a captura.

Os resquícios do plano de fundo nas imagens são ruídos que podem prejudicar a classificação. Por estar usando o mesmo algoritmo, as imagens no celular precisam ter um fundo homogêneo e serem estáveis, algo complicado de obter neste tipo de dispositivo. Podemos perceber que mesmo com um fundo homogêneo e o celular relativamente estável, não foi possível uma imagem da mão inteira constante (Figura 44).

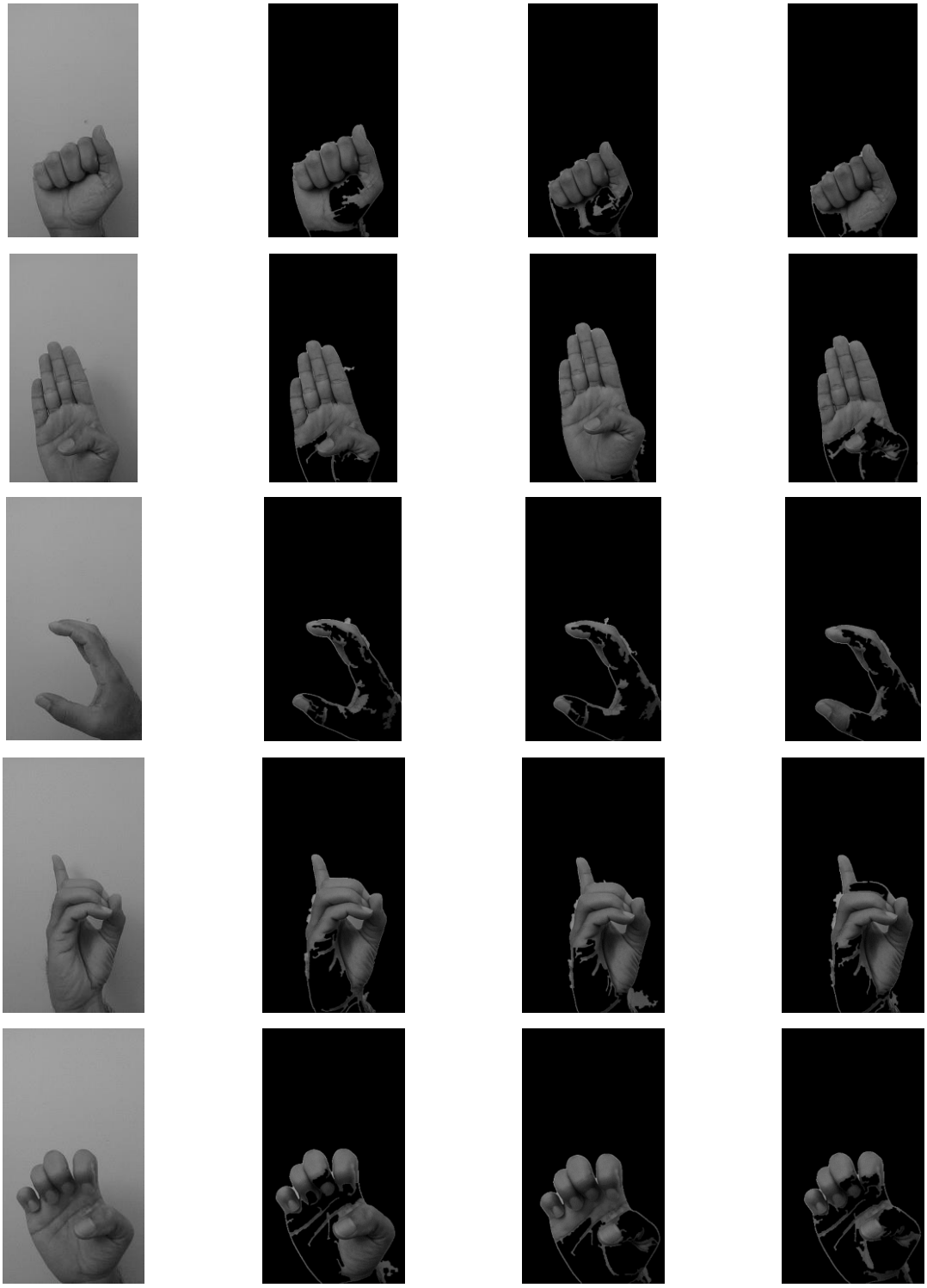


Figura 44: Algoritmo 5.3.3.2 no contexto móvel

O mesmo algoritmo quando aplicado em um plano de fundo complexo como mostra a Figura 45 trouxe resultados que seriam muito complicados de tratar para obter uma remoção completa do plano de fundo.

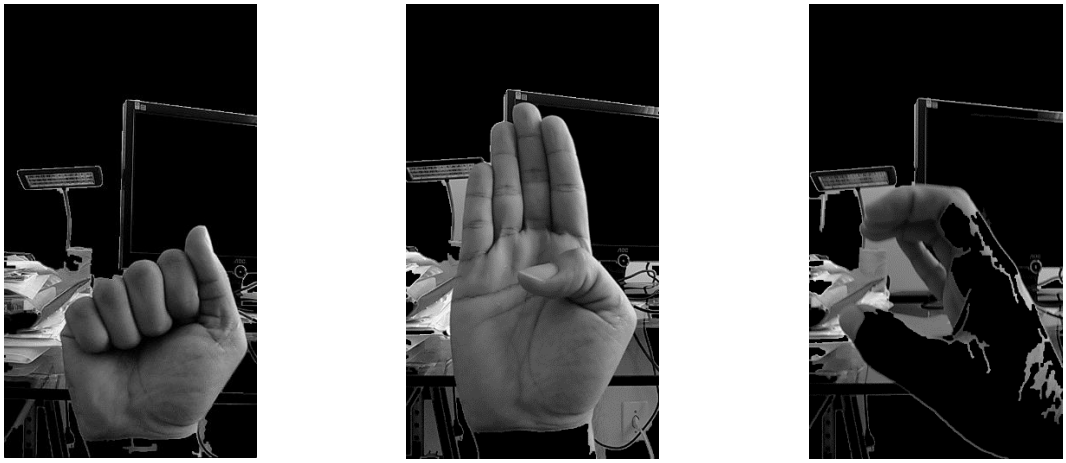


Figura 45: Algoritmo 5.3.3.2 no contexto móvel em plano de fundo complexo

Aplicando o algoritmo de remoção de plano de fundo, *BackgroundSubtratorMOG2*, do *OpenCV*, tivemos uma melhora significativa quanto à remoção do plano de fundo, porém a sensibilidade ainda está bastante presente. A Figura 46 ilustra alguns dos resultados obtidos desta técnica.

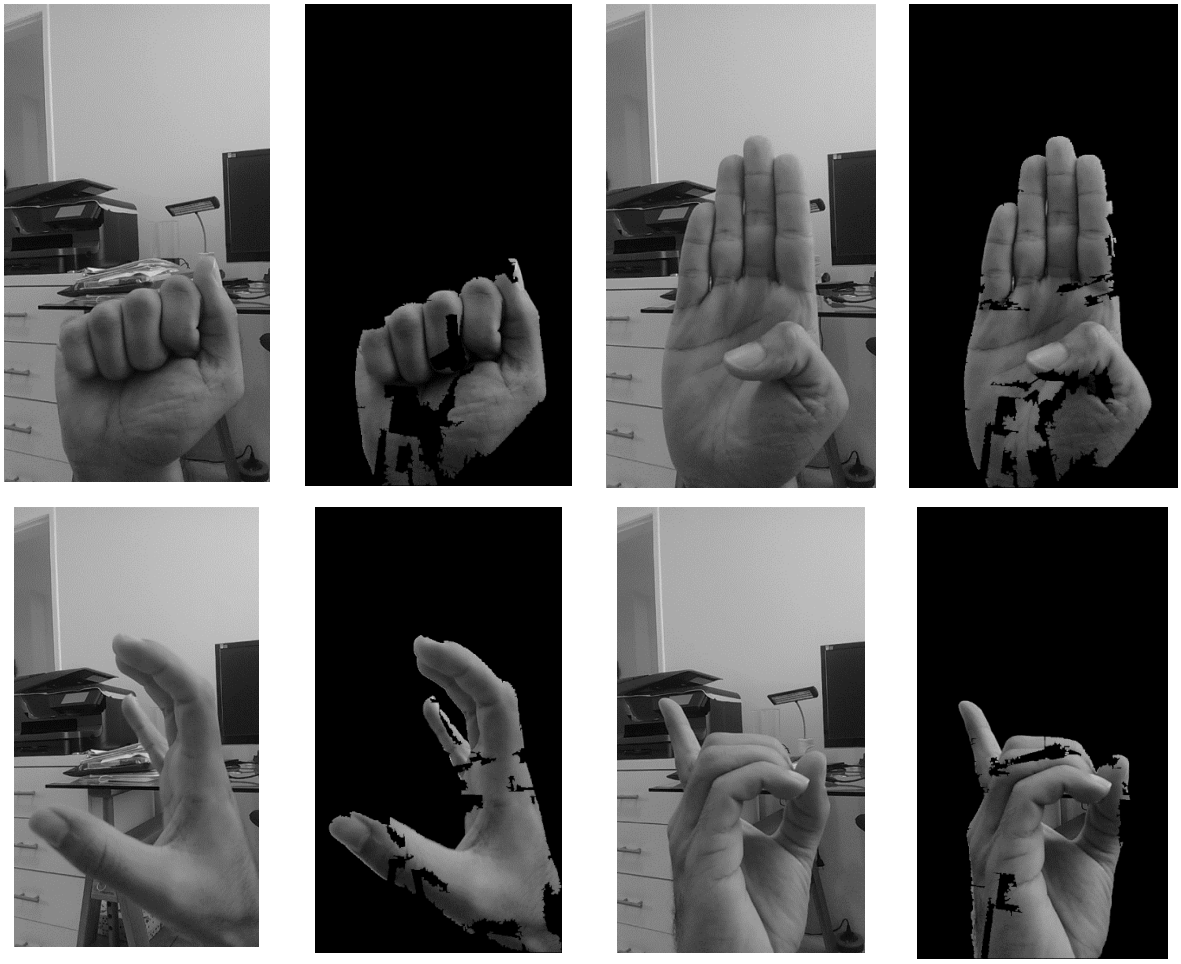


Figura 46: Remoção do plano de fundo complexo
Usando *BackgroundSubtratorMOG2*

Devido a sensibilidade apresentada na remoção do plano de fundo utilizado o *BackgroundSubtratorMOG2*, o resultado apresentado apresentou nenhuma imagem com o resultado OK – Plano de fundo removido com sucesso deixando somente a mão em destaque. Devido a isto, o critério de acerto passou a ser o Parcial - Plano de fundo removido, porém mão não está completa por apresentar áreas pretas como se fossem parte do plano de fundo.

	OK	**Parcial**	**Erro**	**Taxa de Remoção**
A	0	8	8	80%
B	0	7	3	70%
C	0	10	0	100%
D	0	3	7	30%
E	0	4	6	40%
F	0	8	2	80%
G	0	10	0	100%
H	0	7	3	70%
I	0	9	1	90%
J	0	4	6	40%
K	0	7	3	70%
L	0	9	1	90%

Tabela 2: Resultado obtido utilizando *BackgroundSubtratorMOG2*

A Tabela 2 apresenta os resultados da remoção no contexto móvel. Neste contexto foi utilizado um plano de fundo complexo e conforme explicado anteriormente, apresenta uma alta sensibilidade, não apresentando os mesmos resultados de um ambiente homogêneo. Apesar desta sensibilidade, podemos observar que o plano de fundo foi removido completamente.

7.2 Classificação

Para a classificação, testes foram realizados utilizando duas técnicas de inteligência artificial conforme explicado anteriormente (SVM e ELM), cujos resultados são apresentados nesta seção.

7.2.1 Resultados do SVM

Como explicado na seção 3.1, página 25, o SVM foi investigado através de dois tipos de experimento: uma classe contra todas e classes duas a duas. O SVM é treinado usando as imagens da base (em torno de 40 imagens para cada letra) e para a classificação são usadas 10 imagens diferentes.

O resultado do primeiro experimento é mostrado na Tabela 3. Este primeiro experimento teve como objetivo verificar a taxa de acerto em uma classificação do tipo um contra todos. Dez imagens capturadas do celular são fornecidas como entrada para a rede neural do tipo SVM que por sua vez, irá atribuir, para cada imagem, um rótulo correspondente. Então, dado um conjunto de entrada "*X*" e um conjunto de treinamento consistindo das letras de A a L, vamos procurar saber qual a letra que obteve o maior número de acertos com o treinamento um contra todos. Cada linha da Tabela 3 representa um conjunto de entrada de 10 imagens (coluna 1) classificadas para cada letra da coluna. Ou seja, cada célula representa a classificação de 10 imagens entre a letra da coluna contra todas as outras letras. Por exemplo: a primeira linha refere-se ao conjunto de teste de 10 imagens da letra "*A*". A primeira célula, (A, A), desta linha representa a classificação utilizando o treinamento da letra "*A*" contra todas as demais letras. Neste caso, o treinamento tem dois rótulos: 1 representando a letra "*A*" e 0 representando todas as demais letras. Com isto, esta célula representa a quantidade de acertos, (ou seja, quantidade de 1s) dentro do conjunto de teste. A quantidade de 1s representa a quantidade de imagens classificadas como a letra "*A*". Para esta célula, a rede neural SVM não classificou nenhuma imagem como sendo "*A*". A quarta célula, (A, D), desta mesma linha (conjunto de teste contendo 10 imagens da letra "*A*") representa o treinamento da letra "*D*" (com o rótulo 1, por ser a letra que estamos querendo classificar), contra todas as demais letras (rótulo 0). A rede classificou 6 imagens das 10 como sendo a letra "*D*". A décima célula obteve o resultado de 8 imagens das 10 como sendo da letra "*J*". Olhando todos os resultados desta linha, temos a letra "*J*" com o maior número de acertos, sendo ela portanto, a letra final para o conjunto de teste "*A*".

Analisando uma outra linha da Tabela 3, a linha 5, podemos verificar a quantidade de acertos seguindo a mesma linha de raciocínio acima, porém agora utilizando um conjunto de teste contendo 10 imagens da letra "*D*". Temos na primeira célula, (D, A), um conjunto de teste contendo 10 imagens da letra "*D*" e o treinamento

da rede da letra “A” contra as demais. Neste caso, a rede não classificou nenhuma imagem como sendo “A”. Na quarta célula, (D, D), o classificador com treinamento D contra as demais letras resultou na classificação das 10 imagens todas pertencendo à letra “D”. Podemos observar na célula 8, (D, H), que houve uma classificação de 9 imagens sendo “H” e o mesmo aconteceu para a célula 9, (D, J), porém 9 imagens foram classificadas como sendo “J”. Como a quarta célula, (D, D), obteve o maior número, o resultado final da classificação ficou sendo a letra “D” obtendo assim uma classificação correta.

Quando a classificação tiver acabado para todas as letras, o resultado final para o conjunto de testes vai ser a célula que tiver o maior número. No caso da primeira linha foi a letra J. As células em vermelho destacam essas classificações incorretas, enquanto a diagonal cinzenta marca as células que idealmente deveriam ter 10 imagens classificadas (as 10 imagens do conjunto de teste).

A última coluna da Tabela 3 representa o resultado final do classificador para todos os conjuntos de teste. Utilizando a abordagem um contra todos foi obtida uma taxa de 33% acertando 4/12 letras.

	A	B	C	D	E	F	G	H	I	J	K	L	
A	0	0	0	6	0	0	2	0	0	8	0	0	J
B	0	0	0	0	1	0	1	8	0	2	9	0	K
C	0	0	0	0	0	0	1	5	0	10	1	1	J
D	0	0	0	10	0	0	0	9	0	9	0	0	D
E	0	0	0	0	0	0	3	10	0	2	4	0	H
F	0	0	0	0	2	0	0	0	2	0	6	0	K
G	0	0	0	0	0	0	0	7	0	8	6	0	J
H	0	0	9	1	0	0	0	10	0	9	0	0	H
I	0	0	1	1	0	0	0	0	10	5	0	0	I
J	0	0	0	0	0	0	7	0	0	10	4	0	J
K	0	1	0	0	0	0	0	7	0	0	1	0	H
L	0	0	3	0	0	0	0	0	0	8	0	6	J

Tabela 3: Resultado SVM um contra todos

Para tentar obter uma melhor taxa, foi adotado o treinamento realizado de duas em duas classes (segundo experimento). Por exemplo: o treinamento da letra “A” é uma composição do treinamento de “A” contra “B”, “A” contra “C” e assim

sucessivamente. Como o treinamento ocorre somente entre duas classes distintas, não podemos treinar “A” contra “A”, “B” contra “B” e assim por diante, pois só teríamos uma classe. Por isso, a diagonal da Tabela 4 é marcada como "não se aplica" (já que a classe final é conhecida a priori neste experimento).

Assim na Tabela 4, a primeira linha refere-se ao conjunto de teste de 10 imagens da letra A. Esse conjunto é testado primeiramente com o conjunto de treinamento da letra A contra B, que define se cada imagem corresponde à letra A ou B. Em seguida, o mesmo conjunto será testado com o conjunto de treinamento da letra A contra C (terceira coluna) (novamente, será verificado quantas imagens do conjunto de teste são classificadas como A e quantas são classificadas como C). O mesmo procedimento é repetido para todas as outras letras. Ou seja, cada célula na primeira coluna da Tabela 4 vai representar agora a classificação entre duas letras ao invés de uma contra todas as outras e cada célula interna mostra a quantidade de imagens do conjunto de teste classificadas em cada comparação como sendo da letra da coluna correspondente. Então para a linha 2, coluna 1, teremos a letra A contra B, A contra C, A contra D e assim sucessivamente. A classificação entre A contra A, ou B contra B neste caso não é possível por que só teremos uma classe como resultado. Tomando como exemplo a segunda linha, temos a classificação entre a primeira célula e cada coluna para um conjunto de 10 imagens. Então, a célula (A, B) é o teste de A contra B para o conjunto de teste com 10 imagens. Neste caso o classificador acertou todas as imagens como sendo do sinal A. A célula (A, C) é o teste de A contra C para o conjunto de teste com 10 imagens. Neste caso ele acertou apenas 4 imagens como sendo A e classificou 6 como sendo C. Sendo assim, o sinal A não é o sinal majoritário e sim o C, dentro do conjunto de 10 imagens, por isto a célula ficou em vermelho. Na célula (A, G) o classificador acertou 6 imagens como A e o restante (4) como sendo G. A classe A é o sinal que mais aparece, por isso a célula está em verde.

As células em vermelho destacam situações em que a quantidade de imagens classificadas corretamente é menor do que a metade do número de imagens do conjunto de teste (ou seja menor ou igual a 5). As células verdes indicam as situações em que mais da metade das imagens foram classificadas corretamente. Uma célula verde, portanto, indica que o sinal majoritário foi a letra da linha (o sinal correto) e representa um acerto. A taxa de acerto (mostrada na última coluna) é calculada de acordo com o número de células verdes da linha (número de acertos total para todas

as comparações duas a duas) em relação ao número total de letras da base menos 1 (excetuando-se a própria letra do conjunto de entrada).

Os resultados deste experimento com fundo complexo são mostrados na Tabela 4 para todas as letras da base.

	A	B	C	D	E	F	G	H	I	J	K	L	
A	n/a	10	4	2	4	8	6	3	1	0	10	6	45%
B	4	n/a	3	0	6	3	0	0	9	9	0	10	36%
C	10	10	n/a	0	10	10	10	10	2	6	10	9	81%
D	10	10	10	n/a	10	10	10	0	10	1	4	10	72%
E	7	2	1	0	n/a	9	4	0	4	4	3	0	18%
F	10	10	10	10	3	n/a	10	4	6	4	10	4	63%
G	10	10	6	10	10	10	n/a	7	10	10	4	10	90%
H	10	10	4	10	10	10	10	n/a	10	7	10	10	90%
I	10	10	10	10	10	10	10	10	n/a	10	10	10	100%
J	10	10	10	10	10	10	10	10	6	n/a	10	10	100%
K	10	4	10	1	10	10	10	10	10	10	n/a	10	81%
L	8	10	7	10	10	10	10	10	8	10	10	n/a	100%

Tabela 4: Resultados SVM com classes duas a duas

Trazendo esta abordagem para um contexto mais real em que a classe do conjunto de teste é desconhecida, foi feito o mesmo procedimento acima, porém um mesmo conjunto de teste foi classificado utilizando todos os conjuntos de treinamento (de duas classes cada). Ou seja, um conjunto de teste de imagens de uma letra desconhecida precisa ser testado contra os conjuntos de treinamento A contra B, A contra C, etc.; depois B contra A, B contra C, etc., e assim para todas as letras. Por exemplo: dado um conjunto de teste com 10 imagens, é calculada a porcentagem de classificação para cada letra da base conforme o procedimento acima (comparação de classes duas a duas). Aquela com maior porcentagem é o resultado final do sinal classificado. Assim, na classificação de C com D, por exemplo, 90% das imagens foram classificadas como D (Tabela 5 linha 4). Como esta taxa foi a maior obtida, o resultado final para a entrada C foi a letra D. Os resultados para todas as letras da base são mostrados na Tabela 5.

A primeira coluna da Tabela 5 representa os dados de entrada “desconhecidos” e a primeira linha representa as letras que podem ser classificadas. Na parte interna

da tabela, a linha diagonal (em cinza) indica a posição na tabela da letra que deveria ser classificada (ou seja, a célula idealmente deveria conter 100%). As células diagonais da Tabela 5 correspondem aos resultados das linhas da Tabela 5. Nesta abordagem, de acordo com os resultados acima, o SVM obteve uma taxa de acerto de 33%. Com um alto custo computacional e uma baixa taxa de acerto, a abordagem de SVM foi descartada. Podemos entender melhor a Tabela 5 como cada célula representando uma linha da Tabela 4. Por exemplo, analisando a primeira linha da Tabela 5, a célula (A, A) representa a primeira linha da Tabela 4. A segunda célula (A, B) equivale à segunda linha, e assim por diante, porém cada célula desta linha usando o conjunto de treinamento da letra “A”. Na segunda linha da Tabela 5, a célula (B, A) é equivalente à primeira linha da Tabela 4, a segunda célula (B, B) à segunda linha e assim por diante, porém, cada célula é classificada usando o conjunto de treinamento da letra “B”. Com isto, nós temos uma porcentagem de cada letra para o mesmo conjunto de treinamento. Podemos observar na primeira linha que a célula (A, J) equivalente à 11ª linha da Tabela 3, em que o classificador acertou todas as imagens do conjunto de teste como sendo “J”. Comparando com as outras porcentagens nesta mesma linha, ela é a que obteve o maior valor, sendo assim o resultado final. Na linha da letra “F”, a célula (F, E), equivalente à linha 7 da Tabela 4, acertou 81% das imagens como sendo da letra E. Apesar de ter acertado 72% como sendo da letra “F”, o resultado final ficou sendo “E” já que tinha a maior porcentagem.

	A	B	C	D	E	F	G	H	I	J	K	L
A	45%	0%	63%	63%	36%	9%	63%	27%	90%	100%	54%	45%
B	9%	36%	72%	72%	45%	36%	54%	90%	18%	27%	100%	54%
C	9%	9%	81%	90%	63%	27%	36%	54%	45%	45%	54%	81%
D	18%	27%	72%	72%	9%	9%	54%	90%	36%	72%	81%	63%
E	9%	27%	81%	72%	45%	18%	54%	100%	27%	54%	54%	72%
F	27%	36%	18%	63%	81%	72%	9%	63%	63%	63%	72%	63%
G	27%	18%	72%	72%	54%	0%	100%	81%	36%	54%	90%	9%
H	45%	9%	100%	81%	36%	27%	45%	90%	45%	63%	72%	45%
I	45%	36%	90%	81%	27%	18%	27%	18%	100%	72%	54%	27%
J	54%	9%	36%	72%	18%	36%	81%	9%	90%	100%	72%	36%
K	27%	100%	54%	72%	63%	0%	54%	36%	27%	27%	81%	63%
L	54%	0%	90%	63%	63%	27%	36%	36%	54%	72%	36%	100%

Tabela 5: Resultados SVM com entrada desconhecida

7.2.2 Resultados do ELM

Conforme mencionado antes, a classificação usando ELM inicialmente obteve uma taxa de acerto de 0% devido ao formato de imagens usado, e mesmo usando um elevado número de neurônios, a taxa de acerto não surtia nenhuma melhora. Com o banco de dados refeito para manter a proporcionalidade entre as diferentes imagens da mão, novos experimentos foram realizados para avaliar o treinamento e classificação.

Para obter as imagens a serem classificadas, o aplicativo móvel foi programado para capturar e classificar 10 quadros e verificar qual a classe predominante (ou seja, o sinal reconhecido) para definir o resultado final. Esta abordagem foi feita porque classificar quadro a quadro deixaria o resultado final inconsistente, mudando a todo instante. Esperar uma convergência de uma mesma resposta para todos os quadros poderia levar a um *loop* infinito - por isto a resposta final da classificação foi baseada no sinal retornado mais vezes. Para isto, os quadros são salvos sequencialmente em um arquivo à medida que são capturados, e são enviados em grupos de 10 para a classificação. Os experimentos realizados são descritos a seguir.

7.2.3 Experimento 1: Ajuste de Neurônios na Camada Escondida

Foi feito um teste inicial com as letras A, B, C, D, E. Cada letra contém uma base de 30 imagens 150 x 150 pixels capturadas por um celular e tratadas, antes de serem usadas para treinamento. As imagens usadas para o teste de classificação foram capturadas pelo mesmo celular, porém usando o algoritmo de remoção do plano de fundo do OpenCV. Foram usadas 5 imagens de teste para cada letra, as quais foram selecionadas por terem um bom resultado de remoção de plano de fundo. Resolvemos fazer essa seleção para melhor analisar a quantidade ideal de neurônios da camada intermediária da rede. A quantidade mínima foi de 5.000 (a mesma quantidade adotada no artigo anteriormente citado de (NETO, LUDERMIR, & ZANCHETTIN, 2015), chegando a um número máximo de 15.000 neurônios na camada escondida e a rede foi treinada e testada 3 vezes a fim de obter um resultado mais constante. A Tabela 6 mostra os resultados iniciais obtidos na primeira execução.

BASE	A	B	C	D	E	Total
5.000 Neurônios						
Acertos	5	5	5	2	5	22
Erros	0	0	0	3	0	3
%	100%	100%	100%	40%	100%	88%
10.000 Neurônios						
Acertos	5	5	5	2	5	22
Erros	0	0	0	3	0	3
%	100%	100%	100%	40%	100%	88%
15.000 Neurônios						
Acertos	4	5	5	2	5	21
Erros	1	0	0	3	0	4
%	80%	100%	100%	40%	100%	84%

Tabela 6: Resultados iniciais para experimento 1 execução 1

Utilizando 5, 10 e 15 mil neurônios, 5 letras na base (aproximadamente 120 imagens) e classificação para letras A, B, C, D, E.

A Tabela 7 mostra os resultados obtidos depois da segunda execução e podemos ver que não houve muita variação entre os resultados. Devido aos pesos encontrados aleatoriamente a cada treinamento e classificação, houve uma variação na quantidade de imagens que foram classificadas corretamente.

BASE	A	B	C	D	E	Total
5.000 Neurônios						
Acertos	4	5	5	3	5	22
Erros	1	0	0	2	0	3
%	80%	100%	100%	60%	100%	88%
10.000 Neurônios						
Acertos	5	5	4	2	5	21
Erros	0	0	1	3	0	4
%	100%	100%	80%	40%	100%	84%
15.000 Neurônios						
Acertos	4	5	5	2	5	21
Erros	1	0	0	3	0	4
%	80%	100%	100%	40%	100%	84%

Tabela 7: Resultados iniciais para experimento 1 execução 2

Utilizando 5, 10 e 15 mil neurônios, 5 letras na base (aproximadamente 120 imagens) e classificação para letras A, B, C, D, E.

BASE	A	B	C	D	E	Total
5.000 Neurônios						
Acertos	5	5	5	4	5	24
Erros	0	0	0	1	0	1
%	100%	100%	100%	80%	100%	96%
10.000 Neurônios						
Acertos	4	5	5	4	5	23
Erros	1	0	0	1	0	5
%	80%	100%	100%	80%	100%	92%
15.000 Neurônios						
Acertos	4	5	5	2	5	21
Erros	1	0	0	3	0	4
%	80%	100%	100%	40%	100%	84%

Tabela 8: Resultados iniciais para experimento 1 execução 3

Utilizando 5, 10 e 15 mil neurônios, 5 letras na base (aproximadamente 120 imagens) e classificação para letras A, B, C, D, E.

Adicionando mais letras à base de treinamento, passando a abranger de A a L, cada letra contando agora com 40 imagens de 150 x 150 pixels, e usando as mesmas imagens de teste da classificação anterior, tivemos os resultados mostrados Tabela 9.

BASE	A	B	C	D	E	Total
5.000 Neurônios						
Acertos	4	4	3	5	5	21
Erros	1	1	2	0	0	4
%	80%	80%	60%	100%	100%	84%
10.000 Neurônios						
Acertos	5	3	2	4	5	19
Erros	0	2	3	1	0	6
%	100%	60%	40%	80%	100%	76%
15.000 Neurônios						
Acertos	5	3	3	4	5	20
Erros	0	2	2	1	0	5
%	100%	60%	60%	80%	100%	80%

Tabela 9: Novos resultados para experimento 1 execução 1

Utilizando 5, 10 e 15 mil neurônios, 12 letras na base (aproximadamente 500 imagens) e classificação para letras A, B, C, D, E

BASE	A	B	C	D	E	Total
5.000 Neurônios						
Acertos	5	4	4	3	5	21
Erros	0	1	1	2	0	4
%	100%	80%	80%	60%	100%	84%
10.000 Neurônios						
Acertos	4	3	2	5	5	19
Erros	1	2	3	0	0	6
%	80%	60%	40%	100%	100%	76%
15.000 Neurônios						
Acertos	5	3	3	4	5	20
Erros	0	2	2	1	0	5
%	100%	60%	60%	80%	100%	80%

Tabela 10: Novos resultados para experimento 1 execução 2

Utilizando 5, 10 e 15 mil neurônios, 12 letras na base (aproximadamente 500 imagens) e classificação para letras A, B, C, D, E

BASE	A	B	C	D	E	Total
5.000 Neurônios						
Acertos	5	4	4	3	5	21
Erros	0	1	1	2	0	4
%	100%	80%	80%	60%	100%	84%
10.000 Neurônios						
Acertos	4	3	3	4	5	19
Erros	1	2	2	1	0	6
%	80%	60%	60%	80%	100%	76%
15.000 Neurônios						
Acertos	5	4	3	4	5	21
Erros	0	1	2	1	0	4
%	100%	60%	60%	80%	100%	84%

Tabela 11: Novos resultados para experimento 1 execução 3

Utilizando 5, 10 e 15 mil neurônios, 12 letras na base (aproximadamente 500 imagens) e classificação para letras A, B, C, D, E

A partir destes resultados iniciais, concluímos que aumentar a quantidade de neurônios além de 5.000 não trouxe melhora na taxa de acerto, só aumentando o tempo de duração tanto do treinamento quanto da classificação. O treinamento para as letras de A a L durou: 15 segundos para 5 mil neurônios; 31 segundos para 10 mil neurônios e 47 segundos para 15 mil neurônios. Com isto, foi adotado o número de 5.000 neurônios na camada intermediária para os testes seguintes.

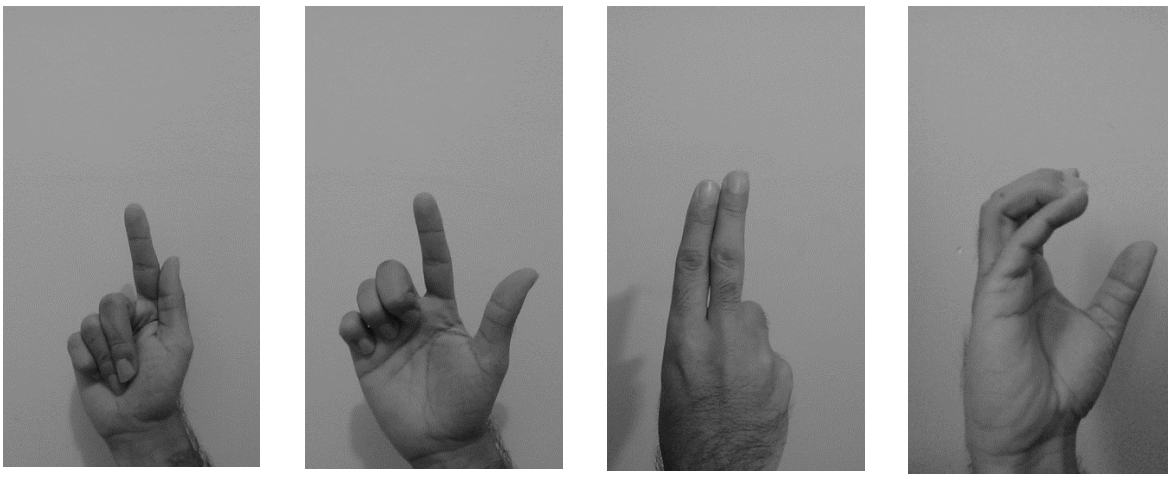
7.2.4 Experimento 2: Resultados em um Plano de Fundo Homogêneo

Após os testes iniciais, foram capturadas 10 imagens para cada letra de A a L, sequencialmente através da câmera do celular, sem selecionar as imagens com melhor remoção de plano de fundo. Os testes nesse experimento foram realizados com um número fixo de neurônios na camada intermediária (5.000). A Tabela 12 mostra os novos resultados. Para cada letra da primeira coluna indicam-se as letras correspondentes identificadas pelo algoritmo de classificação (dispostas na primeira linha). Por exemplo, na primeira linha, tivemos 100% de acerto na classificação das 10 imagens da letra A. Já para a letra B (segunda linha), o algoritmo acertou 5 imagens, mas também classificou quatro imagens como sendo da letra H, e uma como letra I. O neurônio que tiver o maior valor na camada de saída, é o resultado final. Na última coluna é apresentado o resultado final da classificação a partir do maior número de cada linha.

	A	B	C	D	E	F	G	H	I	J	K	L		
A	10	0	0	0	0	0	0	0	0	0	0	0	A	100%
B	0	5	0	0	0	0	0	4	1	0	0	0	B	50%
C	0	0	10	0	0	0	0	0	0	0	0	0	C	100%
D	0	3	0	5	2	0	0	0	0	0	0	0	D	50%
E	0	0	0	0	10	0	0	0	0	0	0	0	E	100%
F	0	0	0	0	0	10	0	0	0	0	0	0	F	100%
G	0	1	0	0	1	0	0	0	0	0	0	8	L	0%
H	0	4	6	0	0	0	0	0	0	0	0	0	C	0%
I	0	0	0	0	0	0	0	0	10	0	0	0	I	100%
J	0	0	0	0	0	0	0	0	0	10	0	0	J	100%
K	0	0	0	0	0	0	0	0	0	0	10	0	K	100%
L	0	0	0	0	0	0	0	0	0	0	0	10	L	100%

Tabela 12: Resultados da classificação plano de fundo homogêneo
Utilizando 5.000 neurônios e base de dados com as letras de 'A' a 'L'

Neste teste, o classificador obteve uma taxa de 84% de acerto, acertando 10 letras de um total de 12. Os piores casos de classificação foram as letras G e H, para as quais o classificador não acertou a letra correspondente com nenhuma das imagens testadas como podemos ver na Figura 47, a diferença entre as letras. O erro é compreensível no caso das letras G e L, cujos sinais se diferenciam principalmente pela posição do polegar. Já a confusão entre o H e o C é menos justificável.



G L H C

Figura 47: Diferenças entre as letras G, L, H e C
O sinal G foi classificado como L, e o sinal H foi classificado como C

7.2.5 Experimento 3: Resultados em um Plano de Fundo Complexo

Este experimento envolve a classificação utilizando um plano de fundo complexo, capturando imagens sequencialmente sem nenhuma pré-seleção. Na Figura 48 é apresentado o plano de fundo complexo utilizado para este teste.

Figura 48: Exemplo de plano de fundo complexo

A Tabela 13 apresenta os resultados obtidos usando dez imagens. Com taxa de acerto de 58.3%, os resultados obtidos para um fundo complexo ficaram bem abaixo dos resultados anteriores, porém isso se justifica pela complexidade da remoção deste tipo de plano de fundo.

	A	B	C	D	E	F	G	H	I	J	K	L		
A	0	0	0	2	1	0	0	0	0	6	0	1	J	0%
B	0	0	0	0	0	1	0	9	0	0	0	0	H	0%
C	0	0	0	0	0	1	0	5	0	3	0	1	H	0%
D	0	0	0	10	0	0	0	0	0	0	0	0	D	100%
E	0	0	0	0	0	5	0	5	0	0	0	0	F/H	0%
F	0	0	0	0	4	6	0	0	0	0	0	0	F	60%
G	2	0	0	0	0	0	6	0	0	0	2	0	G	60%
H	0	0	0	0	0	0	0	10	0	0	0	0	H	100%
I	0	0	0	0	0	0	0	0	10	0	0	0	I	100%
J	1	0	0	0	1	0	0	0	2	5	1	0	J	50%
K	0	1	0	0	0	0	0	0	0	0	9	0	K	90%
L	0	0	10	0	0	0	0	0	0	0	0	0	C	0%

Tabela 13: Resultados da classificação com plano de fundo complexo execução 1
Utilizando 5.000 neurônios para a base de ‘A’ a ‘L’

Como a rede neural do tipo ELM seleciona os pesos aleatoriamente, foram feitas mais duas execuções a fim de garantir a consistência da rede. A Tabela 14 e a Tabela 15 mostram a segunda e terceira execuções para as letras de A a L.

	A	B	C	D	E	F	G	H	I	J	K	L		
A	0	0	0	2	0	0	0	0	0	8	0	0	J	0%
B	0	0	0	0	0	0	0	10	0	0	0	0	H	0%
C	0	0	0	0	0	0	0	0	4	4	0	2	I/J	0%
D	0	0	0	10	0	0	0	0	0	0	0	0	D	100%
E	0	0	0	2	0	2	0	6	0	0	0	0	H	0%
F	0	0	0	0	3	7	0	0	0	0	0	0	F	70%
G	0	1	0	0	0	0	8	1	0	0	0	0	G	80%
H	0	0	1	2	0	0	0	7	0	0	0	0	H	70%
I	0	0	0	0	0	0	0	0	10	0	0	0	I	100%
J	2	0	0	0	2	0	1	0	1	4	0	0	J	40%
K	0	2	0	0	0	0	0	0	0	0	8	0	K	80%
L	1	0	9	0	0	0	0	0	0	0	0	0	C	0%

Tabela 14: Resultados da classificação com plano de fundo complexo - execução 2
Utilizando 5.000 neurônios para a base de ‘A’ a ‘L’

Podemos verificar que na segunda e terceira execuções, a rede ELM continuou com um acerto global de 58.3%, acertando 7 de 12 letras e mantendo a consistência das letras acertadas em cada execução, porém houve algumas variações em quase todas as letras devido aos pesos serem computados aleatoriamente a cada novo treinamento.

	A	B	C	D	E	F	G	H	I	J	K	L		
A	0	0	0	2	1	0	1	0	0	6	0	0	J	0%
B	0	0	0	0	0	9	1	0	0	0	0	0	F	0%
C	0	0	0	0	0	7	0	0	0	3	0	0	F	0%
D	0	0	0	10	0	0	0	0	0	0	0	0	D	100%
E	0	0	0	2	0	8	0	0	0	0	0	0	F	0%
F	0	0	0	0	3	7	0	0	0	0	0	0	F	70%
G	0	0	0	1	0	0	4	2	0	0	2	0	G	40%
H	0	0	0	0	0	1	0	9	0	0	0	0	H	90%
I	0	0	0	0	0	0	0	0	10	0	0	0	I	100%
J	2	0	0	0	0	0	0	0	2	5	1	0	J	50%
K	0	0	0	0	0	0	0	0	0	0	10	0	K	100%
L	0	0	10	0	0	0	0	0	0	0	0	0	C	0%

Tabela 15: Resultados da classificação com plano de fundo complexo - execução 3
Utilizando 5.000 neurônios para a base de ‘A’ a ‘L’

A Tabela 16 mostra os resultados das letras que tiveram sua classificação correta nas três execuções. As letras D e I mantiveram a mesma taxa de acerto enquanto as outras letras tiveram pouca variação para menos ou para mais, como as letras F, H, J e K. A letra G mostrou uma maior variação entre cada execução: 60% na primeira execução, 80% na segunda e caindo para 40% na terceira execução. De forma geral, podemos concluir que a rede neural apresentou um resultado global constante, mantendo uma consistência entre as execuções.

	Acertos na Execução 1	Acertos na Execução 2	Acertos na Execução 3
D	10	10	10
F	6	7	7
G	6	8	4
H	10	7	9
I	10	10	10
J	5	4	5
K	9	8	10

Tabela 16: Acertos da ELM nas 3 execuções

Apesar dos resultados ruins com fundo complexo, a abordagem proposta tem potencial de se tornar uma solução viável e eficiente para solucionar o problema da comunicação entre surdos de diferentes países. A maioria dos problemas decorre da deficiência na limpeza do ruído complexo e pela pequena quantidade de exemplos de treinamento. Em relação ao primeiro problema, para melhorar a remoção do plano de fundo, podem ser usadas técnicas como detecção de regiões salientes baseada no contraste global (CHENG, MITRA, HUANG, TORR, & HU, 2011). Esta técnica consiste em segmentar a imagem em regiões que recebem valores de saliência (calculados de acordo com o escore de contraste global, medido pelo contraste da região e distância espacial para outras regiões). Para isto, em cada região é analisado o histograma para produzir um mapa de saliência.

Em relação ao segundo problema (quantidade de exemplos), com o decorrer do uso do sistema, a tendência é haver uma melhoria nas taxas de acerto da classificação já que novas imagens podem ser inseridas pelos usuários para treinar o sistema. Além disso, as imagens classificadas corretamente (de acordo com o *feedback* do usuário) podem ser retidas e usadas pelo sistema, que poderia adicionar as informações das imagens corretas para um novo treinamento. Com isto, a base teria uma diversidade maior de exemplos positivos possivelmente aumentando a taxa de acerto da classificação.

7.2.6 Comparação com Abordagens da Literatura

Não foi possível realizar a comparação de forma ideal do modelo proposto com abordagens existentes, pois não conseguimos acesso a outras implementações, porém é possível comparar os resultados apresentados com algumas publicações existentes na literatura. Por esta abordagem adotar uma base, metodologia e experimentos diferentes das usadas nos trabalhos da literatura não é possível realizar uma comparação direta, mas alguns indícios podem ser verificados que posicionam a proposta no contexto dos demais trabalhos.

A proposta de (NETO, LUDERMIR, & ZANCHETTIN, 2015) obteve uma taxa de 95% de acerto, com desempenho bem parecido com os obtidos usando plano de fundo homogêneo, que foi o utilizado pelos autores.

O artigo de (DAVIS & SHAH, 2002) obteve uma taxa de 94% para reconhecer 7 gestos: esquerda, direita, em cima, em baixo, rotacionar, segurar e parar. Seus resultados foram obtidos através de 10 tentativas.

A proposta de (SORIA, SCHAEFFER, & VILLAREAL, 2013) obteve uma taxa de 93% para reconhecer a quantidade de dedos levantados em qualquer mão.

Os resultados de (BLACK & JEPSON, 1998) informam o quadro em que os gestos considerados (começar, cortar, limpar, imprimir, salvar e sair) foram identificados, dentro de um conjunto de 850 quadros. Vale ressaltar que estes gestos são baseados nas trajetórias das mãos. Em outro artigo neste mesmo contexto de trajetórias, (MCKENNA & GONG, 1999) reconheceram 4 gestos: esquerda, direita, acenar acima e acenar abaixo. De 12 tentativas, os dois primeiros gestos tiveram 100% de acerto enquanto o terceiro obteve 66% e o último 91% de acerto.

Para o plano de fundo complexo, como nenhum artigo utilizou este tipo de abordagem, não podemos aferir uma comparação relevante.

7.3 Tradução entre Sinais

A tradução entre os sinais ocorre através do aplicativo móvel, que envia as imagens ao servidor e recebe o resultado da classificação realizada para exibir ao usuário. Foi usado como exemplo ilustrativo o sinal para a palavra xícara na língua americana (ASL - *American Sign Language*) e em LIBRAS, mostrados na Figura 49.

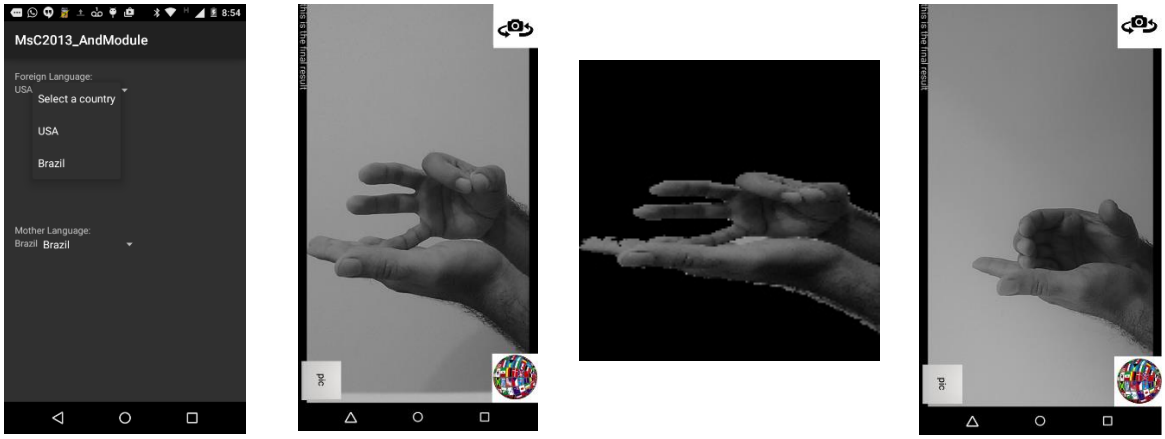
(a) (b)

Figura 49: O sinal xícara em (a) ASL[44] e (b) LIBRAS[45]

A Figura 50 mostra o fluxo da tradução entre os sinais, no dispositivo móvel desde o início do aplicativo até o resultado final. Primeiramente, como mostrado na Figura 50 (a), o país da linguagem destino (*foreign language)* foi definido como Estados Unidos (USA), e o país da linguagem local (*mother language)* como Brasil. Em seguida, é realizada a captura do sinal correspondente a xícara, que seria, no contexto da interação com o dispositivo móvel, realizado pelo usuário interlocutor, cuja língua é ASL (Figura 50 (b)). Esta imagem então é tratada para retirar o plano de fundo e redimensionada (Figura 50 (c)), e suas informações são enviadas para o servidor, que realiza a classificação e identifica o sinal como sendo de xícara em ASL. Através dos arquivos de mapeamento, o servidor encontra a imagem correspondente a xícara em LIBRAS. O aplicativo móvel recebe essa informação e exibe ao usuário o sinal de xícara em LIBRAS (Figura 50 (d)).

[44] http://www.lifeprint.com/asl101/pages-signs/c/cup.htm

[45] http://www.incluirtecnologia.com.br/



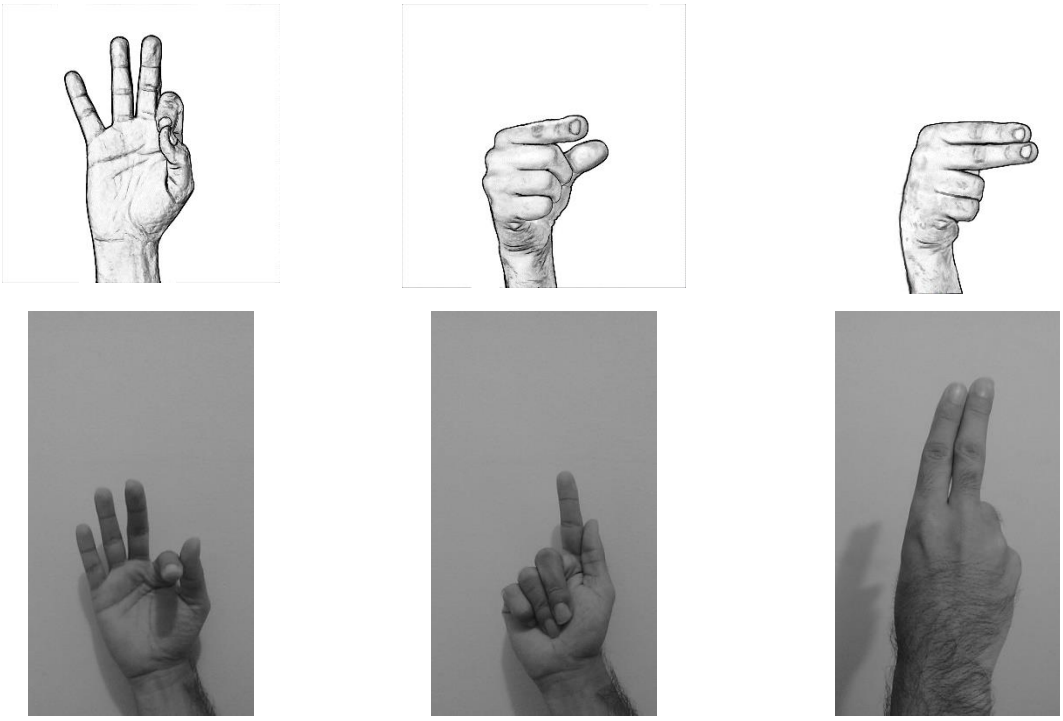
(a) (b) (c) (d)

Figura 50: Fluxo de tradução entre os sinais

(a) Tela de seleção de linguagens local e destino; (b) Captura de um quadro da câmera; (c) Imagem tratada e redimensionada; (d) Resultado final da classificação

O resultado da tradução é essencialmente o resultado da classificação, aliada a um mapeamento (feito no módulo web) entre as linguagens. Uma classificação errada resulta em uma tradução incorreta. As classificações do sinal xícara tanto em ASL (Figura 49 (a)) como em LIBRAS (Figura 49 (b)) obtiveram uma taxa de 100% de acerto nos testes realizados.

Abaixo apresentamos mais três exemplos de tradução entre sinais de ASL e LIBRAS nas letras F, G e H (mostrados na Figura 51). A primeira linha representa as letras em ASL e a segunda, em LIBRAS.



F G H

Figura 51: Diferença de F, G, H entre ASL[46] e LIBRAS (imagens elaboradas pelo autor)

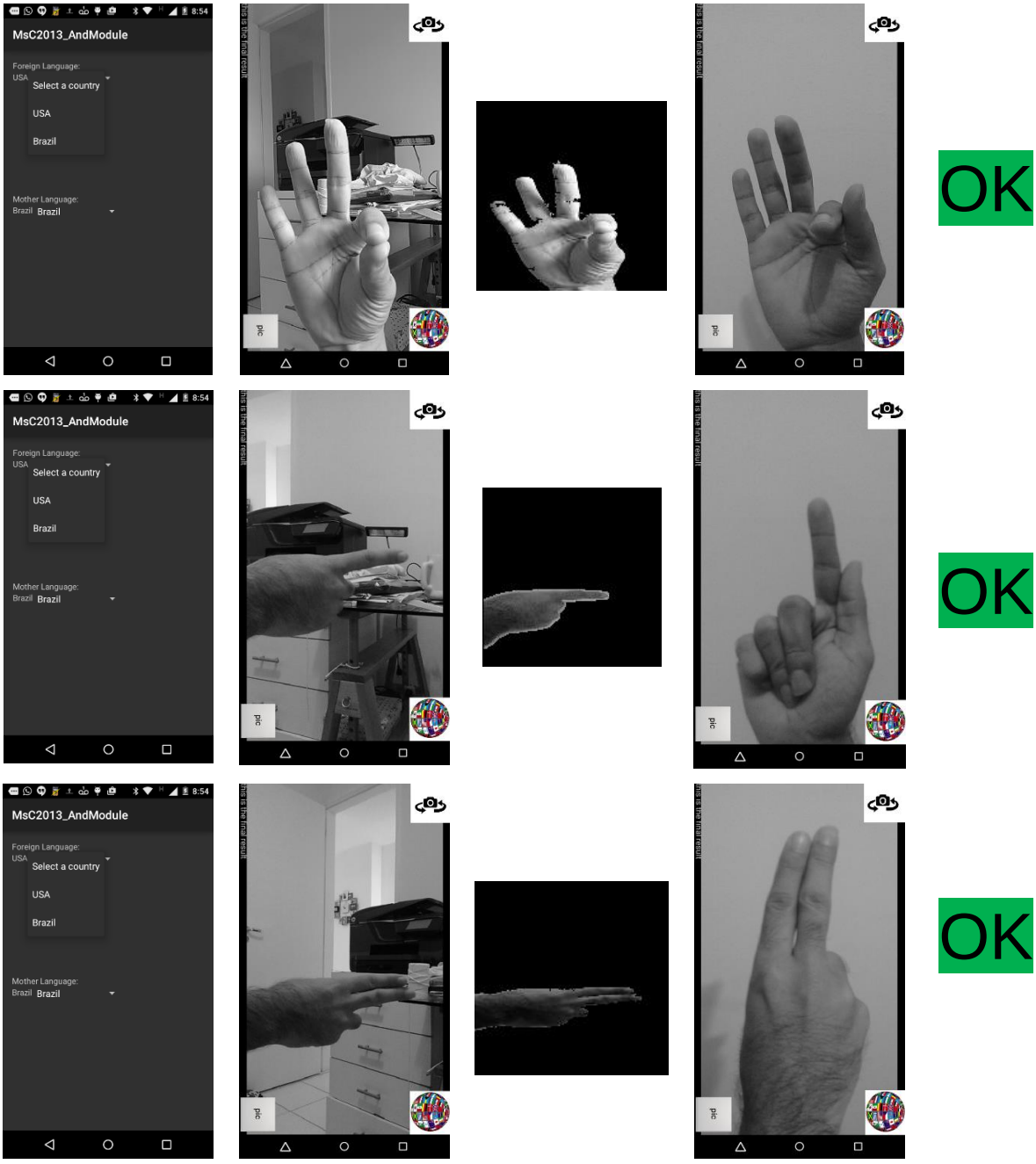
A classificação para estas três letras utilizando o alfabeto em ASL em plano de fundo complexo mostrado na Figura 52, obteve o resultado mostrado na Tabela 17:

	A	B	C	D	E	F	G	H	I	J	K	L	
F		2				5					3		50%
G						2	7	1					70%
H						1	3	5					50%

Tabela 17: Resultados para as letras F, G, H em ASL

A Figura 52 mostra o fluxo da tradução entre os sinais.

[46] http://www.lifeprint.com/ASL101/fingerspelling/abc-gifs/index.htm



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

Figura 52: Resultados da tradução de F, G, e H

(a) Seleção das linguagens a serem traduzidas. Neste caso foi escolhida Brasil como linguagem materna e USA como linguagem estrangeira; (b) Um sinal feito em ASL; (c) Tratamento da imagem a ser classificada; (d) Resultado final na lingua materna

Capítulo 8
Considerações Finais

O trabalho desenvolvido nesta pesquisa de mestrado apresenta um sistema de tradução entre linguagens de sinais utilizando diferentes tecnologias de forma integrada. O desenvolvimento ocorreu em duas plataformas, sendo dividido em dois módulos principais: o servidor e o móvel. O módulo servidor foi dividido nos seguintes submódulos: *web*, *webservice*, *Windows service*, processamento de imagens, treinamento e por fim o classificador. Os módulos proveem diversas funcionalidades que compõem o sistema, conforme descrito a seguir.

Através da página *web*, o usuário pode fazer o *login* e cadastrar países e sinais para a linguagem de um determinado país. As imagens dos sinais são usadas para o treinamento do classificador de sinais. A criação de um servidor para hospedar os sinais e permitir inserção de conteúdo pelos usuários mostrou o potencial de envolver ativamente os mesmos no processo de criação e atualização de uma linguagem de sinais em sistemas desse tipo. A ideia de desenvolver uma página *web* trouxe uma liberdade quanto à população da base de dados já que os usuários são responsáveis por criar e atualizar um sinal a partir de qualquer lugar e em qualquer hora. Ao mesmo tempo, como os módulos de tratamento, classificação e treinamento estão centralizados em um servidor, eles podem ser atualizados sem haver nenhum impacto para o usuário.

No módulo móvel, o usuário pode capturar através da câmera um sinal da linguagem de um interlocutor de uma nacionalidade diferente, utilizando o aplicativo proposto. O aplicativo se comunica com o servidor para solicitar a classificação do sinal capturado, e retorna ao usuário o sinal correspondente em sua própria linguagem de sinais.

O desenvolvimento de cada módulo do sistema trouxe muitos desafios desde a sua implementação até a integração das duas plataformas. Um dos grandes desafios foi o tratamento das imagens que serviram de base para o treinamento, em particular a remoção do plano de fundo das imagens para destacar a mão. O estudo de remoção do plano de fundo, que resultou no módulo de processamento de imagens, foi a parte que mais precisou de ajustes para obter a remoção completa, tanto no celular quanto no servidor. No módulo servidor, as imagens precisam ter um plano de fundo homogêneo para obter um bom resultado e a presença de sombra pode às vezes alterar o resultado. Apesar disto, o resultado final obtido teve uma baixa taxa de erro. A integração deste módulo com o módulo *web* provê um melhor *feedback* para o usuário da imagem final a ser usada no treinamento, podendo o usuário aceitar o tratamento de uma imagem específica ou fazer o *upload* de uma nova imagem, caso o resultado do tratamento não seja considerado satisfatório por ele.

No módulo móvel, houve ainda muita instabilidade quando o dispositivo é segurado na mão humana, gerando inconsistências entre os quadros capturados pela câmera do celular, principalmente com um plano de fundo complexo. A instabilidade faz com que nem sempre a mão seja destacada completamente do plano de fundo, como visto na seção de resultados. Apesar de alguns ruídos na imagem da mão, foi possível remover o plano de fundo das imagens capturadas no celular.

Um outro desafio foi realizar a classificação dos sinais. Foram experimentadas várias abordagens para a classificação. A rede neural ELM trouxe melhores resultados do que o SVM e o classificador *cascade*, e também menor tempo de treinamento. Não foi possível fazer a classificação no dispositivo móvel devido ao seu limitado poder computacional. Inicialmente, tentamos realizar essa classificação no celular utilizando 50 neurônios na camada intermediária da ELM, o que gerou um arquivo de tamanho aceitável (45 MB). Este arquivo, a ser baixado no celular, continha as matrizes de pesos, a serem carregadas em memória para realizar a classificação. Porém, com esta configuração, a taxa de acerto foi praticamente nula, com raros acertos. Alterando a configuração da rede através do aumento da quantidade de neurônios na camada intermediária, para melhorar os resultados, o tamanho das matrizes de pesos aumentou e consequentemente o tamanho do arquivo chegou a ter 4 GB, inviabilizando o seu *download* e o carregamento das matrizes em memória no celular. Com isto a classificação no dispositivo móvel foi descartada e passou a ser realizada no servidor.

8.1 Principais Contribuições

A principal contribuição deste trabalho é prover uma tradução entre linguagens de sinais. Para isto, foi desenvolvido um sistema em duas plataformas, com várias funcionalidades que trazem contribuições específicas sumarizadas a seguir:

- **Desenvolvimento de uma página web para centralizar todos os sinais**: armazenar todas as imagens dos sinais no celular inviabilizaria o usuário de realizar outras ações como tirar fotos e baixar aplicativos, dada a quantidade de sinais e o fato de que para cada sinal pode haver várias imagens para se obter uma classificação com um bom nível de acerto. Além disso, seria complexo sincronizar o banco de sinais nos celulares de todos os usuários do sistema para todos terem sempre as mesmas imagens. Por estas razões, foi utilizado um servidor, resolvendo assim o problema de armazenamento de todas as imagens dos sinais. Para fazer com que o usuário ficasse mais independente quanto às atualizações dos sinais, foi utilizado o mesmo servidor para hospedar e disponibilizar uma página web, onde o usuário pode submeter novas imagens. A cada novo sinal criado, um arquivo de mapeamento é atualizado para permitir a tradução pelo aplicativo móvel.
- **Desenvolvimento de um método de tratamento de imagens para remoção do plano de fundo**: as imagens enviadas pelos usuários precisam passar por um tratamento para separar a mão do plano de fundo e assim identificar o sinal feito pela mão. Para isto é utilizado o algoritmo de detecção de bordas *Canny* junto com filtro de dilatação, seguido do algoritmo de balde para se criar uma máscara que irá ser subtraída da imagem original.
- **Implementação de um método de treinamento e classificação**: a partir das imagens tratadas, são extraídas informações para treinar o classificador a reconhecer um determinado sinal. Isso é feito no servidor pois seria muito custoso para um celular realizar um treinamento desse tipo. A classificação das imagens capturadas pela câmera do celular é feita a partir das informações contidas no arquivo de treinamento.

- **Desenvolvimento de um aplicativo móvel para tradução entre sinais**: o aplicativo móvel é responsável pela interação com o usuário no momento da tradução dos sinais. A tradução é realizada de acordo com as informações contidas no arquivo de mapeamento gerado pelo servidor.

Em conjunto, essas contribuições constituem um sistema que abrange diversos aspectos da comunicação dos surdos. Em relação à interação do usuário, existem duas interfaces (móvel e *web*), usadas em dois contextos diferentes. O aplicativo móvel insere-se no contexto da tradução *on-line*, ou seja, no momento que o surdo precisa se comunicar com um interlocutor surdo estrangeiro. Neste contexto, basta o usuário definir as duas línguas a serem utilizadas na tradução e acionar o aplicativo para capturar o sinal. Havendo uma conexão com a Internet disponível, o que é cada vez mais comum nos dias de hoje, o aplicativo retornará o resultado da consulta ao servidor, e toda a classificação ocorre de forma transparente ao usuário. Além da simplicidade da interface, é importante destacar que a proposta aqui apresentada não obriga o usuário a utilizar acessórios extra (como luvas ou marcadores ópticos) para que os sinais sejam reconhecidos. O uso de acessórios é uma limitação de trabalhos relacionados, pois inserem uma exigência ao usuário nem sempre simples de ser cumprida. No presente trabalho, o usuário necessita apenas de um aplicativo instalado em um *smartphone* com acesso à Internet, que a maioria das pessoas já possui.

O segundo contexto de interação é o da interface *web*, onde os usuários participam da construção da base de dados. Essa funcionalidade está alinhada com os preceitos da Web 2.0, que envolvem os usuários ativamente na produção de conteúdo, e não mais apenas como receptores passivos da informação disponível na *web*. Esse empoderamento (*empowerment*) do usuário funciona como uma motivação para participar de um projeto benéfico a uma comunidade. A proposta também se alinha com a técnica *crowdsourcing*, usada por aplicativos como o Waze, por exemplo, e que consiste em alimentar sistemas com informações fornecidas pelos usuários. Essas técnicas permitem a construção colaborativa de uma base de dados internacional que seria muito difícil de se atingir por uma pequena equipe de responsáveis pelo sistema. Através dessa abordagem, o sistema pode conter um número sempre crescente de línguas, de acordo com a nacionalidade dos usuários cadastrados e empenhados em melhorar a sua comunicação e de seus pares.

A Web 2.0 e o *crowdsourcing* apoiam-se em princípios de colaboração entre as pessoas, apostando que a comunidade de usuários envolvidos terá interesse em contribuir e não em prejudicar os sistemas. Ainda assim, algumas técnicas de moderação e controle podem ser usadas para prevenir ações prejudiciais e inserções indevidas de informações. O Waze, por exemplo, permite que usuários confirmem ou não informações colocadas por terceiros. Seguindo essa ideia, o sistema aqui proposto poderia submeter novas imagens de sinais enviadas à aprovação de um número mínimo de usuários cadastrados antes dela ser utilizada no treinamento, por exemplo.

8.2 Limitações e Trabalhos Futuros

Há algumas limitações neste trabalho que devem ser levadas em consideração para obter uma tradução correta. A primeira limitação ocorre na criação de um sinal na página *web*. Como o mapeamento entre sinais é feito através dos nomes dos sinais, eles precisam ser cadastrados com os mesmos nomes por usuários de diferentes países. Ou seja, o sistema atualmente só consegue fazer o mapeamento entre os sinais de diferentes linguagens se eles estiverem cadastrados com o mesmo nome. Essa limitação pode ser superada com a utilização de um módulo de tradução de palavras, por exemplo através de uma integração com a API do *Google Translate*.

Um desafio ainda maior refere-se aos surdos que não conhecem nenhuma outra língua que não seja sua linguagem de sinais local - por exemplo, um surdo brasileiro que não sabe Português, apenas LIBRAS. Para atender a esses casos, o sistema deveria ser expandido para representar os conceitos por imagens em vez de palavras. Por exemplo, para cadastrar o sinal de xícara, o usuário enviaria a imagem de uma xícara e o sinal correspondente em LIBRAS. Porém, trata-se de uma abordagem complexa já que nem todas as palavras podem ser facilmente representadas por imagens (conceitos abstratos como sentimentos, por exemplo).

Outra limitação ocorre por estarmos analisando somente sinais estáticos, que podem ser representados através de apenas uma imagem. Mas em linguagem de sinais há vários sinais que usam como complemento movimentos da própria mão e/ou do braço. Aos sinais que usam movimento, é dado o nome de sinais dinâmicos (como explicado na seção 2), que não foram abordados neste trabalho. Um exemplo deste

tipo de sinal é a letra Z do alfabeto que é feita com o dedo indicador desenhando o formato desta letra no ar. Como trabalho futuro, podemos incorporar a classificação desses tipos de sinais fazendo uma modificação no sistema de arquivo para os sinais. Como para cada sinal há uma pasta com várias imagens, cada exemplo de sinal dinâmico pode ser tratado como um sinal estático, já que ele consiste de várias imagens também. Neste caso vai haver várias pastas para um mesmo sinal.

Além disso, foi visto na seção 2 que o uso de expressões faciais é importante para uma linguagem de sinal pois dá um sentido a mais a um sinal e a uma frase. É a expressão facial que dá sentimento de raiva, tristeza, felicidade, exclamação e dúvida. Como trabalho futuro, pode-se fazer um estudo de reconhecimento de expressões faciais juntamente com o reconhecimento dos sinais.

Uma outra limitação refere-se ao plano de fundo nos módulos servidor e móvel. No módulo servidor, quando um usuário faz o envio de uma imagem para um determinado sinal, assume-se que a imagem tenha um fundo homogêneo. Em caso de um plano de fundo que não seja homogêneo, os resultados não serão ideais prejudicando assim a classificação. O módulo móvel apresenta uma limitação quanto aos planos de fundo complexos. Os resultados apresentam instabilidade e pode ser complicado identificar a imagem da mão inteira. Esta variação de resultado pode influenciar o resultado final com uma classificação errada. Outra limitação do módulo móvel é devida à instabilidade ao segurar o dispositivo. Os testes realizados neste trabalho foram feitos com uma base para segurar o celular a fim de mantê-lo o mais estável possível. Outro ponto de limitação leva em consideração a constante mudança no plano de fundo complexo. Toda vez há uma modificação, o aplicativo perde a referência do plano de fundo anterior, prejudicando assim a remoção, precisando a cada mudança, haver uma atualização do plano de fundo. Um estudo de remoção do plano de fundo mais aprofundado deve ser feito, incluindo outras técnicas, para melhorar estas limitações.

Um outro trabalho futuro que pode melhorar a taxa de acerto da classificação é a retenção das informações das imagens que tiveram sua classificação correta através de *feedback* do usuário. Poderia funcionar da seguinte forma: a cada *feedback* positivo do usuário a respeito de uma classificação, o sistema poderia adicionar as informações das imagens corretas ao treinamento e treinar novamente a rede. Isto poderia aumentar a taxa de acerto significativamente já que o treinamento seria baseado em exemplos classificados com sucesso.

Por fim, como este sistema foi feito para a comunicação entre surdos, ele pode ser expandido para realizar uma comunicação entre surdos e não-surdos traduzindo assim uma linguagem de sinal para texto e vice-versa. Ainda neste ponto, pode-se integrar com um módulo *text-to-speech* como já existe em celulares, sistemas operacionais e em algumas outras aplicações.

Podemos concluir que este trabalho propõe uma abordagem diferente dos demais, que é uma tradução entre linguagens de sinais, e mostra que muita coisa pode ainda ser melhorada a partir daqui. Poder saber como é um sinal em uma outra linguagem, a partir do sinal em sua própria linguagem, traz facilidades para uma pessoa e ainda pode ser uma forma para aprender uma nova linguagem. Além disso, a possibilidade dos próprios usuários fornecerem imagens dos sinais em diversas linguagens, criando novos sinais ou melhorando o reconhecimento de sinais já existentes, pode ser um fator de incentivo para colaborar com a melhoria da comunicação entre as pessoas surdas e um fator motivacional para melhorar os sinais constantemente.

Referências

a) Opencv Dev Team. (2 de Agosto de 2015). ***Background Subtraction***. Fonte: opencv: http://docs.opencv.org/master/db/d5c/tutorial_py_bg_subtraction.html

b) OpenCV Dev Team. (25 de Fevereiro de 2015). ***Canny Edge Detector***. Fonte: opencv: http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html

c) Opencv Dev Team. (25 de Fevereiro de 2015). ***Cascade Classifier Training***. Fonte: opencv: http://docs.opencv.org/doc/user_guide/ug_traincascade.html

d) Opencv Dev Team. (25 de Fevereiro de 2015). ***Introduction to Support Vector Machines***. Fonte: opencv: http://docs.opencv.org/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

E) Opencv Dev Team. (25 de Fevereiro de 2015). ***Miscellaneous Image Transformations***. Fonte: opencv: http://docs.opencv.org/modules/imgproc/doc/miscellaneous_transformations.html

F) Opencv Dev Team. (25 de Fevereiro de 2015). ***Support Vector Machines for Non-Linearly Separable Data***. Fonte: opencv: http://docs.opencv.org/doc/tutorials/ml/non_linear_svms/non_linear_svms.html

ALMEIDA, M. P., & ALMEIDA, M. E. (2012). **HISTÓRIA DE LIBRAS: CARACTERÍSTICA E SUA ESTRUTURA**. *Revista Philologus, Ano 18, nº 54 –*

Suplemento: Anais da VII JNLFLP. Rio de Janeiro: CiFEFiL, pp. 315-327. Disponível em http://www.filologia.org.br/revista/54supl/031.pdf.

Android, G. (2 de Agosto de 2015). ***Managing Your App's Memory***. Fonte: developer.android.com: https://developer.android.com/training/articles/memory.html

ARULAPALAM, M. S., MASKELL, S., GORDON, N., & CLAPP, T. (Agosto de 2001). **A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking**. *IEEE Transactions on Signal Processing, Volume 50, nº 2*, pp. 174-188. Disponível em http://www.cse.psu.edu/~rtc12/CSE598C/arulampalamTutorialPF.pdf.

BARROS, P. V., JÚNIOR, N. T., BISNETO, J. M., FERNANDES, B. J., BEZERRA, B. L., & FERNANDES, S. M. (Setembro de 2013). **An Effective Dynamic Gesture Recognition System Based on the Feature Vector Reduction for SURF and LCS**. *Artificial Neural Networks and Machine Learning – ICANN 2013. Proceedings of the 23rd International Conference on Artificial Neural Networks*, pp. 412-419. Disponível em http://www.academia.edu/10951426/An_Effective_Dynamic_Gesture_Recognition_System_Based_on_the_Feature_Vector_Reduction_for_SURF_and_LCS.

BARROS, P. V., JUNIOR, N. T., BISNETO, J. M., FERNANDES, B. J., BEZERRA, B. L., & FERNANDES, S. M. (2013). **Convexity local contour sequences for gesture recognition**. *SAC '13 Proceedings of the 28th Annual ACM Symposium on Applied Computing* (pp. 34-39). New York, NY, USA: ACM.

BAY, H., TUYTELLAARS, T., & GOOL, L. V. (Junho de 2008). **SURF: Speeded Up Robust Features**. *Computer Vision and Image Understanding, Volume 110, nº 3*, pp. 346-359. Disponível em http://www.vision.ee.ethz.ch/~surf/eccv06.pdf.

BEZERRA, B. L., ZANCHETTIN, C., & ANDRADE, V. B. (2012). **A MDRNN-SVM Hybrid Model for Cursive Offline Handwriting Recognition**. *Artificial Neural Networks and Machine Learning – ICANN 2012 - Lecture Notes in*

Computer Science (pp. 246-254). Lausanne, Switzerland: Springer Berlin Heidelberg.

BLACK, M. J., & JEPSON, A. D. (Junho de 1998). **A Probabilistic Framework For Matching Temporal Trajectories: Condensation-based recognition**. *Computer Vision - ECCV'98 - Proceedings of the 5th European Conference on Computer Vision Freiburg, Germany, Volume I*, pp. 909-924. Disponível em http://files.is.tue.mpg.de/black/papers/Black-1998-PFM.pdf.

BOBICK, A. F., & WILSON, A. D. (Agosto de 2002). **A state-based approach to the representation and recognition of gesture**. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume 19, nº 12*, pp. 1325-1337.

BRASIL. (2005). *Decreto nº 5.626, de 22 de dezembro de 2005. Regulamenta a lei no 10.436, de 24 de abril – Brasília. MEC/SESSP 2005.*

BRECAILO, S. F. (2012). ***Expressão Facial e Corporal na comunicação em LIBRAS***. Curitiba. Disponível em http://www.imap.curitiba.pr.gov.br/wp-content/uploads/2014/03/apostila_curso_expressao_corporal%20%281%29.pdf.

CAI, J., & GOSHTASBY, A. (Fevereiro de 1999). **Detecting human faces in color images**. *ELSEVIER - Image and Vision Computing 18*, pp. 63-75. Disponível em http://cecs.wright.edu/~agoshtas/facechroma.pdf.

CHENG, M.-M., MITRA, N. J., HUANG, X., TORR, P. H., & HU, S.-M. (Junho de 2011). **Global Contrast based Salient Region detection**. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 409-416. Disponível em http://mmcheng.net/mftp/Papers/SaliencyTPAMI.pdf.

DAVIS, J., & SHAH, M. (2002). **Visual Gesture Recognition**. *IEE Proceedings of the Vision, Image and Signal Processing, Volume 141, nº 2* (pp. 101-106). IET. Disponível em http://crcv.ucf.edu/papers/shah/94/DAS94.pdf.

DAY, A. M. (Setembro de 1988). **Planar convex hull algorithms in theory and practice**. *Computer Graphics Forum*, pp. 177-193.

DEUSDADO, L. D. (2002). ***Ensino da Língua Gestual Assistido por Personagens 3D Virtuais***. Portugal, Disponível em http://sim.di.uminho.pt/mestrados/deusdado/teseLeonel.pdf: UNIVERSIDADE DO MINHO DEPARTAMENTO DE INFORMÁTICA.

FALCÃO, L. A. (2007). ***Aprendendo a LIBRAS e reconhecendo as diferenças: uma proposta de intervenção na formação profissional de ouvintes***. UFRJ. Disponível em http://www.facol.com/gestus/artigos/artigo6-completo.htm: VIII Congresso Brasileiro de Linguistica Aplicada.

FALCÃO, L. A. (2010). ***Surdez, Cognição Visual e Libras. Estabelecendo Novos Diálogos***. Recife: Luiz Albérico.

FALCÃO, L. A. (2012). ***EDUCAÇÃO DE SURDOS E A FORMAÇÃO DOCENTE NA PERSPECTIVA INCLUSIVA - O CASO CONCRETO DO PARAGUAY***. Assunção - Paraguai. Disponível em http://www.cultura-sorda.org/wp-content/uploads/2015/04/TESIS-BarbosaFalcao-Doctorado-2012.pdf: Universidad Americana.

FALCÃO, L. A. (s.d.). ***PRÁTICA DOCENTE NA EDUCAÇÃO DE PESSOAS SURDAS ATRAVÉS DA COGNIÇÃO E DA DESCRIÇÃO VISUAL SINALIZADA: ENSAIOS PEDAGÓGICOS***. Recife, PE. Disponível em http://visaoinclusiva.com.br/PDFS/A%20TECNICA%20DA%20DESCRICAO%20VISUAL%20SINALIZADA.pdf: Visão Inclusiva.

FELIPE, T. A. (2013). **The verbal-visual discourse in Brazilian sign language - Libras**. *Bakhtiniana, Rev. Estud. Discurso [online], Volume 8, nº 2*, pp. 67-89. Disponível em http://www.scielo.br/pdf/bak/v8n2/en_05.pdf.

FIBIGER, R. S. (2004). ***Estudo Comparativo de Técnicas de Visão Computacional para Detecção de Pele Humana***. Campo Grande - MS. Disponível em http://www.gpec.ucdb.br/pistori/orientacoes/monografias/roger2004.pdf: Universidade Católica Dom Bosco.

GONÇALVES, A. R. (2010). **Máquina de Vetores Suporte**. Disponível em http://www-users.cs.umn.edu/~andre/arquivos/pdfs/svm.pdf.

GOUVEIA, M. F., BEZERRA, B. L., ZANCHETTIN, C., & MENESES, J. R. (2014). **Handwriting recognition system for mobile**. *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on* (pp. 3918-3923). San Diego, CA: IEEE.

GUNNAR, J. (Junho de 1973). **Visual Perception of Biological Motion and a Model for Its Analysis**. *Perception and Psychophysics, Volume 14, nº 2*, pp. 201-211.

GUPTA, L., & MA, S. (Fevereiro de 2001). **Gesture-based interaction and communication: automated classification**. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Volume 31, nº 1*, pp. 114-120.

HONG, P., TURK, M., & HUANG, T. S. (Março de 2000). **Gesture modeling and recognition using finite state machines**. *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 410-415.

HUANG, G.-B. (Setembro de 2014). **An Insight into Extreme Learning Machines: Random Neurons, Random Features and Kernels**. *Cognitive Computation, Volume 6, nº 3*, pp. 376-390. Disponível em http://www3.ntu.edu.sg/home/egbhuang/pdf/ELM-Randomness-Kernel.pdf.

HUANG, G.-B., ZHOU, H., DING, X., & ZHANG, R. (Abril de 2012). **Extreme Learning Machine for Regression and Multiclass Classification**. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART B: CYBERNETICS, Volume 42, nº 2*, pp. 513-29. Disponível em http://www3.ntu.edu.sg/home/egbhuang/pdf/ELM-Unified-Learning.pdf.

HUANG, G.-B., ZHU, Q.-Y., & SIEW, C.-K. (Dezembro de 2006). **Extreme learning machine: Theory and applications**. *Neurocomputing, Volume 70, nº 1-3*, pp. 489-501.

ISARD, M., & BLAKE, A. (Agosto de 1998). **CONDENSATION—Conditional Density Propagation for Visual Tracking**. *International Journal of Computer*

Vision, Volume 29, nº 1, pp. 5-28. Disponível em http://research.microsoft.com/pubs/66183/isard-blake-98.pdf.

JÚNIOR, N. T., BARROS, P. V., FERNANDES, B. J., BEZERRA, B. L., & FERNANDES, S. M. (2013). A Dynamic Gesture Prediction System Based on. *Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI & CBIC), 2013 BRICS Congress on* (pp. 501-506). Ipojuca: IEEE.

LISETTI, C. L. (2000). **Automatic classification of single facial images**. *Pragmatics and Cognition, Volume 8*, pp. 185-235.

LOWE, D. (1989). **Adaptive radial basis function nonlinearities and the problem of generalisation**. *Proceedings of first IEE international conference on artificial neural networks, Volume 1*, pp. 171-175.

MCKENNA, S. J., & GONG, S. (Março de 1999). **Gesture recognition for visually mediated interaction using probabilistic event trajectories**. *Proceedings of the Gesture-Based Communication in Human-Computer Interaction - International GestureWorkshop, GW'99 Gif-sur-Yvette, France*, pp. 141-151. Disponível em http://www.bmva.org/bmvc/1998/pdf/p107.pdf.

MITRA, S., & ACHARYA, T. (2003). ***Data Mining: Multimedia, Soft Computing, and Bioinformatics***. New York: Wiley.

MITRA, S., & ACHARYA, T. (Maio de 2007). **Gesture Recognition: A Survey**. *IEEE Transactions Systems, Man and Cybernetics, Volume 37, nº 3*, pp. 311-324.

MOORE, G. E. (19 de April de 1965). **Cramming more components onto integrated circuits**. *Electronics, Volume 38, nº 8*, pp. 4, Disponível em http://www.monolithic3d.com/uploads/6/0/5/5/6055488/gordon_moore_1965_article.pdf.

MORDVINTSEV, A., & ABID, K. (31 de Out de 2014). ***Background Subtraction***. Fonte: opencv-python-tutroals.readthedocs: http://opencv-python-

tutroals.readthedocs.org/en/latest/py_tutorials/py_video/py_bg_subtraction/py_bg_subtraction.html

NETO, F. M., LUDERMIR, T. B., & ZANCHETTIN, C. (2015). **Extreme learning machine for real time recognition of brazilian sign language**. *Artigo a ser aceito na ACM*.

NOBUYUKI, O. (Janeiro de 1979). **A Threshold Selection Method from Gray-Level Histograms**. *IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS, Volume 9, nº 1*, pp. 62-66. Disponível em http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4310076.

OLIVEIRA, V. A., & CONCI, A. (2009). **Skin Detection using HSV color space**. *Workshops of Sibgrapi*, Disponível em http://www.matmidia.mat.puc-rio.br/sibgrapi/media/posters/59928.pdf.

O'ROURKE, J., & BADLER, N. (Novembro de 1980). **Model-based image analysis of human motion using constraint propagation**. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume:PAMI-2, nº 6*, pp. 522-536.

OSCARSSON, E. (1982). **TV-Camera Detecting pedestrians for traffic light control**. *Technological and Methodological Advances Measurement (ACTA IMEKO), Volume 3*, pp. 275-282.

ROSEBROCK, A. (Abril de 2015). ***Zero-parameter, automatic Canny edge detection with Python and OpenCV***. Fonte: pyimagesearch: http://www.pyimagesearch.com/2015/04/06/zero-parameter-automatic-canny-edge-detection-with-python-and-opencv/

SANTOS, D. G., NETO, R. C., FERNANDES, B. J., & BEZERRA, B. L. (Julho de 2014). **A Dynamic Gesture Recognition System based on CIPBR Algorithm**. *Polibits nº 50*, pp. 13-19. Disponível em http://www.scielo.org.mx/pdf/poli/n50/n50a3.pdf.

SANTOS, D. G., NETO, R. C., FERNANDES, B., & BEZERRA, B. (Setembro de 2015). **FSSGR: Feature Selection System to Dynamic Gesture**

Recognition. *New Trends in Image Analysis and Processing - ICIAP 2015 Workshops, Volume 9281 of the series Lecture Notes in Computer Science*, pp. 234-241.

SOARES, L. A. (2014). ***UM ESTUDO DE DOMÍNIO DA LÍNGUA BRASILEIRA DE SINAIS A FIM DE COLHER REQUISITOS PARA A CRIAÇÃO DE UM MODELO COMPUTACIONAL DESCRITIVO DESSE IDIOMA***. Recife, Disponível em http://www.cin.ufpe.br/~in1020/arquivos/monografias/2013_2/lucas.pdf: UFPE - UNIVERSIDADE FEDERAL DE PERNAMBUCO.

SORIA, D. J., SCHAEFFER, S. E., & VILLAREAL, S. E. (Junho de 2013). **Hand-gesture recognition using computer-vision techniques**. *WSCG 2013: Communication Papers Proceedings: 21st International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision in co-operation with EUROGRAPHICS Association*, pp. 1-8. Disponível em http://wscg.zcu.cz/WSCG2013/!_2013-WSCG-Communications-proceedings.pdf.

StackOverflow. (2 de Agosto de 2015). ***Detect application heap size in Android***. Fonte: stackoverflow: http://stackoverflow.com/questions/2630158/detect-application-heap-size-in-android/9428660#9428660

StatSoft. (2015). ***Support-Vector-Machines***. Fonte: http://www.statsoft.com: http://www.statsoft.com/Textbook/Support-Vector-Machines

STROBEL, K. (2009). ***HISTÓRIA DA EDUCAÇÃO DE SURDOS***. FLORIANÓPOLIS. Disponível em http://www.libras.ufsc.br/colecaoLetrasLibras/eixoFormacaoEspecifica/historiaDaEducacaoDeSurdos/assets/258/TextoBase_HistoriaEducacaoSurdos.pdf: Universidade Federal de Santa Catarina - Licenciatura em Letras-LIBRAS na modalidade a distância.

SUCHARITA, V., JYOTHI, S., & MAMATHA, D. M. (2013). **Evaluation of the Digital images of Penaeid Prawns Species Using Canny Edge Detection**

and Otsu Thresholding Segmentation. *International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS), Volume 6, nº 2*, pp. 117-121. Disponível em http://www.academia.edu/5406421/Evaluation_of_the_Digital_images_of_Penaeid_Prawns_Species_Using_Canny_Edge_Detection_and_Otsu_Thresholding_Segmentation.

VAPNIK, V., & CORTES, C. (Setembro de 1995). **Support-Vector Networks.** *Machine Leaming, Volume 20, nº 3*, pp. 273-297. Disponível em http://download.springer.com/static/pdf/467/art%253A10.1007%252FBF00994018.pdf?originUrl=http%3A%2F%2Flink.springer.com%2Farticle%2F10.1007%2FBF00994018&token2=exp=1443929083~acl=%2Fstatic%2Fpdf%2F467%2Fart%25253A10.1007%25252FBF009.

WANG, R. Y. (2008). **Real-Time Hand-Tracking as a User Input Device**. *ACM Symposium on User Interface Software and Technology (UIST)*, Disponível em http://people.csail.mit.edu/rywang/handtracking/uist.pdf.

YAMATO, J., OHYA, J., & ISHII, K. (1992). **Recognizing human action in time-sequential images using hidden Markov model**. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Proceedings of the CVPR '92* (pp. 379-385). Champaign, IL: IEEE. Disponível em http://www.ee.iisc.ernet.in/new/people/faculty/soma.biswas/STIP_pdf/Yamato-HumanAction.pdf.

ZIVKOVIC, Z. (2004). **Improved Adaptive Gaussian Mixture Model for Background Subtraction**. *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004, Volume 2* (pp. 28-31). IEEE. Disponível em http://www.zoranz.net/Publications/zivkovic2004ICPR.pdf.

ZIVKOVIC, Z., & HEIJDEN, F. V. (Maio de 2006). **Efficient adaptive density estimation per image pixel for the task of background subtraction**. *Pattern Recognition Letters, Volume 27, nº 7*, pp. 773-780. Disponível em http://www.zoranz.net/Publications/zivkovicPRL2006.pdf.