UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PIETRO MARTINS DE OLIVEIRA

Uma Interface Para Geração de Comandos Em Tempo Real Para Um Robô Móvel com Rodas Baseada em Fluxo Ótico e Reconhecimento de Características Faciais

Maringá
2015

PIETRO MARTINS DE OLIVEIRA

Uma Interface Para Geração de Comandos Em Tempo Real Para Um Robô Móvel com Rodas Baseada em Fluxo Ótico e Reconhecimento de Características Faciais

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Franklin César Flores
Coorientador: Prof. Dr. Nardênio Almeida Martins

Maringá
2015

Dados Internacionais de Catalogação na Publicação (CIP)
(Biblioteca Central - UEM, Maringá, PR, Brasil)

O48i

Oliveira, Pietro Martins de
Uma interface para geração de comandos em tempo real para um robô móvel com rodas baseada em fluxo ótico e reconhecimento de características faciais / Pietro Martins de Oliveira. -- Maringá, 2015.
76 f. : il. : color., figs., tabs.

Orientador: Prof. Dr. Franklin César Flores.
Coorientador: Prof. Dr. Nardênio Almeida Martins.
Dissertação (mestrado)- Universidade Estadual de Maringá, Centro de Tecnologia, Departamento de Informática, Programa de Pós-Graduação em Ciência da Computação, 2005.

1. Interface humano – Computador. 2. Robô móvel. 3. Cadeiras de rodas inteligentes. I. Flores, Franklin César, orient. II. Martins, Nardênio Almeida, coorient. III. Universidade Estadual de Maringá. Centro de Tecnologia. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação IV. Título.

CDD21.ed.025.284

MGC-001837

FOLHA DE APROVAÇÃO

PIETRO MARTINS DE OLIVEIRA

Uma interface para geração de comandos em tempo real para um robô móvel com rodas baseada em fluxo ótico e reconhecimento de características faciais

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação pela Banca Examinadora composta pelos membros:

BANCA EXAMINADORA

Prof. Dr. Franklin César Flores
Universidade Estadual de Maringá – DIN/UEM

Prof. Dr. Yandre Maldonado e Gomes da Costa
Universidade Estadual de Maringá – DIN/UEM

Profa. Dra. Leticia Rittner
Universidade Estadual de Campinas – DCA/UNICAMP

Aprovada em: 21 de agosto de 2015.
Local da defesa: Sala 101, Bloco C56, *campus* da Universidade Estadual de Maringá.

AGRADECIMENTOS

Agradeço por todos aqueles que sempre me apoiaram e deram sua contribuição para que este trabalho pudesse ser finalizado. De maneira especial:

Aos meus pais Rosangela Maria Martins e Pascoal José Oliveira, que me passaram toda a fundamentação primordial de ser humano.

Aos meus professores Dr. Franklin César Flores e Dr. Nardênio Almeida Martins, sou grato pela oportunidade de ter aprendido como se leciona e faz pesquisa, sem perder a maneira positiva de encarar as adversidades.

Aos colegas de classe, sem os quais não seria possível superar alguns limites, e atingir objetivos grandiosos.

A cada amigo que se prontificou a realizar os experimentos, dando sua contribuição na geração de resultados, obrigado pela paciência e disposição.

Agradeço, à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro concedido a este trabalho, indispensável ao desenvolvimento científico.

Resumo

A reabilitação e melhoria das condições de vida de portadores de necessidades motoras especiais oferecem muitas possibilidades de pesquisa. O presente trabalho tem o objetivo de desenvolver uma nova interface baseada em fluxo ótico e reconhecimento de expressões faciais, com o intuíto de permitir que uma pessoa, que possua os movimentos do pescoço para cima, comande uma cadeira de rodas elétrica. Essa interface faz uso de uma *webcam* voltada para o rosto do usuário, para capturar uma sequência de imagens as quais são processadas com a finalidade de reconhecer padrões faciais que sirvam como comandos para que o sistema realize alguma ação em tempo real. Essa ação é enviada a um robô que representa a cadeira de rodas. O projeto é dividido em duas partes, a primeira denominada Módulo de Visão Computacional (MVC) e a outra é o Módulo Robótico (MR). O primeiro módulo aplica técnicas de visão computacional para reconhecer as expressões do rosto, e movimentos da cabeça, interpretando as intenções de se locomover do usuário, e assim, enviando comandos para o robô. O segundo módulo é um Robô Móvel com Rodas (RMR) que possui grau de manobrabilidade igual a (2,0), o qual é o mesmo grau que possui uma cadeira de rodas. Esse robô recebe os comandos gerados pelo MVC por meio de uma interface de comunicação *bluetooth*, e então realiza os movimentos que uma cadeira de rodas faria em uma situação real. Foram realizados testes envolvendo diferentes tipos de iluminação ambiente, e também pessoas com tons de pele variados. Os resultados experimentais mostram que é possível comandar o movimento de um RMR pela analise de imagens que capturam os movimentos da cabeça e expressões do rosto. Ambientes internos apresentaram altas taxas no reconhecimento correto, e também baixíssimas taxas de falso reconhecimento de comandos.

Palavras-chaves: cadeiras de rodas inteligentes, robô móvel com rodas, interface homem-máquina.

Abstract

Rehabilitation and improvement of living conditions of people with special motor needs offers many research possibilities. This study has the goal of developing a new interface based on optical flow and facial expressions recognition, aiming to enable people who are able to move muscles above the neck to command an electric wheelchair. This interface uses a webcam directed to the face of the user, in order to capture a sequence of images, which are processed aiming to recognize facial patterns that serve as commands for the system to perform some action in real time. This action is sent to a robot that represents the wheelchair. The system is divided into two modules, the first named Computer Vision Module MVC and the other is the Robotic Module MR. The first module applies computer vision techniques to recognize facial expressions and head movements, interpreting the intentions of moving from the user, and thus, sending commands to the robot. The second module is a wheeled mobile robot RMR which has its degree of maneuverability equal to (2,0), which are the same degrees of an wheelchair. This robot receives the commands generated by the MVC by means of a bluetooth communication interface, and then performs the movements that a wheelchair would do in a real situation. Tests were conducted involving various types of environment lighting, and also people with different skin tones. The experimental results show that it is possible to control the movements of an RMR by means of the analysis of images that captures the head movements and facial expressions. Indoor environments showed high rates in the correct recognition, and also very low false command recognition rates.

Keywords: intelligent wheelchair; wheeled mobile robot; human-machine interface.

Lista de ilustrações

Lista de tabelas

Lista de siglas e abreviaturas

API Application Programming Interface 19

FO Fluxo Ótico 18–20, 22, 27, 29, 32, 34, 38–40, 46, 48, 56, 59, 61

IBGE Instituto Brasileiro de Geografia e Estatística 15

ICC Interface Comandada por Cérebro 15

ICSP In-circuit Serial Programming 37

MHI Motion History Images 8, 10, 22, 32–34, 46, 47, 56, 58, 59, 61

MR Módulo Robótico 5, 6, 19, 38, 40, 49, 50, 57

MVC Módulo de Visão Computacional 5–7, 19, 38–40, 48, 49, 57, 60, 61

NF Negativos Falsos 54, 55

NV Negativos Verdadeiros 55

PF Positivos Falsos 55

PV Positivos Verdadeiros 54, 55

RMR Robô Móvel com Rodas 5, 6, 8, 18–20, 22, 35, 36, 38, 40, 49, 50, 52, 57, 58, 60

SIMD Single Instruction Multiple Data 35

TNF Taxa de Negativos Falsos 55

TNV Taxa de Negativos Verdadeiros 55–57, 59

TPF Taxa de Positivos Falsos 55

TPV Taxa de Positivos Verdadeiros 54–57, 59

USB Universal Serial Bus 36

Sumário

1 Introdução

No último censo realizado pelo Instituto Brasileiro de Geografia e Estatística (IBGE) os portadores de necessidades motoras especiais foram divididos em quatro categorias levando em consideração a capacidade da pessoa caminhar e/ou subir escadas: Pessoas com Nenhuma Dificuldade; Alguma Dificuldade; Grande Dificuldade e; aquelas que Não Conseguem de Modo Algum. No último caso, para que a pessoa possa se locomover, sempre há a necessidade da ajuda de outra neste processo. Esse censo aponta que cerca de 730000 pessoas não conseguem se locomover de maneira autônoma no Brasil, isto representa 0,3% da população (IBGE, 2010).

Um dos produtos que mais auxiliam pessoas com dificuldade de locomoção é a cadeira de rodas. Pode-se enquadrar essas cadeiras em duas categorias: manuais e elétricas. Para impulsionar e direcionar uma cadeira manual é preciso que o usuário aplique a força de seus braços. É necessário ter habilidade com a cadeira além de força nos braços. De maneira geral, as cadeiras elétricas normalmente são equipadas com um *joystick* que é operado manualmente, para comandar o movimento da cadeira, livrando o usuário da aplicação da força de seus braços. As limitações de alguns usuários, contudo, são ainda mais severas, como nos casos onde há ausência de movimentos abaixo do pescoço, o que encoraja o desenvolvimento de interfaces alternativas para a utilização de uma cadeira elétrica.

Uma breve busca na internet revela que cadeiras de rodas motorizadas sem suporte para pessoas sem os movimentos dos braços, além das pernas, quando compradas novas, podem custar de R$ 6000,00 até R$20000,00. Encontrar de cadeiras de rodas próprias para tetraplégicos é um pouco mais difícil que as convencionais, sendo que essas geralmente são comandadas através de um *joystick* operado pelo queixo. Em uma busca realizada em março de 2015, foi encontrada uma cadeira de rodas com acionamento pelo queixo, que custa cerca de R$ 10000,00. A pouca variedade de tipos de controle da cadeira para tetraplégicos, e o custo elevado deste tipo de artigo, são outros fatores que indicam a necessidade de investimento em pesquisas nessa área.

1.1 Trabalhos Relacionados

Diversos trabalhos encontrados na literatura propõem maneiras de aprimorar o funcionamento de cadeiras elétricas para usuários que possuem apenas o movimento dos músculos acima do pescoço, propondo a substituição do *joystick* tradicional. Alguns exemplos são: Interface Comandada por Cérebro (ICC) (LOPES *et al.*, 2011; BASTOS-FILHO *et al.*, 2013a), reconhecimento de comandos de voz (ALMEIDA, 2007), eletro-oculografia e eletromiografia (MOON *et al.*, 2003; HAN *et al.*, 2003; BASTOS-FILHO *et*

al., 2013a), controle por sucção e sopro (PARANHOS; KIYOKY; FILHO, 2012; BASTOS-FILHO *et al.*, 2013a), sensores de pressão presos à cabeça (KUPETZ; WENTZELL; BUSHA, 2010), acelerômetro preso à cabeça (MANOGNA; VAISHNAVI; GEETHANJALI, 2010; PAJKANOVIC; DOKIC, 2013) e, por fim, utilizando visão computacional para interpretar as intenções de movimento do usuário.

Alguns desses trabalhos são bastante conhecidos, como a cadeira de rodas inteligente Wheelesley que apresenta um sistema de navegação para ambientes internos, além de uma interface com o usuário para que este possa tomar decisões sobre a trajetória desejada (YANCO, 1998). Destaca-se ainda, o projeto NavChair, que para diminuir o esforço do usuário da cadeira, emprega um sistema composto por sensores que auxiliam no desvio de obstáculos, atravessar portas e seguir paredes, e também permite o controle de movimento compartilhado pelo uso de um *joystick* (LEVINE *et al.*, 1999).

Vários outros trabalhos descrevem projetos de cadeira de rodas inteligentes que dão prioridade ao controle de trajetória e navegação, aliados a algum tipo de interface. Em (ESCOBEDO *et al.*, 2012), os autores utilizam um sistema de navegação baseado em mapas, em conjunto com um sensor Microsoft Kinect voltado para o rosto do usuário. Neste projeto o Kinect é responsável por realizar o rastreamento da face e reconhecer comandos de voz, e por meio de uma rede bayesiana classifica-se as leituras do sensor e as traduz em comandos de alto nível para a cadeira de rodas. Já o trabalho de (TOMARI; KOBAYASHI; KUNO, 2012), utiliza o sensor Kinect voltado para o ambiente, com o intuito de criar um mapa de navegação segura. Para realizar a interface entre cadeira e o cadeirante, (TOMARI; KOBAYASHI; KUNO, 2012) utilizam uma câmera para medir a inclinação da cabeça do usuário e assim comandar a trajetória.

Recentemente, (BASTOS-FILHO *et al.*, 2013b) apresentaram uma cadeira de rodas portando um sistema que consiste em uma interface multi-modal na qual comandos são gerados à partir das piscadas dos olhos do usuário, movimentos dos olhos, movimentos da cabeça, por sopro-e-sucção e também através de sinais cerebrais. A cadeira de rodas também pode operar como um veículo autônomo, seguindo fitas metálicas e utilizando seus sensores. Os experimentos realizados pelos autores validam o sistema desenvolvido, mostrando que a redundância de sensores e técnicas de navegação autônoma melhoram dramaticamente a usabilidade, por outro lado podem adicionar complexidade computacional e elevar os custos. Outros trabalhos como (MAZO, 2001), também apresentam interfaces que incluem vários tipos de sensores, procurando obter modularidade para contornar os diferentes tipos de dificuldades de cada usuário.

O Trabalho apresentado em (PAJKANOVIC; DOKIC, 2013) desenvolve uma técnica na qual um acelerômetro é usado para processar os dados oriundos do movimento da cabeça do usuário para controlar os atuadores mecânicos da cadeira de rodas. Seus resultados mostram que os comandos puderam ser executados com 95,16% de sucesso, e sinalizam que a utilização de sensores adicionais melhora a habilidade do sistema em

reconhecer comandos.

Existem vários tipos de interfaces de controle na literatura. Em (KUPETZ; WENTZELL; BUSHA, 2010), o usuário da cadeira de rodas usa um boné, cuja parte de trás carrega um arranjo com LEDs infravermelhos. A movimentação da cadeira é realizada por um sistema que traduz a posição da cabeça do usuário em velocidade e controle direcional, através de uma câmera. Há também um sensor de pressão que ativa os freios.

Entre as abordagens que aplicam visão computacional para comandar uma cadeira de rodas encontradas na literatura, existem trabalhos que utilizam segmentação de pele (YU; CHEN; KUO, 2012), redes neurais artificiais (MARTINS; VALGôDE, 2006; FARIA *et al.*, 2007), classificadores supervisionados e não-supervisionados (FERNANDES; SILVA; PISTORI, 2005; WEI; HU, 2011; FIA *et al.*, 2007; MAZO, 2001), rastreamento do movimento dos olhos (YANCO, 1998; CARLSON; DEMIRIS, 2012; CARLSON; DEMIRIS, 2010; RASCANU; SOLEA, 2011; BASTOS-FILHO *et al.*, 2013a; GAJWANI; CHHABIRA, 2010), monitoramento do centróide da face (KUNO; SHIMADA; SHIRAI, 2003) e visão estéreo (MATSUMOTO; INO; OGASAWARA, 2001).

Em (WEI; HU, 2011) descreve-se uma interface homem-máquina que integra eletromiografia e imagens capturadas por uma câmera presa à parte frontal da cadeira de rodas, apontando para a face do usuário. As imagens são processadas por uma combinação dos algoritmos *Adaboost* e *Camshift* para detecção de faces e reastreamento de objetos. Além disso, a interface monitora a atividade de alguns músculos faciais por meio de eletrodos para detectar as piscadas dos olhos, como uma alternativa para a substituição do *joystick* convencional.

É comum encontrar trabalhos que utilizam visão computacional para rastrear os olhos, usando este artifício como interface de comandos. Os objetivos de (CARLSON; DEMIRIS, 2012; CARLSON; DEMIRIS, 2010) são avaliar a performance, atenção e carga de trabalho dos usuários de uma cadeira de rodas que apresenta um tipo de controle colaborativo. Este sistema de cadeira de rodas apresenta rotinas de controle de trajetória que podem ser comandadas via rastreamento dos olhos, ou um *joystick* convencional. O trabalho (RASCANU; SOLEA, 2011) utiliza rotinas implementadas em MATLAB e LabVIEW para determinar as posições das íris do usuário e, assim, gerar comandos de navegação para a cadeira. Estes trabalhos recém mencionados não apresentam informações sobre as condições de iluminação ou sombra em seus testes. Outras abordagens baseadas em rastreamento das íris são apresentadas em (BASTOS-FILHO *et al.*, 2013b; GAJWANI; CHHABIRA, 2010; BASTOS-FILHO *et al.*, 2013a).

Em (KUNO; SHIMADA; SHIRAI, 2003), duas câmeras são usadas para dar suporte à locomoção de uma cadeira de rodas. Uma câmera é voltada em direção ao ambiente, e aliada a sensores ultrassônicos e rotinas de navegação, têm o intuíto de mensurar a localização do usuário em um dado momento, podendo detectar obstáculos ou objetos de interesse. A outra câmera é fixada na cadeira apontando para a face do usuário, para

computar um centróide da face da pessoa e então gerar comandos de movimentação. À medida que este centróide se desloca, os movimentos da cadeira são executados.

Visão estéreo é utilizada em (MATSUMOTO; INO; OGASAWARA, 2001) para criar uma projeção da face, monitorando a direção do olhar do usuário, e assim, locomover a cadeira. Computação afetiva pderia ser outra abordagem possível para capturar aspectos das expressões facias do usuário, de maneira a interpretar os comandos. Em geral, estes métodos são complexos de serem implementados, alguns tendo sua implementação inviabilizada para sistemas de tempo real (BETTADAPURA, 2012; PANTIC; MEMBER; ROTHKRANTZ, 2000).

Em (FIA *et al.*, 2007), os autores apresentam uma interface baseada na gestura da cabeça para gerar comandos de locomoção. Seu método utiliza os mesmos algoritmos de detecção de faces *Adaboost*, e de rastreamento de objetos *Camshift* para extrair as intenções de movimentação do usuário. Como declarado pelos autores, o método ainda necessitaria ser extensivamente experimentado e avaliado, pois sua performance poderia ser afetada por imagens com fundos poluídos, situações de mudança de iluminação ou luz solar, e sombras.

Baseando-se nos métodos encontrados na literatura, pode-se afirmar que o desenvolvimento de tais interfaces de comando é uma tarefa desafiadora, pois a cadeira de rodas normalmente trafega por ambientes não controlados onde obstáculos estão presentes com grande frequência. Além disso, interfaces baseadas no reconhecimento visual do rosto ou cabeça do usuário devem considerar que as condições de iluminação podem mudar imprevisivelmente. Mais, diversas questões como a câmera (i.e., seu tipo, especificações de fábrica e posição) e características do usário (i.e., formato do rosto e variações no tom da pele) devem ser analisadas com cuidado.

1.2 Objetivos

O principal objetivo deste trabalho é desenvolver um protótipo de interface para geração de comandos em tempo real para um robô móvel com rodas, possibilitando futura implementação desta interface em uma cadeira de rodas motorizada, auxiliando pessoas poptadoras de paralisia nos músculos abaixo do pescoço. Tal interface deve realizar o rastreamento das sobrancelhas, e também aplicar o cálculo de Fluxo Ótico (FO) para a geração de comandos e locomoção do RMR, de maneira confiável.

É desejável que o desenvolvimento de uma interface como essa leve em conta que deve executar em tempo real, e deve ser o mais automatizada possível, ser robusta a trepidações, variações na iluminação e características inerentes ao usuário. Uma implementação simples e baixo custo de projeto também são apreciados.

1.2.1 Objetivos Específicos

Espera-se contribuir com uma nova interface robusta e de baixo custo, para que esta seja uma ferramenta a mais no desenvolvimento deste artigo de interesse social, que é a cadeira de rodas motorizada. Além da cadeira de rodas, pode-se imaginar outras aplicações, como comandar um *mouse*, ou enviar comandos para dispositivos eletrônicos comandados remotamente. Assim, os objetivos específicos são:

- Desenvolver um MVC que é capaz de realizar a interface entre o usuário e o RMR. O MVC deve ser capaz de rastrear o movimento das sobrancelhas imagens capturadas e traduzir isso em comandos para o RMR. Além disso, o MVC deve ser capaz de estimar o movimento da região da imagem que encontra-se fora da área do rosto do usuário (fundo). Essa estimativa de movimento será realizada com o cálculo de FO nesta região, com isto será possível gerar comandos para o direcionamento do RMR;
- Desenvolver um protótipo de capacete com uma câmera acoplada. A câmera deve estar voltada para o rosto do usuário, e deve capturar a sequência de imagens, as quais são transferidas para processamento em um sistema de tempo real;
- Montar e programar um RMR, para ser usado como um protótipo, que represente a cadeira de rodas. Este será o MR e terá a finalidade de avaliar como uma cadeira de rodas responderia aos comandos gerados pelo MVC em uma situação real. O MR deve ser capaz de receber os comandos gerados pelo MVC por meio de comunicação *Bluetooth*® . Uma vez recebido o sinal de um comando, o RMR deve executa-lo;
- Executar uma bateria de testes que quantifique o desempenho da interface desenvolvida.

1.3 Método Proposto

O sistema proposto processa um vídeo com o intuíto de monitorar as instruções dadas pelo movimento da cabeça e "objetos"(i.e., sobrancelha) do rosto de uma pessoa, para comandar um protótipo de cadeira de rodas. A câmera, que é propositalmente presa a um capacete (Fig. 1), captura uma sequência de imagens da visão frontal da cabeça do usuário, em tempo real (30 quadros por segundo), e envia este vídeo a um *laptop*, onde os módulos de visão computacional são executados. Os módulos de *software* foram implementados com o suporte das bibliotecas da OpenCV Application Programming Interface (API).

As instruções extraídas da sequência de imagens são enviadas a um RMR que realiza os movimentos indicados pelos comandos recebidos. Este RMR representa a cadeira de rodas, já que âmbos apresentam os mesmos graus de liberdade, dados por $(\delta_m, \delta_s) = (2, 0)$, onde δ_m define o grau de mobilidade correspondendo ao número de rodas convencionais fixas, e δ_s define o grau de dirigibilidade correspondendo ao número de rodas convencionais

Figura 1 – Câmera presa ao capacete, voltada para o rosto do usuário.

orientáveis centradas que podem ser orientadas independentemente para manobrar o RMR (CAMPION *et al.*, 2008; CAMPION; BASTIN; D'ANDRÉA-NOVEL, 2011).

A execução da interface de visão computacional é dividida em dois passos principais: na Inicialização, o método proposto entra na Fase de Calibração na qual, primeiramente, detecta as regiões das sobrancelhas pela aplicação do detector de objetos *Haar Cascades* (VIOLA; JONES, 2001). Para gerar os comandos de *pare* e *ande*, as imagens de recorte das sobrancelhas detectadas são convertidas para o espaço de cores CIELab, e as bandas L e b são então binarizadas independentemente através do método de limiarização clássica de Otsu. Uma composição de tais binarizações provê as segmentações finais das sobrancelhas. Por fim, a assinatura de bits vertical é computada de maneira a realizar o rastreamento do movimento das sobrancelhas nas imagens segmentadas. Para ligar ou desligar o reconhecimento de comandos, e fornecer a detecção de comandos de direcionamento, o movimento da cabeça do usuário é analisado levando em conta o campo de FO computado ao fundo dos quadros da sequência de imagens. No passo de Geração de Comandos, as rotinas de rastreamento das sobrancelhas e de monitoramento do FO são iteradas continuamente, para interpretar as intenções de movimento do usuário enviá-las ao RMR.

Testes considerando pessoas com diferentes tons de pele e iluminações ambiente variadas foram conduzidos para avaliar a acurácia do método de rastreamento das sobrancelhas. Também foram executados testes em que diversos usuários comandavam a locomoção do RMR em um circuito controlado em ambiente interno. Os resultados mostram que é possível gerar comandos a partir da interface desenvolvida, com precisão satisfatória. Os resultados obtidos com a implementação e testes deste trabalho foram aceitos em uma conferência internacional na área de ciência da computação, e também foram encorpados e submetidos à um periódico na área de visão computacional (OLIVEIRA; FLORES; ALMEIDA, 2015b; OLIVEIRA; FLORES; ALMEIDA, 2015a).

1.4 Organização do Texto

Este texto está organizado como descrito a seguir. A Seções 1.2 e 1.3 expuseram a proposta deste projeto, bem como os objetivos específicos. O Capítulo 2 apresenta os conceitos teóricos que foram necessários para se desenvolver o protótipo de interface de

comandos. O Capítulo 3 descreve o funcionamento do método de geração de comandos proposto. Os testes realizados e os resultados alcançados estão descritos no Capítulo 4. Por fim, algumas discussões e conclusões são apresentadas no Capítulo 5.

2 Revisão de Literatura

Para entender o funcionamento dos métodos que estão por trás da interface desenvolvida, é necessário compreender os conceitos que foram utilizados nos módulos do sistema. Primeiramente, apresentam-se as teorias que dizem respeito às técnicas de visão computacional, como o classificador *Haar Cascades*, modelo de cores Lab, limiarização pelo método de Otsu, as assinaturas de bits, o FO Lucas & Kanade com pirâmides, MHI e as bibliotecas do OpenCV. Além dos conceitos de *software*, também são apresentadas informações sobre as partes de *hardware* que compõem o RMR.

2.1 O Classificador *Haar Cascades*

Apresentado por (VIOLA; JONES, 2001), o classificador chamado *Haar Cascades* é um algoritmo de aprendizado que pode ser treinado para reconhecer dois tipos de objetos em uma imagem, a saber: “objeto” e “não-objeto”, também chamados exemplos positivos e negativos respectivamente.

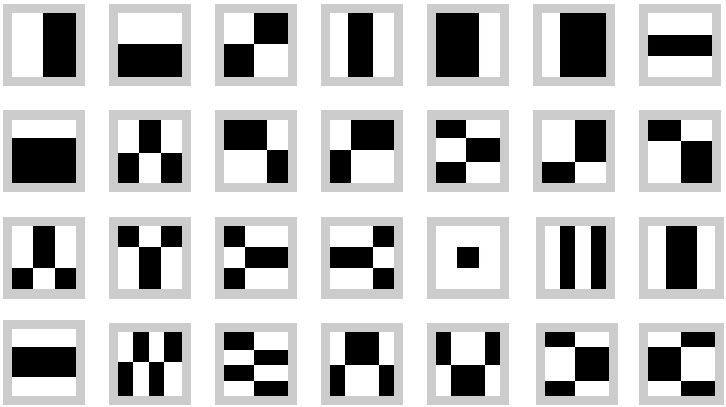


Figura 2 – Exemplos de *Haar-like features*.

Primeiramente, um amplo conjunto de dados é usado como base de amostras de imagens positivas e negativas. A idéia é usar um conjunto de *Haar-like features*, como as ilustradas na Figura 2, para quantificar a representação das classes de objetos; cada *feature* é usada como uma máscara convolucional bidimensional, como ilustra a Figura 3. Quando aplicada em uma dada imagem, a *feature* oferece um valor único, que é a subtração entre soma dos pixels que estão na região preta, e a soma dos que estão na região branca. Essa operação é chamada de "cálculo do coeficiente", e fornece características espaciais dessa imagem, como contornos e áreas homogêneas, podendo assim, atuar como detectores locais. As *features* também são chamadas de classificadores fracos, pois são úteis para identificar

partes do objeto analisado, porém uma feature por si só, não é capaz de classificar o objeto como um todo, sendo necessário um conjunto de *features* para que o classificador se torne robusto.

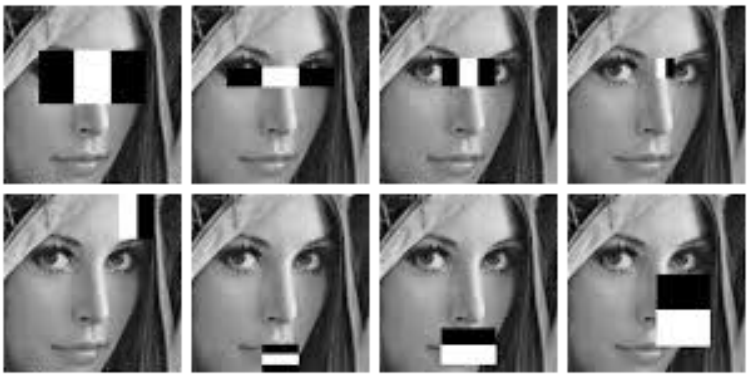


Figura 3 – Cálculo do coeficente: convolução de *Haar-features*.

Baseado na resolução dos detectores usados (*features* com 24×24 pixels), um conjunto exaustivo poderia conter mais de 180000 *features* (VIOLA; JONES, 2001). Entretando, notou-se experimentalmente que pode-se excluir um grande número dessas *features*, pois apenas um subconjunto muito menor será útil para representar a classe de objetos. Para isso, utiliza-se o algoritmo *Adaboost* (FREUND; SCHAPIRE, 1997).

A escolha do subconjunto de *features* é executada nos seguinte passos: Cada *Haar-feature* é considerada por si só, como sendo um classificador fraco. Para cada *feature* é encontrado um limiar que classifica o objeto em positivo ou negativo. São selecionadas então as *features* que apresentam a menor taxa de erro ou classificações errôneas. Ao final, ajustam-se os pesos de cada *feature*, e executa-se de novo os passos anteriores, até que uma precisão satisfatória seja alcançada. Este procedimento reduz o conjunto original de *Haar-like features*, fornecendo um subconjunto muito menor com cerca de 6000 *features* relacionadas com a classe objeto (neste caso, rostos humanos). A idéia é combinar o subconjunto de classificadores fracos resultante em um classificador forte efetivo.

Se para cada janela de 24×24 pixels dentro da imagem original, forem aplicados os 6000 operadores, o processamento ainda será computacionalmente ineficiente. Assim, *Adaboost* é usado novamente, mas agora, para dividir o subconjunto recém obtido em 38 subconjuntos que, combinados em forma de cascata, otimizam a tarefa de classificação dos objetos. A estrutura em cascata (Fig. 4) reflete o fato de que, dentro de uma única imagem, a maioria esmagadora das subimagens serão exemplos negativos (VIOLA; JONES, 2001) que podem ser descartados, assim um objeto é rejeitado nos níveis anteriores da cascata se esse for classificado como negativo. O objeto é reconhecido como positivo somente quando todos os estágios da cascata confirmarem este fato. Caso um único nível da cascata aponte resultado negativo, o objeto é descartado e procura-se uma nova região da imagem para ser processada.

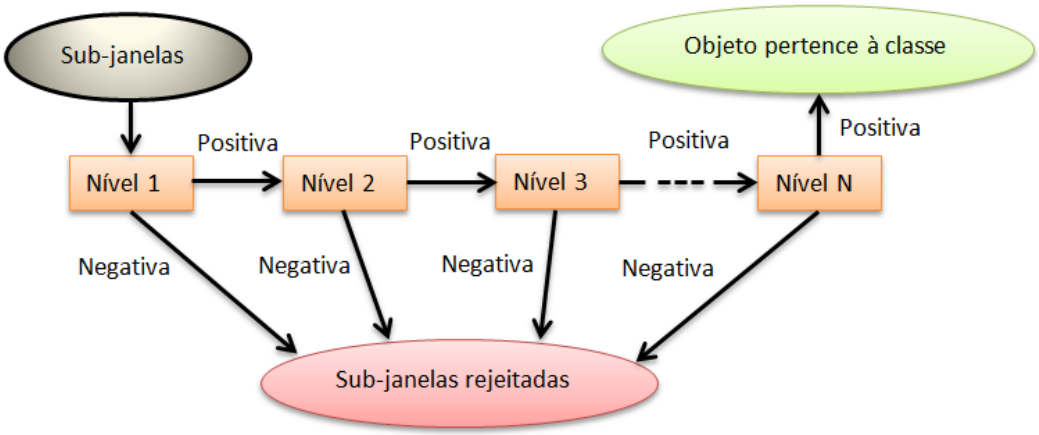


Figura 4 – Esquema do classificador em cascata.

2.2 O Espaço de Cores CIELab

Diversos espaços de cores expressam as cores por sua luminância e crominância de maneira separada. Esses podem ser categorizados como fazendo parte dos espaços de luminância-crominância (BUSIN; VANDENBROUCKE; MACAIRE, 2009). As bandas dos espaços luminância-crominância são usualmente derivados das componentes do espaço de cores RGB por transformações não lineares, como é o caso do CIELab.

Recomendado pela CIE (*Commission Internationale d'Éclairage*, 1986), o Lab é um espaço de cores que propõe um modelo de percepção uniforme, através de uma métrica para estabelecer a correspondência entre a diferença de cores percebidas por um observador humano e a distância medida nesse espaço de cores (BUSIN; VANDENBROUCKE; MACAIRE, 2009). Ele foi desenvolvido para representar as cores do mundo real e para ser independente de dispositivos de captura, ou de exibição.

Neste espaço de cores, a banda L representa os valores de luminosidade ou brilho (componente de luminância). As bandas a e b, quando possuirem valores negativos, mostrarão cores esverdeadas ou azuladas respectivamente, e na medida em que os valores dessas bandas aumentarem, as cores tenderão a serem mais avermelhadas ou amareladas respectivamente. Em ambos os casos, se as bandas a e b apresentarem valores intermediários, as cores tenderão a tons de cinza. O esquema da Figura 5 ilustra as explicações recém mencionadas.

2.3 Limiarização de Otsu

A principal ídeia de limiarização de Otsu é identificar um valor k que maximize a variância existente entre dois conjuntos, um formado pelos pixels que possuem valores de níveis de cinza maiores que este limiar k, e o outro formado pelos que possuem os valores menores que k (OTSU, 1979).

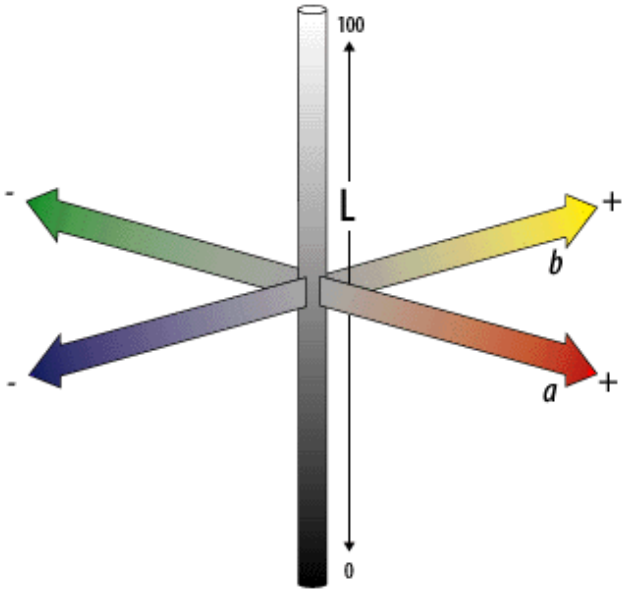


Figura 5 – Percepção visual do modelo de cores CIELab.

Sejam os pixels de uma determinada imagem representados em L níveis de cinza, variando no intervalo $[1, 2, ..., L]$. O número de pixels no nível i é denotado por n_i e o número total de pixels por $N = n_1 + n_2 + ... + n_L$. O histograma de níveis de cinza é normalizado e definido pela distribuição de probabilidade:

$$p_i = n_i/N, \qquad p_i \geq 0, \quad \sum_{i=1}^{L} p_i = 1 \tag{2.1}$$

Agora suponha que duas classes C_0 e C_1 (fundo e objetos de uma imagem, ou vice-versa) possam ser dicotomizados por um valor de limiarização no nível k; onde C_0 denota os pixels que possuem níveis de cinza dentro do intervalo $[1, .., k]$, e C_1 denotando os pixels com $[k + 1, ..., L]$ níveis de cinza. As probabilidades de ocorrência das classes e os níveis médios das classes são dados, respectivamente, por:

$$\omega_0 = Pr(C_0) = \sum_{i=1}^{k} p_i = \omega(k) \tag{2.2}$$

$$\omega_1 = Pr(C_1) = \sum_{i=k+1}^{L} p_i = 1 - \omega(k) \tag{2.3}$$

e

$$\mu_0 = \sum_{i=1}^{k} iPr(i|C_0) = \sum_{i=1}^{k} ip_i/\omega_0 = \mu(k)/\omega(k) \tag{2.4}$$

$$\mu_1 = \sum_{i=k+1}^{L} iPr(i|C_1) = \sum_{i=k+1}^{L} ip_i/\omega_1 = \frac{\mu_T - \mu(k)}{1 - \omega(k)} \tag{2.5}$$

onde

$$\omega(k) = \sum_{i=1}^{k} p_i \tag{2.6}$$

e

$$\mu(k) = \sum_{i=1}^{k} ip_i \tag{2.7}$$

são os momentos cumulativos de ordem zero e primeira ordem do histograma até o nível k, respectivamente, e

$$\mu_T = \mu(L) = \sum_{i=1}^{L} ip_i \tag{2.8}$$

é a média total dos níveis da imagem original. É possível verificar a seguinte relação para quaisquer valores de k:

$$\omega_0\mu_0 + \omega_1\mu_1 = \mu_T, \qquad \omega_0 + \omega_1 = 1. \tag{2.9}$$

As variâncias das classes são dadas por

$$\sigma_0^2 = \sum_{i=1}^{k}(i-\mu_0)^2 Pr(i|C_0) = \sum_{i=1}^{k}(i-\mu_0)^2 p_i/\omega_0 \tag{2.10}$$

e

$$\sigma_1^2 = \sum_{i=k+1}^{L}(i-\mu_1)^2 Pr(i|C_1) = \sum_{i=k+1}^{L}(i-\mu_1)^2 p_i/\omega_1. \tag{2.11}$$

As variâncias das classes são estatísticas de segunda ordem. Para poder ter uma noção do quão eficiente é o limiar, no nível k, a seguinte medida de separabilidade das classes é uma das utilizadas por (OTSU, 1979) como análise discriminante:

$$\eta = \sigma_B^2/\sigma_T^2, \tag{2.12}$$

onde

$$\sigma_B^2 = \omega_0\omega_1(\mu_1 - \mu_0)^2 \tag{2.13}$$

e

$$\sigma_T^2 = \sum_{i=1}^{L}(i-\mu_T)^2 p_i \tag{2.14}$$

são a variância entre-classes, e a variância total dos níveis de cinza, respectivamente. (OTSU, 1979) mostra que maximizar σ_B^2 (2.13), ou equivalentemente η (2.12), significa encontrar um valor de k que apresente a maior separabilidade possível entre as duas classes de níveis de cinza. Por isso, o valor máximo de η, conhecido como η^*, serve também como medida para avaliação da separabilidade das classes (facilidade de limiarização), ou a bimodalidade do histograma da imagem. Esse, é unicamente determinado dentro do intervalo

$$0 \leq \eta^* \leq 1.$$

O limite inferior do intervalo (zero) é alcançável, se e somente se, a imagem apresentar apenas um único nível de cinza, e o limite superior é atingível, por e somente por, imagens que contêm apenas pixels com dois valores de níveis de cinza (OTSU, 1979).

2.4 Assinaturas de bits

Pode-se analisar a forma de objetos em imagens de diversas maneiras, as assinaturas de bits são normalmente usadas para obter informações espaciais sobre objetos em imagens binárias. A assinatura de bits é, basicamente, um mapa no qual suas posições são valoradas de acordo com a quantidade de pixels brancos em cada linha (ou coluna) da imagem binária (GONZALEZ; WOODS, 2006).

O vetor de assinatura de bits vertical está relacionado ao número de pixels brancos presentes nas linhas da imagem, e carrega informação sobre as formas verticais do objeto, enquanto que a i-ésima posição do vetor de assinatura de bits horizontal guarda o número de pixels brancos da coluna i da imágem binária, provendo informação espacial sobre os aspectos horizontais do objeto (PERES *et al.*, 2006).

Seja I uma imagem binária e $b_i(I)$ o número de pixels brancos na i-ésima linha (coluna) de I. A assinatura de bits vertical (horizontal) de I é o vetor

$$b = (b_1(f), b_2(f), b_3(f), \cdots, b_n(f)), \tag{2.15}$$

onde n é o número de linhas (colunas) de I.

2.5 Fluxo Ótico

A cada quadro, pode-se comparar as intensidades dos pixels do quadro anterior com o quadro que está sendo processado em um instante de tempo, para extrair dados do movimento relativo entre esses pixels. Ao vetor que relaciona um pixel da imagem anterior, com um pixel da imagem atual dá-se o nome de FO (*Optical Flow* em inglês), (BEAUCHEMIN; BARRON, 1995). Tal vetor representa o deslocamento que o pixel sofreu de um quadro para o outro.

Seja (x, y) a coordenada de um pixel no plano da imagem de um quadro I qualquer e t o instante em que esse quadro foi adquirido. Considere que os pixels $I(x, y)$ e $I(x + \Delta x, y + \Delta y)$, são processados nos instantes instantes t e $t + \Delta t$ respectivamente. Supondo que as intensidades desses pixels apresentam valores muito próximos, de maneira que tais pixels representem um mesmo objeto que se deslocou do quadro processado no tempo t para o quadro no tempo $t + \Delta t$. Pode-se então tentar calcular o vetor deslocamento $(\Delta x, \Delta y)$ desse pixel, como indica a seguinte aproximação:

$$I(x, y, t) \approx I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{2.16}$$

Ao assumir a prerrogativa que o movimento de um pixel é suficientemente pequeno entre dois quadros adquiridos nos tempos t e Δt, pode-se utilizar uma Série de Taylor

para obter a seguinte aproximação, (BEAUCHEMIN; BARRON, 1995):

$$I(x + \Delta x, y + \Delta y, t + \Delta t) \approx I(x, y, t) + \frac{dI\Delta x}{dx} + \frac{dI\Delta y}{dy} + \frac{dI\Delta t}{dt} + \epsilon \tag{2.17}$$

Substituindo-se (2.16) em (2.17), e considerando o caso ideal, com o erro ϵ igual a zero, obtém-se a seguinte equação:

$$\frac{dI\Delta x}{dx} + \frac{dI\Delta y}{dy} + \frac{dI\Delta t}{dt} = 0 \tag{2.18}$$

Ao dividir-se ambos os termos da equação por Δt obtém-se o seguinte resultado:

$$\frac{dI\Delta x}{dx\Delta t} + \frac{dI\Delta y}{dy\Delta t} + \frac{dI\Delta t}{dt\Delta t} = 0 \tag{2.19}$$

Sabendo que $\frac{\Delta x}{\Delta t} = V_x$ e $\frac{\Delta y}{\Delta t} = V_y$, simplifica-se a equação (2.19) resultando em:

$$\frac{dIV_x}{dx} + \frac{dIV_y}{dy} + \frac{dI}{dt} = 0 \tag{2.20}$$

onde V_x, V_y são as componentes do deslocamento nos eixos x e y respectivamente, ou seja o fluxo ótico do pixel $I(x, y, t)$ e $\frac{dI}{dx}$, $\frac{dI}{dy}$, $\frac{dI}{dt}$ são as derivadas da imagem I no ponto (x, y, t). Reescrevendo tais derivadas como I_x, I_y e I_t respectivamente, tem-se que:

$$I_x = \frac{dI}{dx} \tag{2.21}$$

$$I_y = \frac{dI}{dy} \tag{2.22}$$

$$I_t = \frac{dI}{dt} \tag{2.23}$$

Substituindo as equações (2.21), (2.22) e (2.23) em (2.20):

$$I_xV_x + I_yV_y = -I_t \tag{2.24}$$

As derivadas I_x, I_y e I_t podem ser calculadas, ou seja, são termos conhecidos da equação (2.24), que é chamada de equação do fluxo ótico, (BEAUCHEMIN; BARRON, 1995). Nota-se que existem duas incógnitas e apenas uma equação, por isso, é necessário usar um artifício adicional para conseguir calcular as componentes vetoriais V_x e V_y do deslocamento que se deseja encontrar. Todos os métodos de fluxo ótico existentes, que seguem o modelo acima, introduzem restrições adicionais para estimar o vetor de fluxo ótico.

2.5.1 Método Lucas & Kanade

Para encontrar o vetor deslocamento $V = (V_x, V_y)$, utilizou-se neste trabalho o método de Lucas & Kanade (LUCAS; KANADE, 1981; LUCAS, 1984). Essa abordagem assume que o fluxo ótico é pequeno não só apenas em um pixel, mas sim, que o FO é o mesmo para toda uma vizinhança. Assim, usa-se a equação (2.24) para escrever o sistema de equações abaixo, para a vizinhança em questão. Neste caso, existirão mais equações do que incognitas, o que pode gerar uma superestimativa (erro), que será minimizado pelo método dos mínimos quadrados. Considerando que a vizinhança em questão é composta por n pixels, q_1,q_2 ... q_n, n equações serão geradas, as componentes do deslocamento V_x e V_y poderão ser obtidas resolvendo o sistema linear formado por tais equações:

$$\begin{cases} I_x(q_1)V_x + I_y(q_1)V_y = -I_t(q_1) \\ I_x(q_2)V_x + I_y(q_2)V_y = -I_t(q_2) \\ \qquad\qquad\qquad . \\ \qquad\qquad\qquad . \\ \qquad\qquad\qquad . \\ I_x(q_n)V_x + I_y(q_n)V_y = -I_t(q_n) \end{cases}$$

Novamente, $I_x(q_i)$, $I_y(q_i)$ e $I_t(q_i)$ são as derivadas parciais da imagem em relação às variáveis x, y e t, sobre o ponto q_i num determinado quadro (instante de tempo). As equações recém mencionadas acima podem ser escritas na forma matricial:

$$Av = b \tag{2.25}$$

onde: $A = \begin{bmatrix} I_x(q_1)\ I_y(q_1) \\ I_x(q_2)\ I_y(q_2) \\ \quad . \\ \quad . \\ \quad . \\ I_x(q_n)\ I_y(q_n) \end{bmatrix}$, $v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}$ e $b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \quad . \\ \quad . \\ \quad . \\ -I_t(q_n) \end{bmatrix}$.

Considerando-se uma vizinhança com n pixels, a matriz A tem dimensões $n \times 2$, v é um vetor de dimensão 2×1 e b é um vetor de $n \times 1$. Esta forma matricial nada mais é que o sistema linear mencionado anteriormente, o qual possui mais equações do que incógnitas.

Para resolver o sistema é preciso aplicar transformações matriciais que irão normalizá-lo. Multiplicando ambos os termos da equação (2.25) pela transposta de A (A^T) à esquerda, obtém-se:

$$A^T Av = A^T b \tag{2.26}$$

Multiplicando à esquerda, ambos os termos da equação (2.26) pela matriz pseudo-inversa de A, ou seja $(A^TA)^{-1}$, é possível isolar o vetor de incógnitas v, resultando em:

$$v = (A^TA)^{-1}A^Tb \tag{2.27}$$

conhecendo os valores da matriz A e dos vetores v e b, aplica-se a definição de multiplicação de matrizes, para expandir a equação matricial anterior em:

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i)I_y(q_i) \\ \sum_i I_y(q_i)I_x(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix} \begin{bmatrix} -\sum_i I_x(q_i)I_t(q_i) \\ -\sum_i I_y(q_i)I_t(q_i) \end{bmatrix} \tag{2.28}$$

para $i = 1..n$.

2.5.1.1 Método Lucas & Kanade com Pirâmides

Sejam I e J duas imagens 2D em níveis de cinza. Os dois valores $I(u) = I(u_x, u_x)$ e $J(v) = J(v_x, v_x)$, em níveis de cinza, representam pixels nas duas imagens, nos pontos $u = [u_x\ u_x]^T$ e $v = [v_x\ v_x]^T$ respectivamente, onde u_x, v_x, u_y e v_y são as coordenadas cartesianas dos pontos genéricos u e v. A imagem I pode ser também chamada de primeira imagem, e a imagem J de segunda. O vetor de coordenadas do pixel localizado no canto superior esquerdo é $[0\ 0]^T$, e o pixel localizado no canto inferior direito tem seu vetor de coordenadas valendo $[n_x - 1\ n_y - 1]^T$, fazendo com que n_x e n_y sejam respectivamente a largura e a altura das duas imagens, (BOUGUET, 2000).

Considere um ponto $u = [u_x\ u_y]^T$ na primeira imagem I. O objetivo principal é realizar o rastreamento deste pixel, localizando um ponto $v = u + d = [u_x + d_x\ u_y + d_y]^T$, na segunda imagem J de modo que $I(u)$ e $J(v)$ sejam "similares". O vetor $d = [d_x\ d_y]^T$ é o deslocamento do ponto genérico u na imagem, ou seja, o fluxo ótico em u. A noção de similaridade recém comentada é apresentada pela equação (2.29). Sejam m_x e m_y dois inteiros, o deslocamento d, no ponto u da imagem é definido como o vetor que minimiza a função ϵ definida a seguir:

$$\epsilon(d) = \epsilon(d_x, d_y) = \sum_{x=u_x-m_x}^{u_x+m_x} \sum_{y=u_y-m_y}^{u_y+m_y} (I(x,y) - J(x+d_x, y+d_y))^2 \tag{2.29}$$

Nota-se que os somatórios da função ϵ fazem com que entrem no cálculo uma vizinhança de pixels de tamanho $(2m_x + 1) \times (2m_y + 1)$. Esta vizinhança também é chamada de janela de integração. Os valores mais comuns para m_x e m_y são 2, 3, 4, 5, 6, 7 pixels, (BOUGUET, 2000).

Usa-se o esquema de pirâmides para conseguir estimar movimentos de valor maior, enquanto pode-se manter a janela de integração pequena. Esta afirmação ficará mais clara ao final desta seção.

Seja uma imagem I de tamanho $n_x \times n_y$. O nível zero da pirâmide é $I^0 = I$, essa imagem é essencialmente a de menor resolução (I é a imagem original). As dimensões da

imagem nesse nível são $n_x^0 = n_x$ e $n_y^0 = n_y$. A representação da pirâmide é construída de maneira recursiva: computa-se I^1 a partir de I^0, então computa-se I^2 a partir de I^1 e assim sucessivamente. Seja $L = 1, 2, ...$ um nível genérico da pirâmide, e seja I^{L-1} a imagem no nível $L-1$. Denota-se por n_x^{L-1} e n_y^{L-1} a largura e a altura de I^{L-1} respectivamente. Assim, (BOUGUET, 2000) define a imagem I^{L-1} pela seguinte equação:

$$\begin{aligned} I^L(x,y) = &\frac{1}{4}I^{L-1}(2x,2y)+ \\ &\frac{1}{8}[I^{L-1}(2x-1,2y) + I^{L-1}(2x+1,2y) + I^{L-1}(2x,2y-1)+ \\ &I^{L-1}(2x,2y+1)]+ \\ &\frac{1}{16}[I^{L-1}(2x-1,2y-1) + I^{L-1}(2x+1,2y+1) + I^{L-1}(2x+1,2y-1)+ \\ &I^{L-1}(2x-1,2y+1)] \end{aligned} \tag{2.30}$$

A equação (2.30) é definida somente para valores de x e y de maneira que $0 \leq 2x \leq n_x^{L-1} - 1$ e $0 \leq 2y \leq n_y^{L-1} - 1$. Portanto, a largura n_x^L e a altura n_y^L de I^L são os maiores inteiros que satisfazem as duas condições:

$$n_x^L \leq \frac{n_x^{L-1} + 1}{2} \tag{2.31}$$

$$n_y^L \leq \frac{n_y^{L-1} + 1}{2} \tag{2.32}$$

As equações (2.30), (2.31) e (2.32) são usadas para construir recursivamente as representações das duas imagens I e J: $\{I^L\}_{L=0,...,L_m}$ e $\{J^L\}_{L=0,...,L_m}$, onde L_m é a altura da pirâmide. Segundo o autor do método, (BOUGUET, 2000), a equação (2.30) sugere que o filtro $[\frac{1}{4}\ \frac{1}{2}\ \frac{1}{4}] \times [\frac{1}{4}\ \frac{1}{2}\ \frac{1}{4}]^T$ passa-baixa seja utilizado para obter anti-serrilhamento e suavização antes de fazer a amostragem da imagem para o nível superior da pirâmide.

Uma vez que já se sabe como representar cada imagem em seu respectivo nível da pirâmide, pode-se retomar o objetivo de rastrear o vetor deslocamento d. Assim, para cada nível L, define-se $u^L = [u_x^L\ u_y^L]$ como sendo as coordenadas que correspondem ao ponto u nas imagens piramidais I^L. Seguindo a definição de representação de pirâmide, equações (2.30), (2.31) e (2.32), os vetores u^L são calculados da seguinte maneira:

$$u^L = \frac{u}{2^L} \tag{2.33}$$

A operação de divisão na equação (2.33) é aplicada em cada coordenada independentemente, sendo importante lembrar que $u^0 = u$.

O algoritmo de rastreamento do deslocamento utilizando pirâmides funciona basicamente assim: primeiramente, o fluxo ótico é calculado no nível L_m mais alto da pirâmide.

Então, o resultado deste cálculo é propagado para o nível $L_m - 1$ inferior, na forma de uma estimativa inicial sobre o deslocamento do pixel. Dada essa estimativa inicial, o fluxo ótico é calculado no nível $L_m - 1$, e o resultado é propagado para o nível $L_m - 2$, e assim por diante, até que o nível 0 (imagem original) seja alcançado. A Figura 6 ilustra esquematicamente como essa propagação ocorre de um nível para outro.

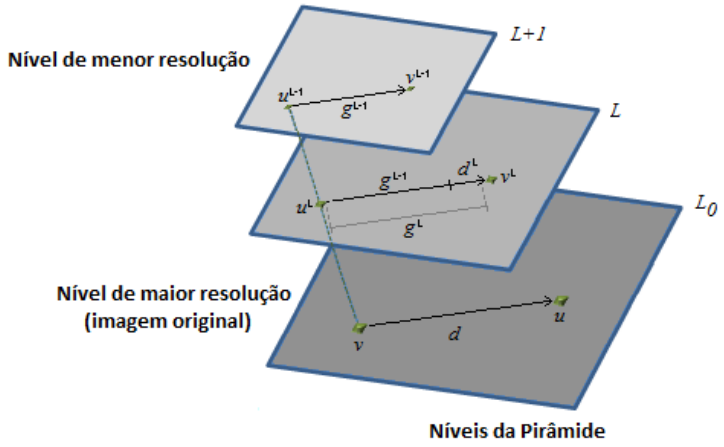


Figura 6 – Esquema visual da Método Lucas & Kanade com Pirâmides.

No nível L, a estimativa do fluxo ótico $g^L = [g_x^L \; g_y^L]$ é obtida pela propagação do nível L_m até $L + 1$. Assim, para computar o fluxo ótico no nível L é necessário encontrar o deslocamento residual $d^L = [d_x^L \; d_y^L]$ do pixel que minimiza a nova função ϵ^L:

$$\epsilon^L(d^L) = \epsilon^L(d_x^L, d_y^L) = \sum_{x=u_x-m_x}^{u_x+m_x} \sum_{y=u_y-m_y}^{u_y+m_y} (I(x,y) - J(x + g_x^L + d_x^L, y + g_y^L + d_y^L))^2 \quad (2.34)$$

Observe que o tamanho da janela de integração se mantém constante para todos os níveis L. Note também, que a estimativa g^L é usada para no cálculo da velocidade na segunda imagem J. Assim, resta apenas o vetor deslocamento residual $d^L = [d_x^L \; d_y^L]$ que é pequeno, facilitando o cálculo do fluxo ótico via o método Lucas & Kanade aplicado iterativamente.

Utilizando esquema de pirâmides é possível constatar que há um ganho na quantidade de estimativa de movimento, pois para os maiores valores de L a janela de integração engloba uma maior porção da imagem.

2.6 *Motion History Images*

Além do FO, o MHI, literalmente traduzido para o português como Imagem do Histórico de Movimento, é uma alternativa para analisar movimento em sequências de imagens. Proposto por (BRADSKI; DAVIS, 2002), esse método é útil na análise de

movimento em vídeos, podendo fazê-lo em regiões muito pequenas dos quadros. O princípio geral dessa técnica é obter a diferência entre quadros consecutivos (chamada silhueta), e a cada diferença encontrada, atribuindo um valor de tempo (*timestamp*) a essa diferença. A cada nova silhueta encontrada, o valor do tempo corrente é atribuido a essa, fazendo com que as silhuetas anteriores tenham um valor de tempo mais antigo. Ao longo de uma sequência de imagens em níveis de cinza, caso haja movimento, as silhuetas são salvas em uma imagem que leva o nome do método (MHI). Nessa imagem, silhuetas com valores de tempo mais novos são rotuladas com tons mais claros, enquanto que as mais antigas levam tons mais escuros. Em regiões onde não houve movimento, ou em que os *timestamps* ultrapassaram um limite paramétrico de tempo, a cor preta é utilizada para sinalizar que, nestes casos, o movimento é muito antigo, e já não é mais considerado.

A extração das silhuetas é a base do método, e de acordo com (BRADSKI; DAVIS, 2002), pode ser realizada de diversas maneiras. Algumas abordagens possíveis seriam utilizar disparidade estéreo ou subtração de profundidade estéreo, iluminação infravermelha de fundo, diferenciação de quadros, projeção com histogramas de cores de fundo, segmentação de *blobs* de textura etc. Pela simplicidade, tanto os autores, quanto a implementação do OpenCV utilizam a diferença absoluta entre quadros. Após realizada a diferenciação entre os quadros, realiza-se uma limizarização do resultado, para que seja gerada uma imagem binária da silhueta.

Utiliza-se uma imagem de pontos flutuantes para representar a Imagem de Histórico de Movimento, onde novos valores de silhuetas são copiados com um valor de tempo em ponto flutuante no formato: segundos.milisegundos. Esta representação da MHI é atualizada da seguinte maneira:

$$MHI_\delta(x,y) = \begin{cases} \tau & \text{se é a silhueta corrente em (x, y)} \\ 0 & \text{se MHI}_\delta\text{(x, y)} < (\tau - \delta) \\ MHI_\delta(x,y) & \text{caso contrário} \end{cases} \tag{2.35}$$

onde τ é o *timestamp* corrente, e δ é a constante de tempo máximo de duração da silhueta, passada por parâmetro. Esse método faz tal representação ser independente da velocidade do sistema, ou taxa de quadros, de maneira que os movimentos podem cobrir a mesma área da MHI a diferentes taxas de captura. A Figura 7 mostra como seria a MHI visualmente, tons de cinza mais claros representam silhuetas de movimento mais recentes, enquanto que tons mais escuros mostram silhuetas mais antigas.

De posse da MHI pode-se aplicar operadores convolucionais, como o Filtro de Sobel, para calcular os gradientes nas direções x e y, encontrando os vetores que apontam para a direção do movimento de cada pixel. A convolução dos operadores de Sobel em x e y gera as derivadas parciais $F_x(x,y)$ e $F_y(x,y)$. Assim, a orientação de cada pixel pode ser

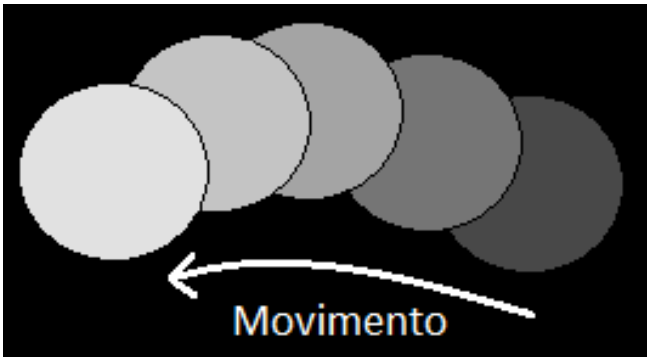


Figura 7 – Representação visual da Imagem de Histórico de Movimento.

obtida da seguinte maneira:

$$\phi(x,y) = arctan\frac{F_x(x,y)}{F_y(x,y)}. \tag{2.36}$$

Os autores de (BRADSKI; DAVIS, 2002) salientam que alguns gradientes podem apresentar valores muito altos, quando as silhuetas mais antigas são setadas para zero, ou quando os operadores encontram as bordas da imagem (fora das bordas da imagem, assume-se que os valores dos pixels são zero). Para contornar tal questão, sabendo-se a duração em que as silhuetas ficam gravadas na MHI, é possível detectar quão grandes são os gradientes e eliminá-los com base em limites superior e inferior, setados via parâmetro. Isso faz com que uma máscara de gradientes válidos seja gerada, permitindo a geração de vetores de movimento similares aos do FO.

Para se encontrar um único vetor, que indica qual foi a direção geral do movimento, considerando todos os pixels em que foi possível calcular a orientação (dentro da máscara de gradientes válidos), (BRADSKI; DAVIS, 2002) utiliza a seguinte fórmula de orientação global:

$$\overline{\phi} = \phi_{ref} + \frac{\sum_{x,y} angDiff(\phi(x,y),\phi_{ref}) \times norm(\tau,\delta,MHI_\delta(x,y))}{\sum_{x,y} norm(\tau,\delta,MHI_\delta(x,y))} \tag{2.37}$$

onde $\overline{\phi}$ é a orientação global de movimento, ϕ_{ref} é o ângulo base de referência (valor máximo do histograma de orientações), $\phi(x,y)$ é o mapa de orientação de movimento obtido pelas convoluções dos gradientes, $norm(\tau,\delta,MHI_\delta(x,y))$ são os valores normalizados da MHI (normalização linear da MHI utilizando o *timestamp* corrente τ e a duração δ), e $angDiff(\phi(x,y),\phi_{ref})$ é diferença angular mínima entre a orientação do pixel e o ângulo de referência.

2.7 A Biblioteca OpenCV

De acordo com a *homepage* desta biblioteca (INTEL, Acessado em 15/07/2015), a OpenCV é liberada sob uma licença de código aberto e por isso pode ser utilizada para

fins acadêmicos e comercial, gratuitamente. Esta biblioteca é útil no desenvolvimento de sistemas que envolvam processamento de imagens e visão computacional. Suas aplicacões variam desde arte interativa, inspeção de minas, mapas de corte na internet, sensoriamento remoto, sistemas médicos e até robótica avançada. Nela existem interfaces em C++, C, Python, e Java, além de suportar os sistemas operacionais Windows, Linux, Max OS, iOS, e Android. A OpenCV foi desenvolvida para ser computacionalmente eficiente e tem forte foco em aplicações de tempo real. Escrita em C/C++ otimizado, a biblioteca pode tirar proveito de processamento paralelo, acompanhando a tendência de utilizar recursos computacionas das placas de vídeo, e processadores Single Instruction Multiple Data (SIMD).

Neste contexto, essa biblioteca se torna útil na captura e processamento das imagens provenientes da câmera que registra o rosto do usuário da cadeira de rodas. Por ter código otimizado e voltado para processamento em tempo real, é indispensável para este trabalho.

2.8 Robôs

Sabe-se que a robótica é a ciência que estuda sistemas compostos por partes mecânicas automáticas e controladas por circuitos integrados, e esses sistemas são comumente chamados de robôs. As aplicações da robótica são as mais variadas, podendo-se utilizar tais sistemas para realizar tarefas repetitivas, tarefas em que há risco de vida, manipulação de materiais pesados ou perigosos, aplicações militares, etc (SCIAVICCO; VILLANI, 2009; ORTIGOZA *et al.*, 2012).

Dentro deste contexto, pode-se citar duas grandes categorias destes sistemas, os robôs manipuladores e os robôs móveis. Os robôs manipuladores são constituídos de partes rígidas articuladas, que não se deslocam como um todo, mas apenas realizam movimentos locais, com o robô fixado a algum substrato. Já a classe dos robôs móveis engloba os sistemas que são constituídos de um corpo rígido equipado com um sistema de locomoção, que possibilita a movimentação do robô, como um todo, em um ambiente terrestre, aquático ou aéreo (SCIAVICCO; VILLANI, 2009; ORTIGOZA *et al.*, 2012). O foco do presente trabalho está nos robôs móveis terrestres. A forma de locomoção dos mesmos pode ser feita por meio de pernas mecânicas, esteiras rastejantes ou por rodas.

Quando se fala em uma cadeira de rodas elétrica, automaticamente remete-se aos RMRs, assim o foco deste trabalho será neste tipo de sistema robótico, por esse apresentar as mesmas características de uma cadeira de rodas.

2.8.1 Turtle-2WD Mobile Platform

O robô *Turtle-2WD*, fabricado pela *DFROBOT*® é uma plataforma que consiste de um corpo rígido (base) que possui duas rodas convencionais fixas, acionadas por atuadores

independentes (motores de corrente contínua) com o intuito de realizar a movimentação e orientação, e uma terceira roda que gira livremente (roda passiva), cuja função é apenas servir de suporte ao RMR, sendo que seus efeitos são desprezíveis na dinâmica do RMR, (DFROBOT, 2013). Uma ilustração dessa plataforma pode ser vizualizada na Figura 8. O RMR é conhecido como unicíclo ou do tipo $(\delta_m, \delta_s) = (2, 0)$, onde δ_m define o grau de mobilidade correspondendo ao número de rodas convencionais fixas, e δ_s define o grau de dirigibilidade correspondendo ao número de rodas convencionais orientáveis centradas que podem ser orientadas independentemente para manobrar o RMR (CAMPION *et al.*, 2008; CAMPION; BASTIN; D'ANDRÉA-NOVEL, 2011).

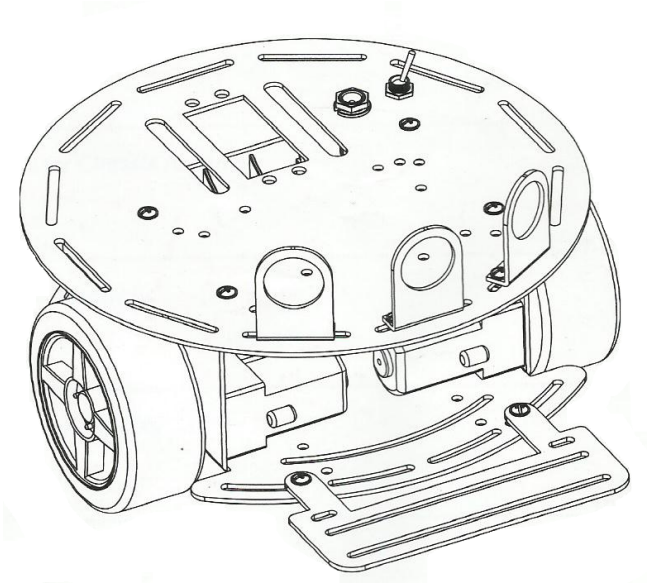


Figura 8 – *Turtle 2WD - Mobile Platform*, FONTE: (DFROBOT, 2013).

Este robô é ideal para representar uma cadeira de rodas, pois esta também possui o mesmo grau de manobrabilidade $\delta = (2, 0)$.

2.9 Arduino

Em termos práticos um Arduino é um pequeno computador que pode ser programado para processar entradas e saídas entre ele e os componentes externos conectados a ele (MCROBERTS, 2012). Sua placa é equipada com um microcontrolador Atmel AVR, um cristal oscilador, um regulador de tensão de 5 volts. O Arduino utilizado neste trabalho também é equipado com um conector Universal Serial Bus (USB), útil para realizar a alimentação do circuito, e para transferir um programa (código em C) para o microcontrolador. É comumente utilizado em sistemas embarcados que devem interagir com seu ambiente por meio de *hardware* e *software*. As possibilidades de desenvolvimento de aplicações que podem utilizar esse dispositivo são inúmeras, como por exemplo na área de impressões 3D, robótica, engenharia, automação residencial, e até em artes (MCROBERTS, 2012).

Neste trabalho foi utilizado o modelo Atmega Arduino Uno 328p, que é equipado com o microcontrolador Atmel ATmega328. Ele possui 14 pinos digitais de entrada/saída (dos quais 6 deles podem ser usados como saídas analógicas PWM), 6 pinos de entrada analógica, um ressonador cerâmico de 16 MHz, o conector USB, uma entrada de energia (*jack*) de 5V, um cabeçalho In-circuit Serial Programming (ICSP) para armazenar o programa no microcontrolador, e um botão de reset. Existem também circuitos complementares padronizados com funções específicas que podem ser acoplados às entradas e saídas do Arduino. Tais circuitos padronizados são denominados *shields*. Pode-se citar como exemplos de shields os sensores de ultrassom, sensores infravermelhos, também o transmissor/receptor *Bluetooth*® , chips controladores auxiliares, pontes H, entre outros (ARDUINO *et al.*, Acessado em 15/07/2015).

2.9.1 Sensores

Um sensor elétrico é um transdutor que converte algumas formas de energia em sinais elétricos, com o intuito de sentir o ambiente, ou seja, detectar algumas características do local onde se encontra este dispositivo. Diversos tipos de sensores apresentam diferentes funções, por exemplo, medir pressão atmosférica, umidade, temperatura, entre outros. Neste trabalho, maior importância é dada ao sensor de distância, também chamados telêmetros (MCROBERTS, 2012), ou sensores de colisão, podendo-se citar dois tipos: os ultrassônicos, e os infravermelhos.

O sensor de distância ultrassônico tem o objetivo de emitir pulsos de som de frequência muito elevada (acima do limite de audição humano), para que esses reflitam em possíveis objetos que estejam no direção desse transdutor. Baseando-se na velocidade do som, e sabendo o tempo decorrido desde a emissão até a captação do ultrassom refletido pelo obstáculo, pode-se estimar a distância percorrida pelas ondas sonoras. Assim é possível determinar se existe algum objeto à frente, e também o valor dessa distância (MCROBERTS, 2012).

Os telêmetros baseados em infravermelho funcionam pelo mesmo princípio dos sensores anteriores, emitindo pulsos, captando suas reflexões em possíveis obstáculos. A diferença aqui é que se emitem feixes ou um campo de luz infravermelha (frequências menores do que a visão humana pode detectar) para detectar presença e distância de objetos à frente do sensor. Diferentes tipos de sensores infravermelho podem ser melhor aplicados dependendo do material que compõe cada obstáculo (MCROBERTS, 2012).

3 Método Proposta

Duas técnicas básicas têm sido usadas para compor cadeiras de rodas inteligentes: 1) técnicas de auto navegação e desvio de obstáculos e; 2) interfaces convenientes que permitam os usuários com deficiência controlar a cadeira de rodas por eles mesmos usando as habilidades físicas que possuem (JU; SHIN; KIM, 2009). Muitos pesquisadores se concentram apenas em sensoriar o ambiente e planejar caminhos para que a cadeira de rodas desvie de obstáculos de maneira autônoma. É importante desenvolver um sistema que permita que os usuários interajam com a navegação, o sistema torna-se pouco robusto se ele não puder ser adaptado às habilidades do usuário. Isto é, o usuário da cadeira possui limitações motoras, então a cadeira deve considerar as intenções do usuário enquanto garante segurança e confiabilidade.

O objetivo deste capitulo é apresentar uma interface composta por dois módulos (a saber MVC e MR), que possa comandar uma cadeira de rodas motorizada, enquadrando este trabalho na categoria (2) supracitada. O MVC é constituído por uma interface capaz de reconhecer os movimentos da cabeça e sobrancelhas do usuário, transmitindo-os em forma de comandos ao MR. A idéia é que o usuário utilize um capacete em que é presa uma câmera, como pode ser visto na Fig. 1. Essa câmera é responsável por capturar os quadros e enviá-los a um *notebook* onde é executado o MVC. O *laptop* analisa os quadros recebidos e traduz esses em comandos, e envia-os ao RMR (que representa a cadeira de rodas) por meio de comunicação *Bluetooth*® , para que esse, por sua vez, execute os movimentos conforme a natureza de cada instrução recebida, conforme ilustra a Figura 9. Neste trabalho, a *webcam* utilizada é uma Microsoft Lifecam HD-300 com ajuste automático de brilho. Já o *notebook* tem as seguintes especificações técnicas: marca HP, com processador Intel I5, 6 GB de memória RAM e sistema operacional Windows 7.

3.1 Módulo de Visão Computacional

O MVC é executado em dois passos: o primeiro chamado Inicialização, contém a fase de calibração, e a inicialização de algumas informações necessárias ao sistema. No passo de Geração de Comandos a identificação dos movimentos da cabeça do usuário é feita por meio do Módulo de FO, e o Módulo das Sobrancelhas raliza o rastreamento das sobrancelhas.

A Figura 10 ilustra o funcionamento geral do MVC. Primeiramente, na Fase de Calibração, o sistema procura pelos olhos, e depois, encontra a posição fixa das sobrancelhas relaxadas. Este passo é usado pelo Módulo das Sobrancelhas e será melhor explicado posteriormente.

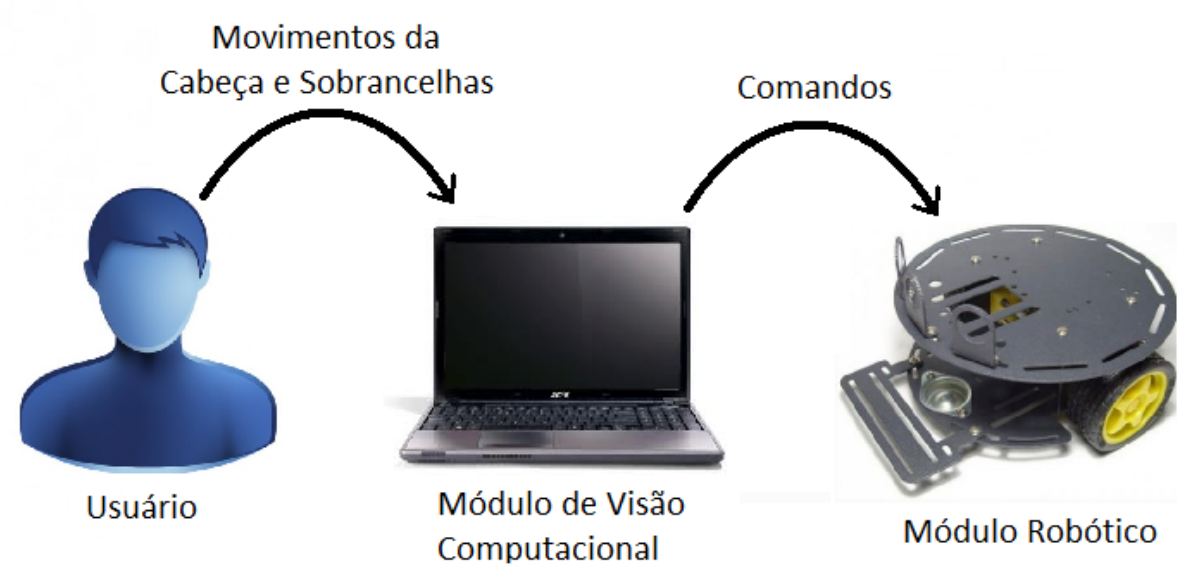


Figura 9 – Descrição esquemática do sistema

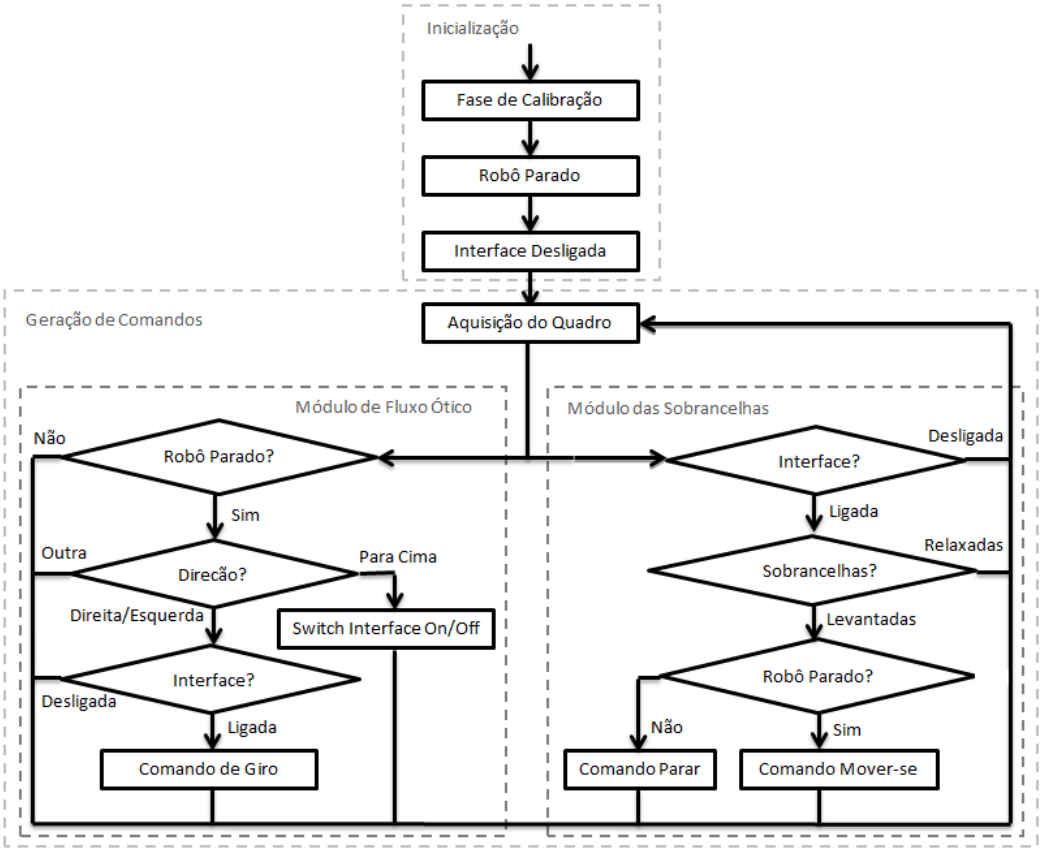


Figura 10 – Funcionamento geral do MVC.

No próximo passo, de Geração de Comandos, o sistema aguarda até que o usuário ligue o MVC, permitindo o reconhecimento dos comandos. O Módulo de FO é detecta os comandos de *ligar* e *desligar*, os quais são gerados somente quando o usuário realiza um movimento para cima com a cabeça. O chaveamento de acionamento do MVC é útil para

controlar os momentos em que não se deseja que o sistema reconheça outros comandos, liberando o usuário para conversar ou gesticular livremente.

Uma vez que o MVC está ligado, três tipo de comandos podem ser reconhecidos, a saber os comandos de chaveamento de acionamento (*ligar* e *desligar*), o chaveamento de movimento (*mover-se* e *parar*), e também os comandos de giro (*girar à esquerda* e *girar à direita*). Pela análise do campo de FO, se o sistema detecta que o usuário move sua cabeça para cima, e sabendo que o MVC está ativado (ligado), é reconhecido o comando para *desligar*, e nenhum outro comando será reconhecido a não ser *ligar* o MVC. Ainda baseado no FO, o sistema pode detectar se o usuário moveu sua cabeça em direção à esquerda ou à direita, gerando os comandos para girar *à esquerda* e *à direita* respectivamente.

O Módulo das Sobrancelhas detecta se as sobrancelhas estão levantadas, em caso positivo, o sistema chaveia seu estado de movimento, gerando os comandos para *mover-se* ou *parar*. Se o comando gerado é o de *mover-se*, o sistema entra no estado em que o MR está em movimento (robô não está parado). Enquanto neste estado, apenas outro comando para chaveamento de estado movimento pode ser reconhecido, gerando o comando *parar*, e retornando o sistema ao estado em que o MR está parado (robô parado). O sistema também pode gerar um comando para *parar* caso o ultrassom, instalado no robô, detecte um obstáculo à sua frente. A decisão de só permitir o reconhecimento de um *parar* quando o sistema está em movimento foi tomada para previnir que o Módulo de FO reconheça comandos indesejados. A Figura 11 mostra um diagrama que ilustra como os estados se alteram conforme os comandos são gerados.

Inicialmente o sistema encontra-se em estado Desligado. Ao executar um comando de ligar, dois estados se tornam ativos: sistema Ligado, e Robô Parado. Ao executar comandos de giro, os estados não se alteram. Quando é detectada a intenção de movimentar o RMR, o estado de MVC ligado permanece o mesmo, mudando apenas o estado de Parado para Movendo-se. Uma vez com o robô se movimentando, apenas o comando de parar pode ser reconhecido mudando o estado de Robô Movendo-se para o de Robô Parado, com o estado de MVC Ligado. Caso os estados estejam em MVC Ligado e Robô Parado, e o comando de desligar seja acionado, então o estado de MVC Desligado é ativado.

3.1.1 Detecção das Sobrancelhas

O posicionamento da câmera permite que o sistema suponha que a região da face do usuário está sempre centralizada na sequência de quadros, localizada logo abaixo do capacete. Isso permite que o sistema localize as *features* faciais (e.g. sobrancelhas) mais facilmente, livrando o sistema de processar todos os quadros à procura do rosto do usuário.

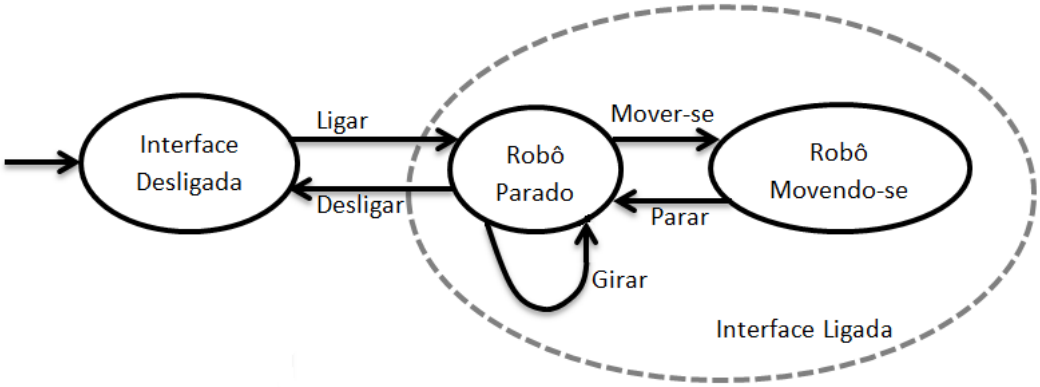


Figura 11 – Diagrama dos estados possíveis que o sistema pode assumir.

3.1.1.1 Fase de Calibração

Inicialmente o classificador *Haar Cascades* é aplicado na região logo abaixo do capacete (região da face), produzindo dois retângulos, cada um contendo um olho detectado. Estes retângulos são proporcionalmente deslocados acima para que sejam alcançadas as regiões das sobrancelhas, como pode ser visto na Fig. 12. Os recortes das sobrancelhas são feitos esperando-se obter janelas contendo apenas sobrancelha e a uma porção de pele que a permeia (Figuras 14.(a), 15.(a) e 15.(e)). Deste modo, espera-se obter histogramas bimodais (Fig. 14.(d)) nas regiões das sobrancelhas, para beneficiar a binarização pelo método de Otsu, uma vez que os recortes apresentarão basicamente dois tipos de objeto: pêlos e pele. A Fase de Calibração determina o recorte das regiões das sobrancelhas, utilizando esses retângulos para recortar as sobrancelhas no mesmo lugar em todos os frames, beneficiando-se da posição fixa da câmera no capacete.

Figura 12 – Fase de Calibração. O classificador *Haar Cascades* procura pelos olhos dentro da região delimitada pelo retângulo preto. Os retângulos verdes são as regiões recortadas das sobrancelhas.

Caso seja identificado que a detecção automática das regiões das sobrancelhas não

esteja adequada, pode-se optar por ajustá-las manualmente, com o auxílio de uma segunda pessoa. Após a execução do detector, existe a opção da realização de uma calibração manual que permite editar o tamanho e a localização dos retângulos que delimitam as sobrancelhas.

O esquema da Figura 13 mostra o como a Fase de Calibração pode ser executada. Inicialmente o sistema tenta procurar pelos dois retângulos das sobrancelhas, quando encontra exatamente duas regiões, essas são apresentadas ao usuário, como na Fig. 12, e então o sistema aguarda por um comando inserido pelo teclado:

- Se a tecla *Esc* for pressionada, o sistema encerra a determinação das regiões das sobrancelhas;
- Caso o usuário aperte a tecla *Espaço*, o sistema executa mais uma vez a detecção automática das sobrancelhas utilizando o classificador de *Haar* já mencionado. Isso faz com que novas regiões das sobrancelhas sejam delimitadas, discartando as anteriores;
- Ao pressionar *Backspace*, abre-se um novo leque de opções para editar os retângulos das sobrancelhas. Inicialmente o sistema dará a opção de o usuário editar a posição do retângulo direito. Para selecionar entre retângulo direito ou esquerdo, pressiona-se R ou L respectivamente. Para alternar entre o modo de edição de posição ou alteração do tamanho do retângulo, pressiona-se a tecla Q. Para alterar efetivamente as configurações dos retângulos, de acordo com o modo de edição, utiliza-se as teclas W, A, S e D como se fossem as setas do teclado, dando a direção da alteração. Pressionando *Backspace* novamente, o sistema retorna ao estado em que aguarda por uma opção de calibração.

3.1.2 Segmentação e Rastreamento das Sobrancelhas

Diversos espaços de cores foram experimentados, como HSV, RGB, YCbCr, entre outros. Porém o espaço de cores CIELab apresentou melhores indícios de bons resultados, portanto, as janelas contendo as sobrancelhas, são convertidas para esse espaço de cores. Baseando-se na expectativa de que o histograma das imagens contidas nas região de recorte das sobrancelhas tenda a ser bimodal, o método de limiarização de Otsu clássico (OTSU, 1979) é aplicado, com o intuito de separar os pixels do objeto (sobrancelha) e fundo (pele). A binarização da região das sobrancelhas é feita nas nas bandas L e b separadamente (Figuras 15.(b), 15.(c), 15.(f) e 15.(g)); a segmentação da sobrancelha é dada pela intersecção das duas bandas binarizadas (Figuras 15.(d) e 15.(h)).

Assume-se que nas ocasiões em que a luz atrapalha a segmentação, a banda b binarizada contém uma boa segmentação das sobrancelhas (Fig. 16).(a)-(f)). Nos casos

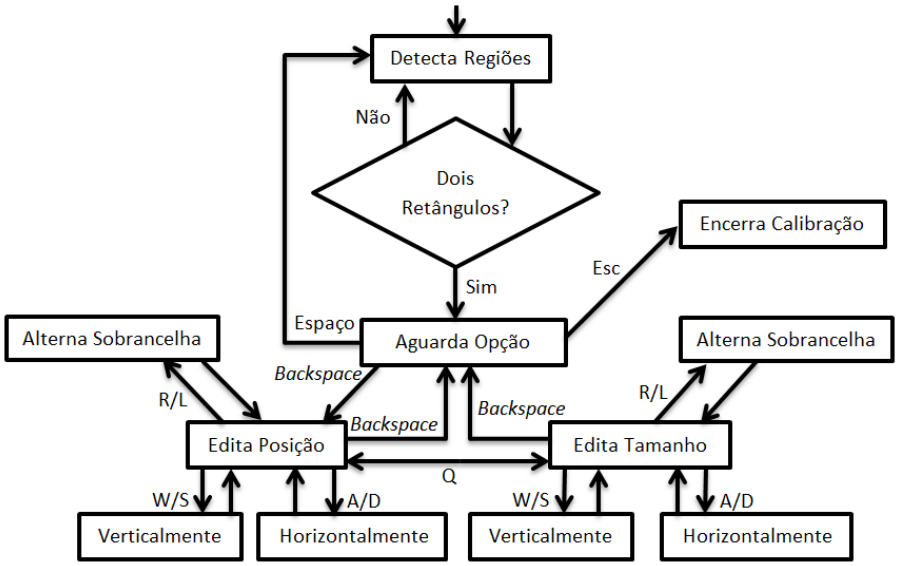
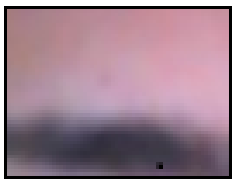
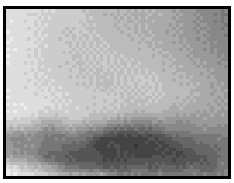


Figura 13 – Esquema de excução da determinação das regiões das sobrancelhas.

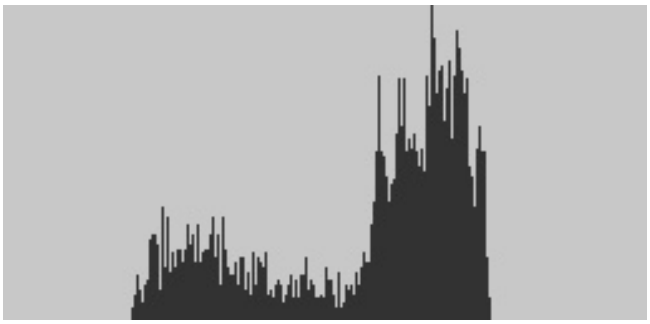


(a) Recorte



(b) Banda L

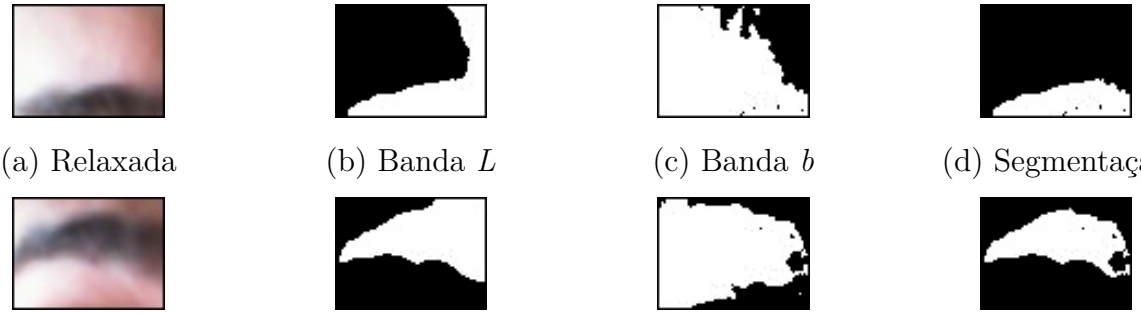
(c) L binarizada



(d) Histograma da banda L

Figura 14 – (a) A região recortada da sobrancelha, (b) Banda L, (c) Banda L limiarizada pelo método de Otsu e (d) Histograma da banda L.

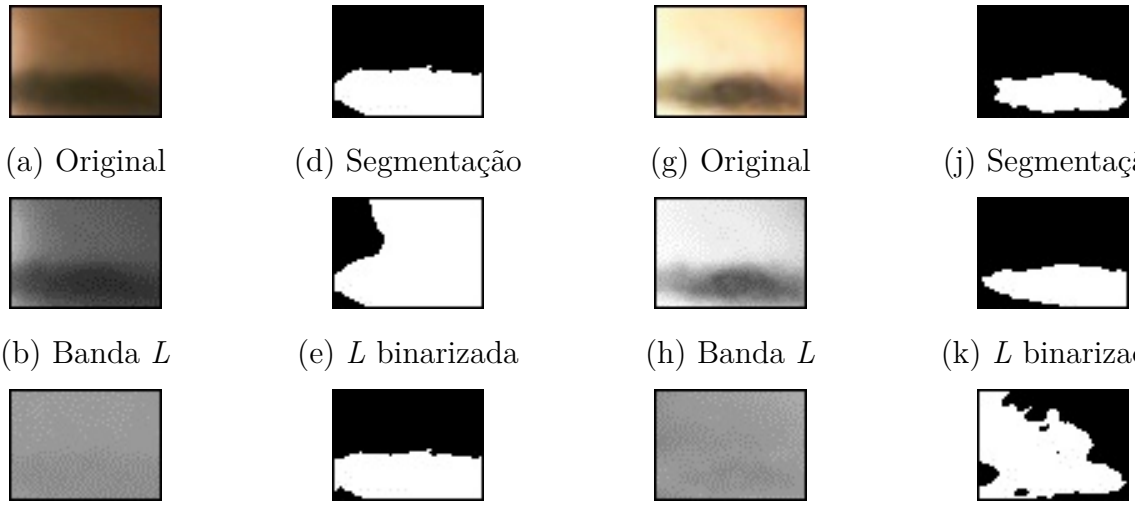
em que as condições de iluminação são boas, a banda L entra em cena e decide a tarefa de segmentar a sobrancelha (Fig. 16).(g)-(l)). Decidiu-se excluir a banda a deste método, pois na maior parte dos casos testados, esta banda apresentou um valor de $\eta*$ indicando que a limiarização desta banda apresenta baixa separabilidade entre pele e sobrancelha (o índice $\eta*$ foi melhor explicado na sessão 2.3). A Figura 17 mostra um exemplo no qual a banda a apresenta o pior índice de separabilidade $\eta*$ (Fig. 17.(d)), e a limiarização pelo método de Otsu confirma o resultado de segmentação (Fig. 17.(c)).



(a) Relaxada (b) Banda L (c) Banda b (d) Segmentação

(e) Levantada (f) Banda L (g) Banda b (h) Segmentação

Figura 15 – Segmentação pela intersecção das bandas L e b. (a) Sobrancelha relaxada em sua posição fixa original; (b) Banda L binarizada da imagem 4.(a); (c) Banda b binarizada da imagem 4.(a); (d) Intersecção de 4.(b) e 4.(c); (e) Sobrancelha levantada; (f) Banda L binarizada da imagem 4.(e); (g) Banda b binarizada da imagem 4.(e); (h) Interseção de 4.(f) e 4.(g)



(a) Original (d) Segmentação (g) Original (j) Segmentação

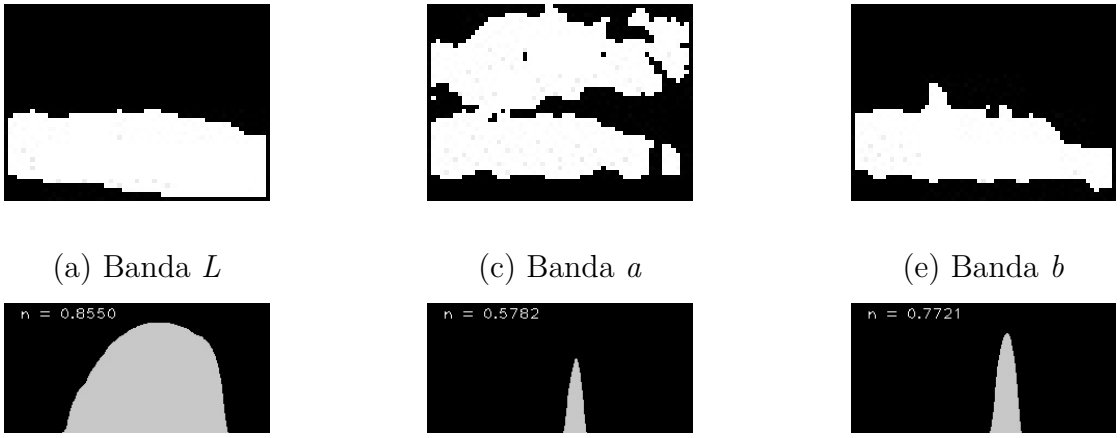
(b) Banda L (e) L binarizada (h) Banda L (k) L binarizada

(c) Banda b (f) b binarizada (i) Banda b (l) b binarizada

Figura 16 – Resultados da intersecção das bandas binarizadas. (a) Sobrancelha original afetada por sombra; (b) Banda L com sombra; (c) Banda b com sombra; (d) Segmentação; (e) Banda L binarizada; (f) Banda b binarizada; (g) Sobrancelha original sem sombreado; (h) Banda L sem sombra; (i) Banda b sem sombra; (j) Segmentação; (k) Banda L binarizada; (l) Banda b binarizada.

3.1.2.1 Determinação do Posicionamento das Sobrancelhas

Para realizar o rastreamento, é computado o vetor de assinatura de bits extraído das imagens das sobrancelhas segmentadas. Esta representação de uma imagem segmentada é computacionalmente barata para calcular e é usada para monitorar o movimento vertical das sobrancelhas. A localização da sobrancelha é dada pela posição da assinatura de bits (Fig. 18.(c)) na qual seu valor é máximo; tal posição indica a linha da imagem segmentada em que a quantia de pixels brancos é maior que em todas as outras linhas (linha vermelha

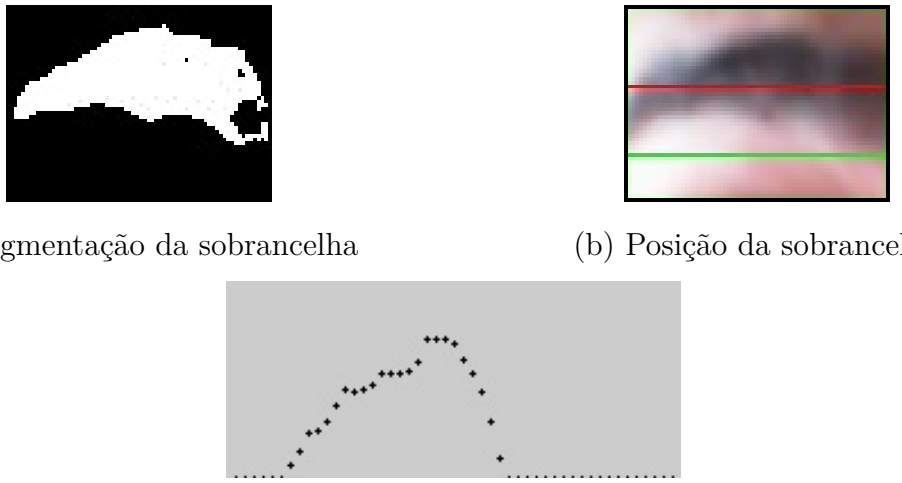


(a) Banda L (c) Banda a (e) Banda b

(b) η Banda L (d) η Banda a (f) η Banda b

Figura 17 – Separabilidade das classes pele e sobrancelha.

da Fig. 18.(b)).



(a) Segmentação da sobrancelha (b) Posição da sobrancelha

(c) Assinatura de bits

Figura 18 – Rastreamento da posição da sobrancelha levantada. (a) Segmentação da sobrancelha; (b) Linha vermelha indicando a posição atual da sobrancelha, e linha a linha verde, a posição fixa encontrada na calibração; (c) Assinatura de bits.

3.1.3 Módulo das Sobrancelhas

Com as explicações recém dadas, é possível descrever como os comandos são gerados à partir do rastreamento das sobrancelhas. No passo de Inicialização do sistema (Fig. 10), a Fase de Calibração, explicada na subseção 3.1.1.1, determina o recorte das regiões das sobrancelhas. Em seguida a segmentação das sobrancelhas ocorre como descrito na

subseção 3.1.2. Depois disso, as posições fixas das sobrancelhas relaxadas (Fig. 18.(b), linha desenhada em verde) são encontradas de acordo com citado na subseção 3.1.2.1.

Após a Inicialização, executa-se o passo de Geração de Comandos como se segue. Os quadros são recortados utilizando as coordenadas dadas pela Fase de Calibração, e então, as sobrancelhas são segmentadas. As posições correntes das sobrancelhas (linha vermelha da Fig. 18.(b)) são encontradas pela análise de suas assinaturas de bits. Finalmente, as posições correntes das sobrancelhas são comparadas às suas respectivas posições originais fixas: se a distância entre elas for maior que um parâmetro P, para as duas sobrancelhas, e as sobrancelhas permanecerem levantadas durante um tempo T suficiente, o comando para chavear o movimento é reconhecido (neste trabalho, $P = 5$, e $T = 0,8$ segundos, determiandos empiricamente).

3.1.4 Interfaces Alternativas Baseadas nas Sobrancelhas

Duas outras abordagens para o Módulo das Sobrancelhas também foram investigadas. Inicialmente, ambas utilizam a detecção das regiões das sobrancelhas como descrito na seção 3.1.1. A diferença na implementação entre as interfaces alternativas ocorre da seguinte maneira:

A primeira abordagem utiliza o método Lucas & Kanade de FO com pirâmides de maneira a criar um campo de fluxo ótico nas regiões das sobrancelhas. Se a maioria dos vetores do campo de FO apontarem para o topo da imagem, então o movimento da sobranecelha é considerado como detectado. O resultado visual deste Módulo das Sobrancelhas é apresentado na Figura 19. As setas vermelhas indicam a direção do movimento das sobrancelhas estimado entre dois quadros.

A segunda aplica a análise de MHI descrita na seção 2.6, para determinar a orientação angular do movimento nas regiões das sobrancelhas recortadas. Se tais orientações angulares diferirem da orientação de 90^o (ângulo reto, apontando para cima) em no máximo 10^o para mais ou para menos, é considerado que houve movimento vertical nas sobrancelhas. Uma ilustração do funcionamento desta interface pode ser visualizado na Figura 20. Os ponteiros dos relógios vermelhos indicam o ânglo detectado durante o movimento das sobrancelhas.

3.1.5 Módulo de Fluxo Ótico

A escolha da posição da câmera no capacete está também relacionada ao fundo da sequência de imagens. Sabendo que a posição da cabeça nas imagens é sempre fixa, gera-se um campo de fluxo ótico na região onde não se encontra a face, e assim, de um quadro para seu consecutivo, indentificam-se os movimentos que a cabeça do usuário realiza.

O campo de FO é computado na sequência de imagens, exceto na região delimitada pelo rosto do usuário; a seleção dessa região onde o campo de FO é computado foi

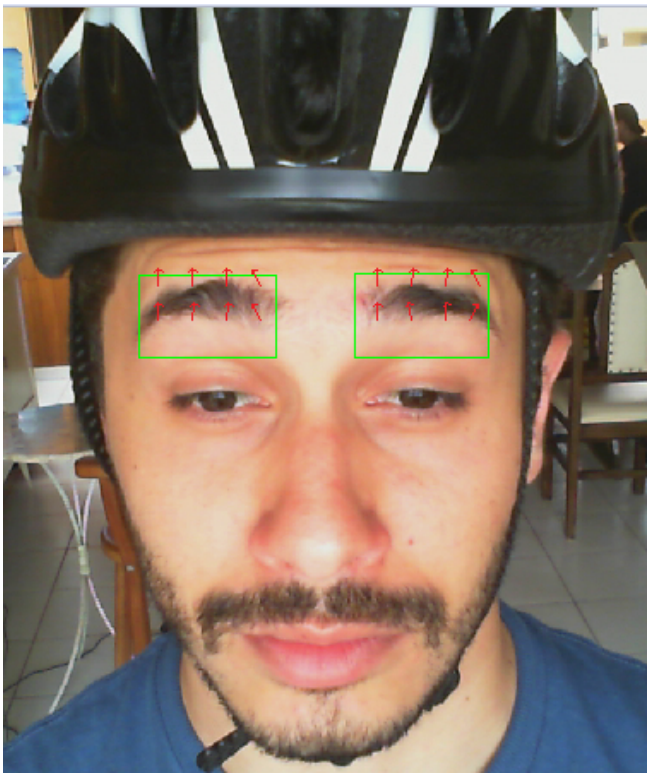


Figura 19 – Campo de fluxo ótico na região das sobrancelhas, calculado entre dois frames consecutivos.

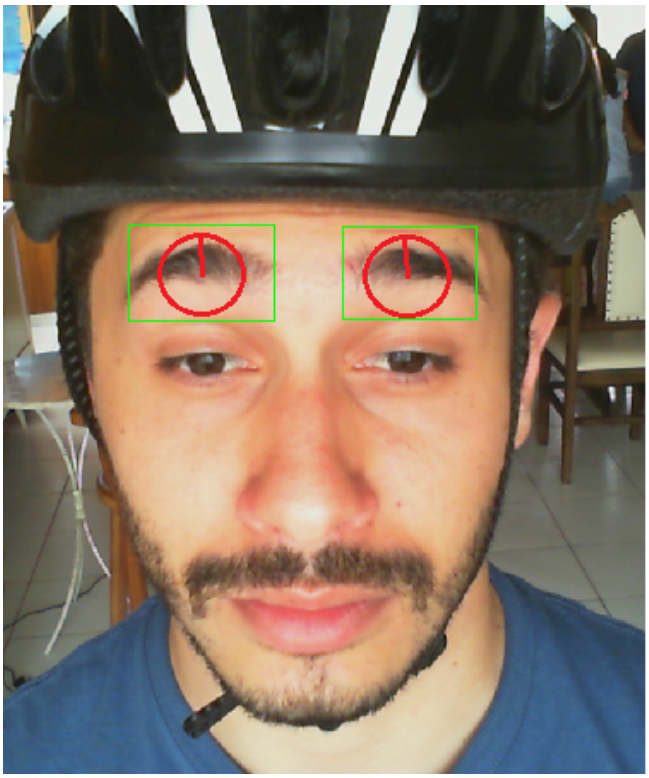


Figura 20 – Orientações angulares das sobrancelhas utilizando MHI.

codificada de maneira empírica no sistema, e é sempre a mesma, para qualquer usuário. Os vetores de deslocamento no campo de FO são subamostrados na imagem, de maneira que cada um deles dista em dez pixels em relação aos seus vizinhos, seja verticalmente ou horizontalmente, como ilustram os vetores azuis da Fig. 21. É importante notar que, apesar dos vetores de deslocamento estarem apontando para a esquerda (mostrando que o fundo se "moveu" da direita para a esquerda), a cabeça se moveu em direção à direita.

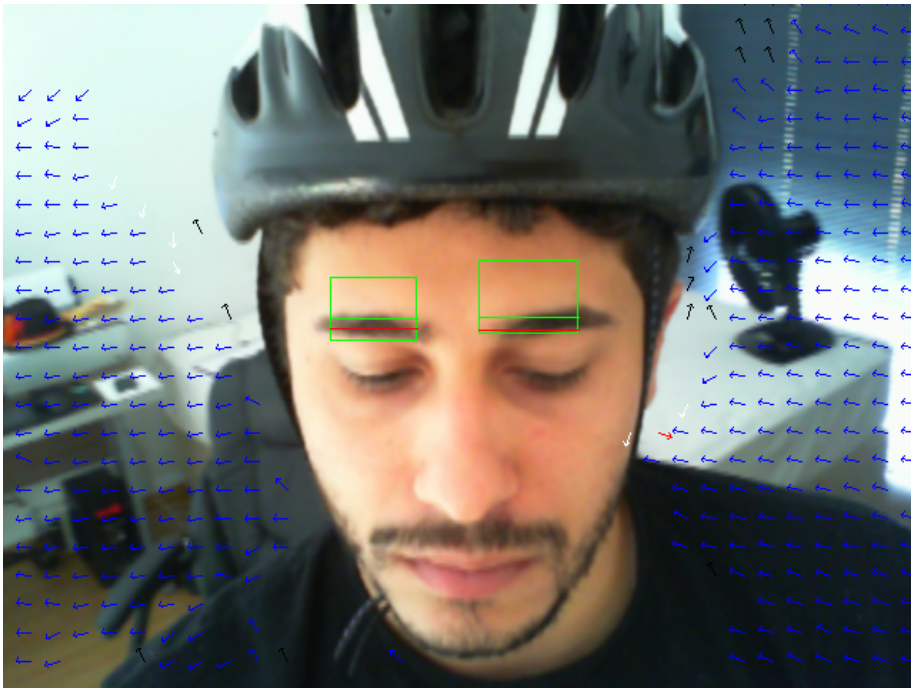


Figura 21 – Campo de fluxo ótico calculado entre dois frames nos quais o usuário moveu sua cabeça da esquerda para a direita

O módulo de FO gera comandos como descrito a seguir: inicialmente, se o MVC está ligado, e o sistema não está se movendo (robô está parado), os vetores de FO são contados, e a direção que possui maior número de vetores apontando para si, determina a direção do movimento da cabeça. No caso de detecção de movimento para cima, com este movimento durando um tempo T_{switch} suficientemente longo (aqui, $T_{switch} = 1$ segundo), o sistema chaveia o funcionamento do MVC, ligando ou desligando-a. Se a maioria dos vetores apontarem para a direita ou esquerda, e novamente, com este movimento durando um tempo T_{steer} suficiente, comandos de giro são gerados.

A determinação da direção dos vetores é dada de acordo com seu ângulo α. Se α se encontrar no intervalo $60^o < \alpha < 120^o$, considera-se que este vetor aponta para cima, e caso α se encontre dentro do intervalo $240^o < \alpha < 300^o$, diz-se que tal vetor aponta para baixo. Quando α variar ou estiver no intervalo fechado $120^o \leq \alpha \leq 240^o$ a direção do vetor é considerada como sendo para a esquerda, e quando α possuir valores tais que $\alpha \leq 60^o$ ou $\alpha \geq 300^o$, sua direção dita como apontando para a direita (ver Fig. 22).

Os comandos de giro podem ser reconhecidos em cinco intensidades diferentes, de

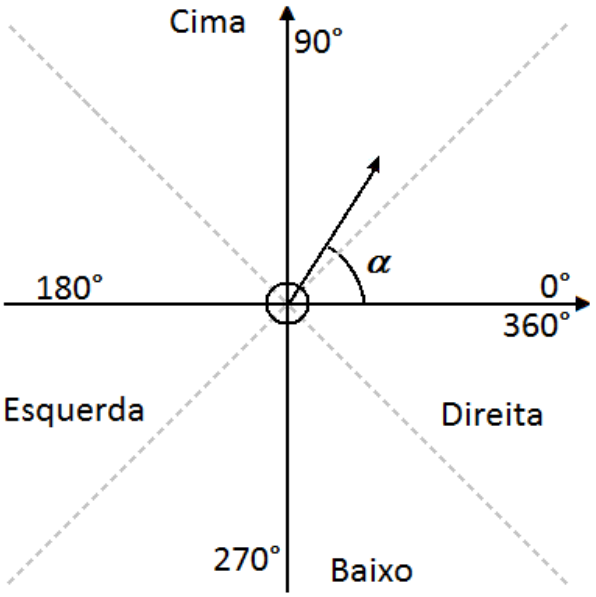


Figura 22 – Classificação do comando de acordo com o ângulo

acordo com o tempo T_{steer}, quanto menor for T_{steer}, menor é o ângulo de giro executado pelo RMR, e quanto maior for T_{steer}, maior será o movimento de giro executado pelo robô. O sistema sempre entra em congelamento logo após o reconhecimento de um comando de giro. Durante este tempo reservado para o congelamento, o MVC não reconhece nenhum outro comando. Isso é importante pois sempre que o usuário vira sua cabeça com o intuito de realizar um giro, é necessário retornar a cabeça para a posição original (olhando para frente), para poder enxergar o que está acontecendo à sua frente, sem que o robô considere este movimento de volta como um novo comando de giro.

3.2 Módulo Robótico

Para avaliar a mobilidade proporcionada pelo MVC, foi desenvolvido o MR, que consistiu na montagem e programação de um RMR que possui os mesmos graus de liberdade que possui uma cadeira de rodas, como citado na seção 2.8. O controle dos atuadores deste RMR é realizado por um programa escrito na linguagem C++, rodando em um Arduino Uno com as configurações que estão descritas também na seção 2.8. Um *transceiver Bluetooth*® é conectado ao Arduino para que exista comunicação entre robô e MVC, que roda em um *laptop* (MVC).

Um sensor ultrassônico também foi fixado à frente do robô, e conectado ao Arduino, para servir como um artifício de segurança, evitando colisões frontais. Devido ao esgotamento dos pinos de entrada e saída do Arduino, não foi possível acoplar sensores simultâneos para o monitoramento do ambiente. Mas como pode ser constatado em outras pesquisas, como em (PLASCENCIA, 2008), vários sensores podem ser usados para evitar colisões e planejar trajetórias.

O funcionamento do MR é ilustrado na Fig. 23. Em um passo inicial, o *transceiver* de comunicação e alguns pinos de entrada e saída são configurados. Na sequência, o programa executa seu laço principal, no qual aguarda pelos caracteres que chegam da comunicação via *Bluetooth*® . Se *R* ou *L* são recebidos, comandos de giro para a direita ou esquerda são rescpectivamente executados. Em caso de recebimento de um comando *F*, o robô começa a se mover para frente, e só para seu movimento se um caracter *S* for recebido, ou se o sensor ultrassônico detectar algum obstáculo a menos de dez centímetros de distância.

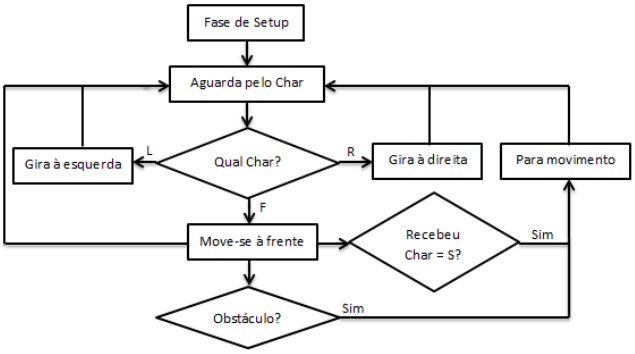


Figura 23 – Fluxo de execução do Módulo Robótico.

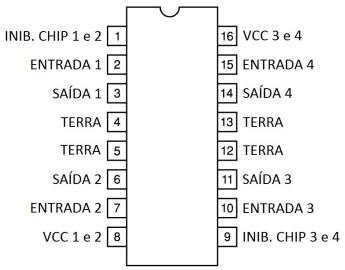


Figura 24 – Pinagem da ponte H, CI-L293D

Para realizar o chaveamento de sinal entre o Arduino e os atuadores, foi utilizado o circuito integrado controlador de motor L293D (ponte H). A pinagem desse circuito pode ser visualizada na Figura 24, e a Figura 25 mostra esse *chip* no circuito que foi desenvolvido para o acionamento das rodas do RMR. O MR final, portando o Arduino, sensor ultrassônico, atuadores, o *plug Bluetooth*® , bem como o *protoboard* com o circuito para o acionamento dos motores pode ser visualizado na Figura 26.

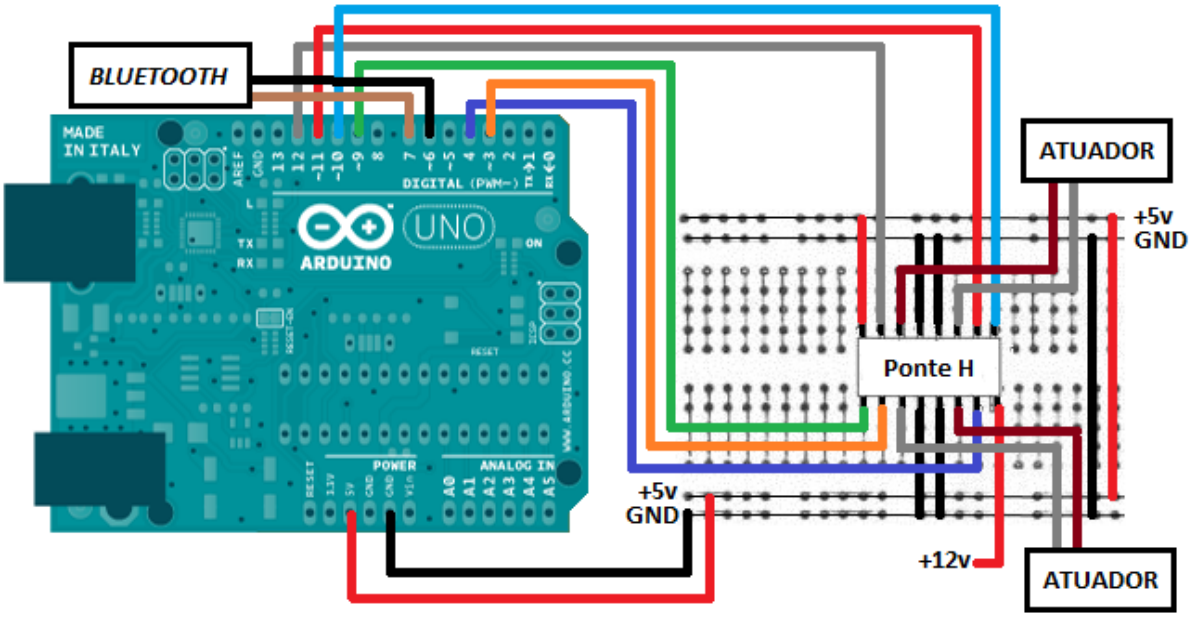


Figura 25 – Circuito de *hardware* para o acionamento das rodas

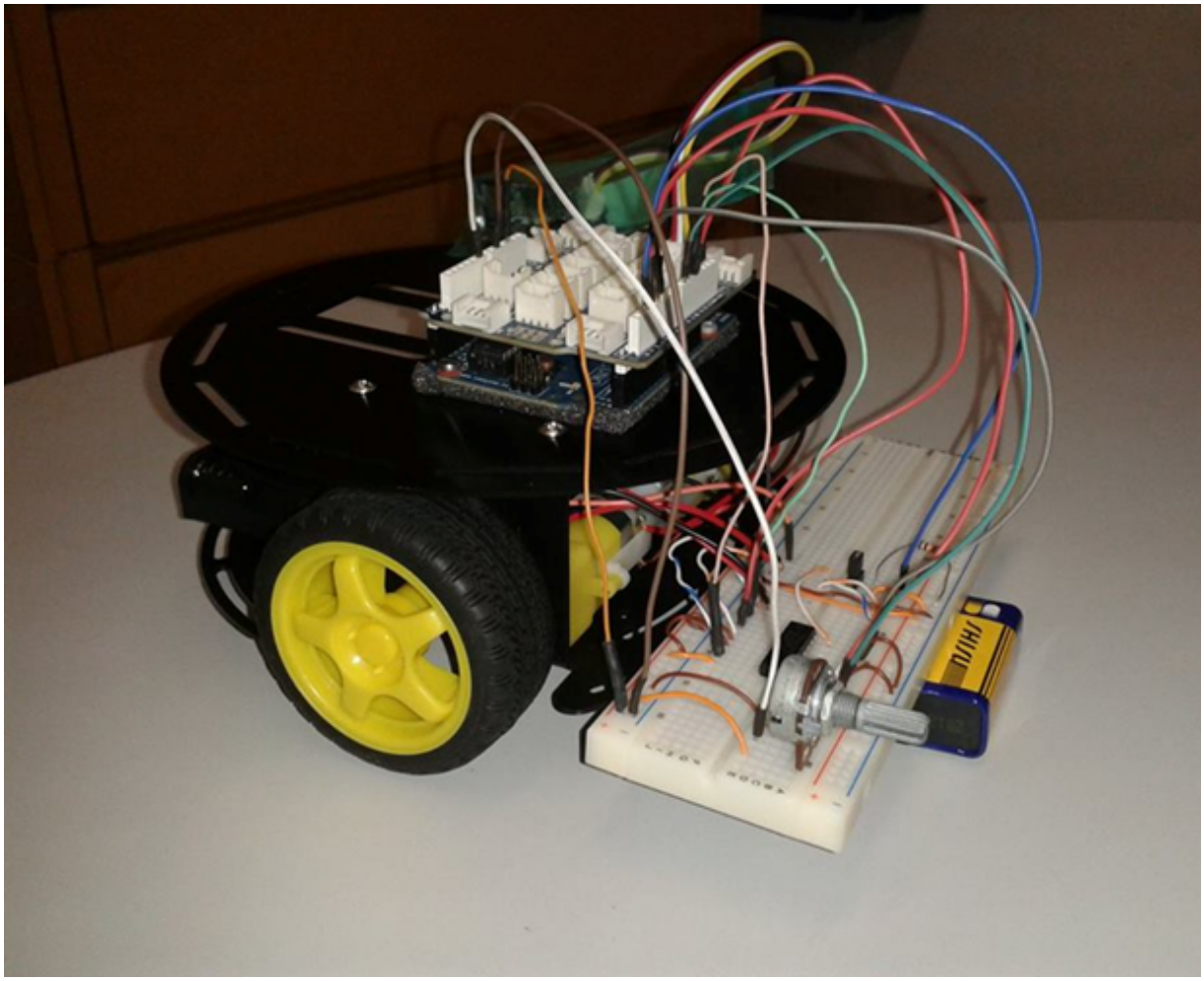


Figura 26 – Robô Turtle-2WD com circuito

4 Resultados Experimentais

Duas baterias de testes foram realizadas para quantificar o desempenho do sistema desenvolvido. Os primeiros testes levaram em consideração o acionamento de comandos pelas sobrancelhas. Esse primeiro conjunto de testes levou em conta diferentes tipos de iluminação ambiente e pessoas com diversos tons de pele. A segunda bateria, foi realizada de maneira que o usuário comandasse os movimentos do RMR em um circuito em ambiente controlado.

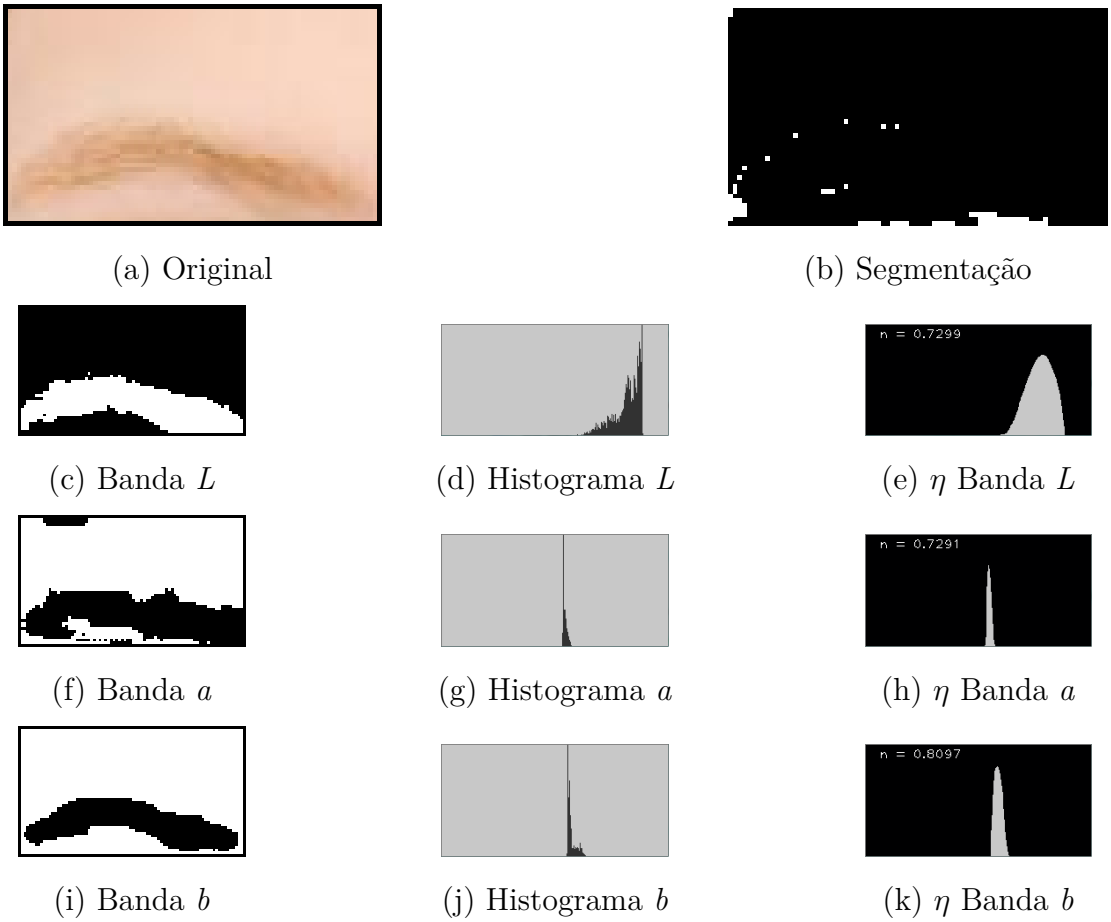
4.1 Resultados do Módulo das Sobrancelhas

Para avaliar a técnica proposta utilizando assinaturas de bits, e realizar uma comparação desta com as outras duas abordagens investigadas, vinte e duas pessoas foram filmadas através do capacete com a câmera. A idade dos usuários variou de dez a trinta e seis anos de idade. Pessoas com diferentes tons de pele foram filmadas, e os vídeos foram tomados nas seguintes condições de iluminação:

- Ambiente interno com boa iluminação (janelas abertas, ou fonte de luz luminescente);
- Ambiente interno com muito pouca iluminação (ambiente fechado com todas as luzes apagadas e pouquíssima iluminação externa);
- Ambiente externo com tempo nublado; e
- Ambiente externo ensolarado.

Ambientes internos, iluminados pelo sol proveniente das janelas, iluminados com lâmpadas luminescentes, levaram a uma boa segmentação das sobrancelhas. Assim, em todos os casos a técnica proposta foi capaz de rastrear as posições das sobrancelhas precisamente. Figuras 14.(a), 15.(a), 15.(e) e 18.(b) são exemplos de recortes das sobrancelhas em ambientes internos. Uma exceção causada pela combinação entre pele e cor da sobrancelha foi identificada, nos casos em que há muito pouco contraste entre os dois. A intersecção entre as bandas binarizadas não é capaz de segmentar a sobrancelha corretamente, com pessoas que possuem pele e cor de sobrancelha muito clara (muito loira ou albina), por exemplo (Fig. 27). Tonalidades de pele escura, que teoricamente também apresentariam baixo contraste puderam ser segmentadas corretamente (o 5 agrupa mais imagens que ilustram esse, e outros casos). Para contornar esse problema, poderiam ser utilizados marcadores nas regiões das sobrancelhas, de maneira a aumentar o contraste.

Ambientes externos nos quais a iluminação é proveniente de iluminação solar difusa (tempo nublado) apresentaram resultados de segmentação similares aos dos ambientes



(a) Original (b) Segmentação

(c) Banda L (d) Histograma L (e) η Banda L

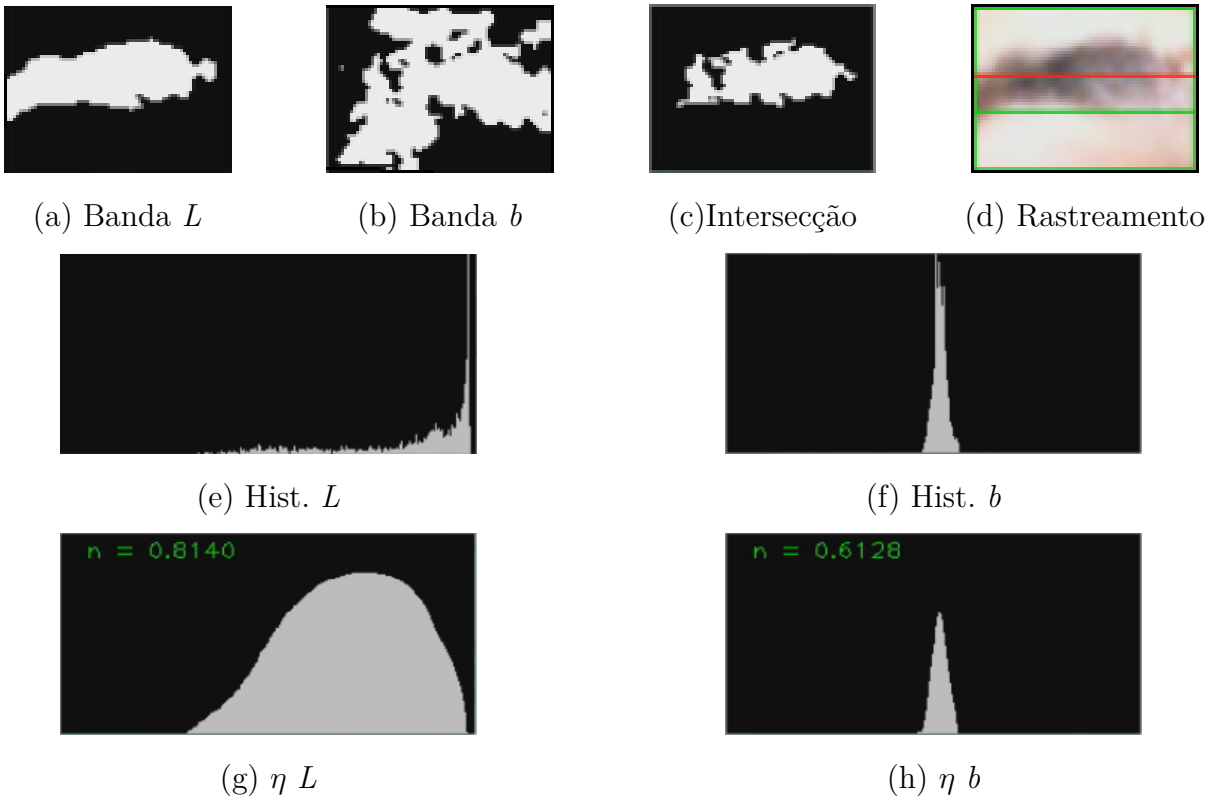
(f) Banda a (g) Histograma a (h) η Banda a

(i) Banda b (j) Histograma b (k) η Banda b

Figura 27 – Resultados com pele e sobrancelhas muito claras. (a) Recorte da sobrancelha; (b) Segmentação resultante; (c) Banda L binarizada; (d) Histograma da banda L; (e) Índice η da banda L; (f) Banda a binarizada; (g) Histograma da banda a; (h) Índice η da banda a; (i) Banda b binarizada; (j) Histograma da banda b; (k) Índice η da banda b;

internos. Iluminação solar, incidindo diretamente nas sobrancelhas de maneira uniforme, apresentou resultados piores quando comparados aos resultados de ambientes internos. A Figura 28 mostra que é possível determinar a posição das sobrancelhas com sucesso nesse caso, mas a imagem binária mostra uma segmentação em que a sobrancelha aparece menor do que realmente é. Isso acontece pelo influência negativa da banda b binarizada sob a segmentação, pois a informação de cor nesse caso foi pouco útil, como sugere o, relativamente baixo, valor do índice η dessa banda (Fig. 28.(h)). Os valores do índice η obtidos com os resultados de segmentação em iluminação solar apontam que a banda L é melhor definida quando há iluminação forte.

Uma outra questão relacionada à iluminação solar direta, é que dependendo do ângulo de incidência da luz, são geradas sombras nas regiões de recorte, fazendo com que a limiarização confunda sombras na pele com as sobrancelhas, como pode ser visto na Figura 29. Nesse caso, ambas as bandas não foram úteis em seperar a pele da sobrancelha, pois as duas sofreram forte influência da sombra presente no recorte.



(a) Banda L (b) Banda b (c)Intersecção (d) Rastreamento

(e) Hist. L (f) Hist. b

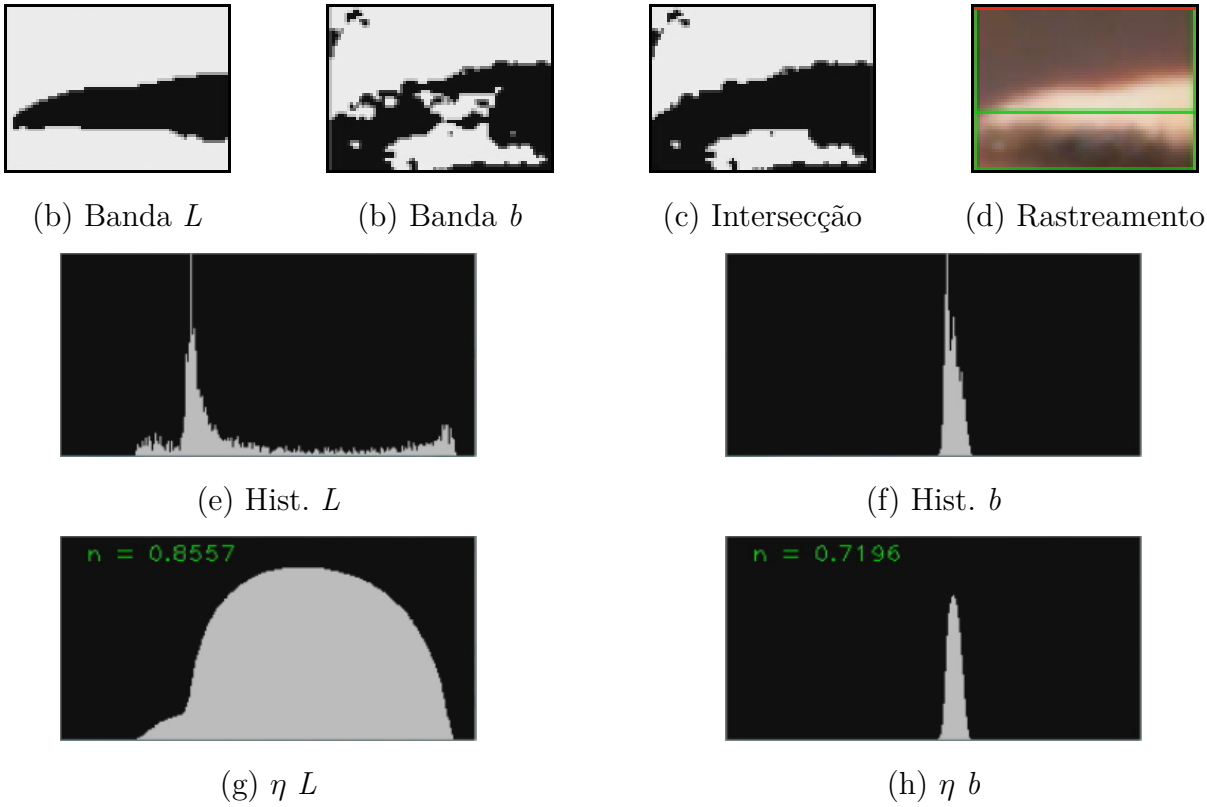
(g) η L (h) η b

Figura 28 – Resultados da segmentação com iluminação solar incidindo diretamente na sobrancelha. (a) Binarização da banda L; (b) Binarização da banda b; (c) Intersecção das bandas; (d) Rastreamento resultante; (e) Histograma da banda L; (f) Histograma da banda b; (g) Índice η da banda L; (g) Índice η da banda b

Como já comentado na seção 3.1.2, no decorrer dos testes foi decidido remover a banda a da composição da segmentação, pois os casos testados mostraram que esta banda tem a tendência de apresentar um comportamento diferente das outras duas quando limiarizada. Os resultados experimentais também indicaram que a banda a é mais afetada pelas mudanças na iluminação do que a banda b.

Cada um dos vídeos de teste possui uma sequência de vinte e oito comandos de sobrancelha. Cada comando foi seguido por uma pausa em que as sobrancelhas se mantiveram relaxadas nas suas posições fixas, representando os momentos em que o usuário não deseja gerar comandos. Os vídeos também incluiram as situações em que o usuário trafega através de diferentes ambientes e condições de iluminação, simulando mudanças repentinas na luminosidade e também as trepidações que a câmera pode sofrer. Foi possível executar todos os vídeos em tempo real, em uma taxa aproximada de oito quadros por segundo.

Duas medidas de precisão foram calculadas para avaliar os testes. A Taxa de Positivos Verdadeiros (TPV) é o número total de Positivos Verdadeiros (PV) divididos pela soma do número de PV mais o número de Negativos Falsos (NF). A outra medida a



(b) Banda L (b) Banda b (c) Intersecção (d) Rastreamento

(e) Hist. L (f) Hist. b

(g) η L (h) η b

Figura 29 – Resultados da segmentação com iluminação solar gerando sombra na região das sobrancelhas. (a) Binarização da banda L; (b) Binarização da banda b; (c) Intersecção das bandas; (d) Rastreamento resultante; (e) Histograma da banda L; (f) Histograma da banda b; (g) Índice η da banda L; (g) Índice η da banda b

Taxa de Negativos Verdadeiros (TNV), que calcula o número total de Negativos Verdadeiros (NV) divididos pela soma do número de NV mais o número de Positivos Falsos (PF). Considera-se um PV quando o comando foi executado, e o sistema o detectou corretamente; um NF é o caso em que o comando foi executado, mas o sistema falhou em reconhecê-lo; PF é quando sistema reconhece um comando, sem que ele tenha de fato tenha sido realizado pelo usuário; e o NV é quando o sistema não reconhece comando algum, sendo que de fato nada aconteceu (MARTINS; DOMINGUES, 2012).

Ocorre que as TPV e TNV são as probabilidades de um comando ser reconhecido corretamente, e de o sistema não reconhecer comandos que não foram executados. Diz-se que a Taxa de Negativos Falsos (TNF) é probabilidade de o sistema não reconhecer um comando que foi executado pelo usuário, e que a Taxa de Positivos Falsos (TPF) é a chance de reconhecer um comando que não foi realizado pelo usuário (MARTINS; DOMINGUES, 2012). As tabelas a seguir demonstram a performance do módulo de interface baseada nas sobrancelhas

As Tabelas 1 e 2 comparam os métodos testados considerando as diversas condições iluminações dos ambientes. Nesses casos a técnica proposta supera as abordagens baseadas

em FO e MHI em reconhecer os comandos gerados pelas sobrancelhas. Os métodos de FO e MHI provaram ser mais sucetíveis a variações na iluminação e trepidações da câmera. Entretando, o método proposto apresentou performance inferior nas condições em que o ambiente interno possuia pouquíssima iluminação. Sendo assim, o reconhecimento de comandos por meio da segmentação não é adequado para ambientes onde há pouca ou nenhuma iluminação.

As Tabelas 3 e 4 consideram as diferentes tonalidades de pele. Os experimentos foram realizados por sete usuários, com diferentes tons de pele: um negro, dois pardos e quatro usuários brancos. Novamente, o método proposto obteve os melhores resultados, neste caso, não considerando as condições de iluminação, indicando que o método seria adequado para variádos tons de pele.

Tabela 1 – TPV. Iluminação × Métodos.

	Método Proposto	Fluxo Ótico	MHI
Interno Escuro	62,50%	20,83%	14,58%
Interno Iluminado	95,53%	84,15%	85,27%
Externo Ensolarado	85,52%	63,16%	78,95%
Externo Nublado	100%	72,92%	75%

Tabela 2 – TNV. Iluminação × Métodos.

	Método Proposto	Fluxo Ótico	MHI
Interno Escuro	92,86%	76,19%	95,24%
Interno Iluminado	97,23%	79,15%	85,96%
Externo Ensolarado	93,51%	74,03%	79,22%
Externo Nublado	96,43%	67,86%	78,57%

Tabela 3 – TPV. Tons de pele × Métodos.

	Método Proposto	Fluxo Ótico	MHI
Branca	90,88%	80,29%	82,06%
Parda	94,05%	69,05%	79,17%
Negra	92,86%	72,32%	65,18%

Tabela 4 – TNV. Tons de pele × Métodos.

	Método Proposto	Fluxo Ótico	MHI
Branca	96,32%	84,14%	87,25%
Parda	95,43%	73,14%	85,14%
Negra	98,29%	63,25%	78,68%

As Tabelas de 5 a 10 apresentam os resultados experimentais de uma maneira mais específica. Nessas Tabelas, TPV e TNV dão foco aos tipos de tons de pele por cada

Tabela 5 – TPV da técnica proposta. Iluminação × Tons de pele.

	Interno Escuro	Interno Iluminado	Externo Nublado	Externo Ensolarado
Branca	70%	92,65%	100%	82,14%
Parda	66,67%	100%	100%	78,57%
Negra	50%	100%	100%	100 %

Tabela 6 – TNV para a técnica proposta. Iluminação × Tons de pele.

	Interno Escuro	Interno Iluminado	Externo Nublado	Externo Ensolarado
Branca	84,21%	97,88%	100%	85,71%
Parda	100%	94,96%	94,74%	96,15%
Negra	100%	98,53%	92,86%	100 %

Tabela 7 – TPV para a abordagem de Fluxo Ótico. Iluminação × Tons de pele.

	Interno Escuro	Interno Iluminado	Externo Nublado	Externo Ensolarado
Branca	25%	87,5%	60%	64,29%
Parda	16,67%	75,89%	87,5%	53,57%
Negra	18,75%	84,38%	75%	75 %

condição de iluminação. Por exemplo, a Tabela 5 mostra as TPV alcançadas pelo método proposto: note que, comparada às outras abordagens (ver Tabelas 7 e 9), a técnica de rastreamento das sobrancelhas pelo método proposto é mais robusta, independentemente das condições de iluminação ou dos tons de pele. Além disso, comparando-se a Tabela 6 com as Tabelas 8 e 10, pode-se concluir que, a técnica proposta reconhece falsos comandos com menor intensidade do que as outras interfaces investigadas.

A Tabela 11 resume os resultados obtidos nos testes com o reconhecimento dos comandos de sobrancelha, mostrando que entre as abordagens desenvolvidas, o método proposto utilizando assinatura de bits, apresentou os melhores resultados.

Há uma relação de custo-benefício entre TPV e TNV, neste trabalho optou-se por dar igual importância a esses dois tipos de medidas, pois deseja-se que os comandos sejam corretamente reconhecidos sem que hajam disparos em falso, de maneira igualitária. Entende-se que reconhecer falsos comandos de parar ou andar, é tão importante quanto acionar os comandos corretos quando o usuário assim desejar. As taxas de acerto utilizando as sobrancelhas são sensíveis a duas variáveis: tempo para reconhecimento ($T = 0,8$ segundos), e o parâmetro de distância ($P = 5$ pixels).

4.2 Geração de Comandos e Pilotagem do Robô: Resultados e Discussões

Um RMR (MR) representando a cadeira de rodas foi comandado utilizando o protótipo da interface de comandos (MVC). Para testar a integração entre robô e MVC,

Tabela 8 – TNV para a abordagem de Fluxo Ótico. Iluminação × Tons de pele.

	Interno Escuro	Interno Iluminado	Externo Nublado	Externo Ensolarado
Branca	89,74%	85,16%	69,57%	82,14%
Parda	72,73%	73,11%	68,42%	76,92%
Negra	58,33%	64,71%	60,87%	60,87 %

Tabela 9 – TPV para a abordagem de MHI. Iluminação × Tons de pele.

	Interno Escuro	Interno Iluminado	Externo Nublado	Externo Ensolarado
Branca	15%	87,5%	75%	82,14%
Parda	25%	83,93%	81,25%	82,14%
Negra	6,25%	78,13%	66,67%	70 %

Tabela 10 – TNV para a abordagem de MHI. Iluminação × Tons de pele.

	Interno Escuro	Interno Iluminado	Externo Nublado	Externo Ensolarado
Branca	94,74%	87,99%	78,26%	82,14%
Parda	100%	84,87%	84,21%	80,77%
Negra	91,67%	79,41%	71,43%	73,91 %

uma pista um ambiente interno controlado com iluminação externa e luzes luminescente, foi montada para que o usuário trafegasse com o RMR por um circuito. A Figura 30 mostra esse o circuito iluminado por cada tipo de iluminação supracitado. Essa pista possuia um conjunto de obstáculos, corredor, portas, rampas de aclive e declive que poderiam ser encontrados em uma situação real. Os quinze usuários não-tetraplégicos permaneciam sentados à frente da pista enquanto realizavam os testes.

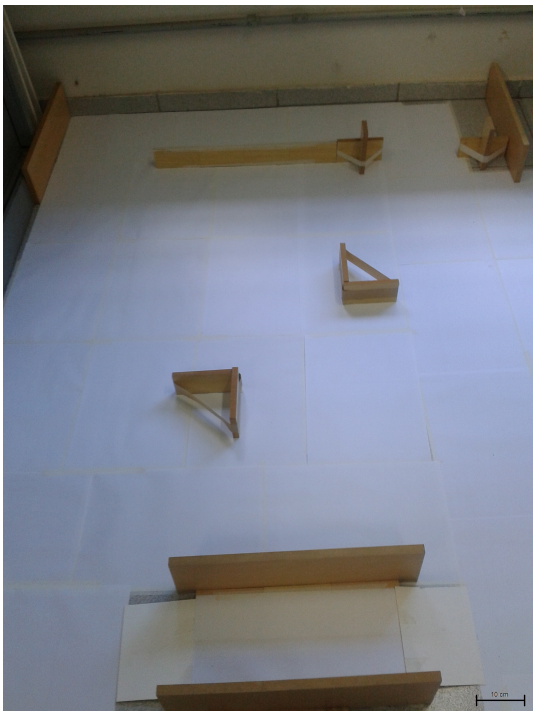
A Figura 31 mostra um diagrama representando o circuito em que o robô trafegou durante os testes experimentais. Após um curto treinamento, o usuário era solicitado a conduzir o robô através da trajetória por três vezes, de maneira a validar a geração de comandos e a execução dos mesmos. O cumprimento desses testes levaram à execução de um total de 1711 comandos, dos quais 94,64% foram executados adequadamente num tempo médio de 4 minutos e 40 segundos por usuário. A Tabela de confusão mostrada na Tabela 12 demonstra a relação entre quais comandos o usuário quis de executar, e quais comandos os protótipos realmente reconheceram e executaram.

Como pode ser visto, o falso reconhecimento entre os comandos de *girar à direita* e *girar à esquerda* apresentaram as maiores taxas para esse teste. Isso pode ser explicado pelo fato de o método de reconhecimento ser baseado no período de tempo em que o comando é executado continuamente pelo usuário. Se o movimento realizado pelo usuário não dura tempo suficiente, a interface não reconhecerá este comando.

Por exemplo, considere uma situação em que o usuário deseja girar à esquerda, mas ele não vira sua cabeça nessa direção por tempo suficiente (ao menos 0,8 segundos). Pode acontecer que o movimento de volta da cabeça (em direção à direita), para retornar à visão frontal, leve o tempo suficiente (mais que 0,8 segundos) para ser reconhecido como um

Tabela 11 – TPV e TNV para cada abordagem.

	Método Proposto	FO	MHI
TPV	92,1%	75,81%	84,52%
TNV	96,43%	77,36%	85,12%



(a) Iluminação externa.

(b) Iluminação luminescente.

Figura 30 – Fotos do circuito montado em ambiente controlado. (a) Pista com fonte de iluminação externa; (b) Pista com fonte de iluminação luminescente.

Tabela 12 – Tabela de confusão do reconhecimento dos comandos. Linhas contêm os comandos reconhecidos, e as colunas representam os comandos que usuário pretendia executar.

	Direita	Esquerda	Sobrancelhas	Ligar/Desligar	Nada
Direita	95,12%	5,2%	0%	0,51%	0%
Esquerda	4,47%	94,69%	0%	0,47%	0%
Sobrancelhas	0%	0%	97,06%	0%	2,02%
Ligar/Desligar	0,41%	0,11%	0%	91,69%	0%
Nada	0%	0%	2,94%	7,33%	97,98%

comando de giro, gerando uma falso reconhecimento. Este erro, para efeitos de avaliação, conta como um erro da interface.

Um outro erro também pode acontecer quando o usuário realiza um movimento de giro, e não retorna sua cabeça à posição central a tempo (considerando o tempo de congelamento - subseção 3.1.5, capítulo 3). Neste caso, o movimento pretendido é executado adequadamente, mas um movimento “residual” na direção oposta também será

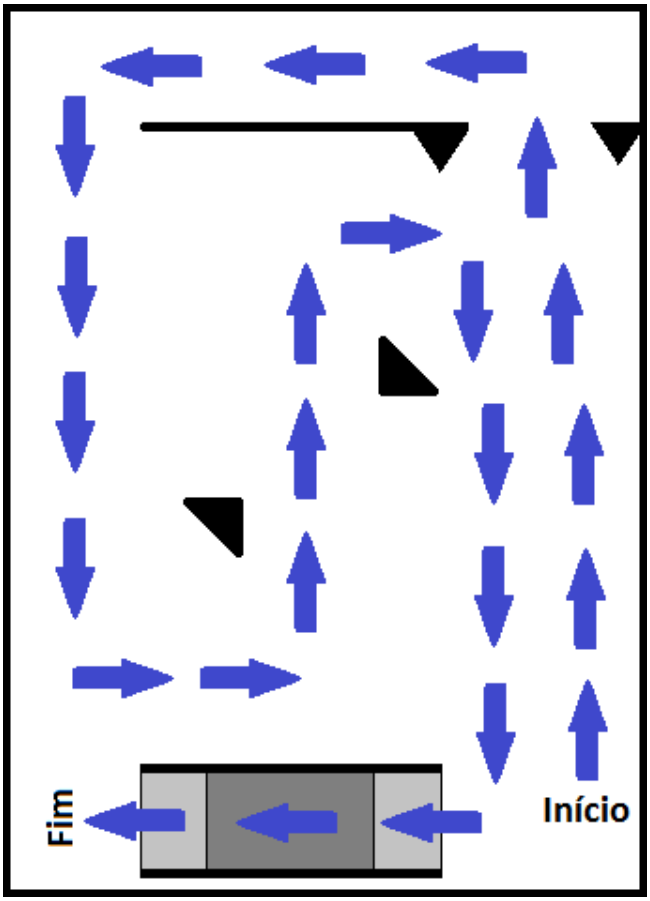


Figura 31 – Circuito realizado pelo RMR.

reconhecido, de maneira indesejada.

Pode ser que também ocorram ligamentos e desligamentos do MVC erronamente. Nesse caso, o usuário não deseja desligar o MVC, mas ele acaba realizando o movimento com a cabeça para cima acidentalmente, causando o reconhecimento indesejado desse comando. Este erro pode ser considerado um tipo de falha humana.

Os erros de reconhecimento dos comandos das sobrancelhas podem ocorrer quando o usuário não mantém sua sobrancelha levantada por tempo suficiente, ou ele realiza movimentos curtos, não alcançando a distância mínima entre a posição fixa das sobrancelhas relaxadas, e e a posição delas levantadas. Em algumas ocasiões, comandos das sobrancelhas foram reconhecidos incorretamente devido à trepidação do capacete.

Ao fim dos testes, os usuários eram perguntados sobre as impressões que tiveram a respeito da interface de comandos. A maioria dos testadores afirmou que eles poderiam alcançar melhores resultados caso tivessem mais tempo para treinar.

Os resultados gerais apontam que os comandos de chaveamento do acionamento do MVC apresentaram 91,69% taxa de reconhecimento correto. Os comandos de giro foram reconhecidos adequadamente em 94,91% dos casos. Já os comandos das sobrancelhas (chaveamento de movimento) foram identificados corretamente em 97,06% dos casos.

5 Conclusão

Este trabalho introduziu uma nova interface de comandos para uma cadeira de rodas motorizada. O sistema completo é composto pela interface de visão computacional, e o módulo robótico que executa os comandos que seriam realizados pela cadeira de rodas. O método de interpretação das instruções do usuário foi baseado em uma técnica de segmentação e rastreamento das sobrancelhas e na análise do fluxo ótico ao fundo das imagens, para que fosse possível controlar uma cadeira de rodas projetada para pessoas que podem realizar os movimentos do pescoço para cima apenas.

A técnica de segmentação e rastreamento das sobrancelhas é fundamentada no recorte adequado da região contendo sobrancelhas e pele da testa, esperando-se obter histogramas bimodais para cada uma das bandas da imagem de recorte representadas pelo modelo de cores Lab. A segmentação das sobrancelhas é alcançada através da intersecção das bandas L e b limiarizadas pelo método de Otsu. Esta combinação provê bons resultados pois há situações em que a segmentação das banda b compensa as situações onde a banda L não tem uma boa segmentação, e vice-versa.

Esta técnica foi testada em diferentes condições de iluminação e com diferentes tons de pele. Ambientes internos apresentaram resultados melhores que os externos, pela possibilidade de maior controle na iluminação. A cor da pele e das sobrancelhas desempenham papel importante na tarefa de segmentação, pois quanto maior o contraste entre elas, melhores os resultados. Para fins de comparação, foram implementadas duas outras abordagens na detecção de comandos pela sobrancelha, uma baseada em FO e outra em MHI.

Os testes apresentados na seção 4.1 demonstram que, dentre as implementações para geração de comandos via sobrancelha, a abordagem via segmentação, rastreamento e assinatura de bits foi a mais precisa. Esses primeiros resultados foram submetidos, aceitos e serão publicados nos anais da *2015 IEEE Region 8* 16^a edição da *International Conference on Computer as a Tool (EuroCon 2015)*, (OLIVEIRA; FLORES; ALMEIDA, 2015b).

Os testes de pilotagem do robô apresentaram as seguinte taxas de acerto: os comandos de giro e ativação do MVC, que são gerados pelo Módulo de Fluxo Ótico, apresentaram em média 93,34% de acerto; os comandos de mover e parar, gerados pelo Módulo das Sobrancelhas, foram reconhecidos corretamente em 97,06% das vezes; e a falsa detecção de comandos ocorreu em apenas 3,02% das ocasiões. Assim, pode-se concluir que análise do campo de fluxo ótico ao fundo das imagens permitiu estimar os movimentos da cabeça do usuário adequadamente, tornando possível o reconhecimento correto comandos dos comandos baseados nessa técnica. Esses últimos números foram incorporados aos resultados e submetidos a um periódico da área de visão computacional (OLIVEIRA; FLORES; ALMEIDA, 2015a).

Trabalhos futuros podem incluir, mas não estão limitados, as seguinte opções:

- Integração da interface proposta a uma cadeira de rodas motorizada real;
- Desenvolvimento de outras aplicações para a interface, como controlar de um ponteiro de *mouse*, ou comandar outros dispositivos como uma casa automatizada;
- Escolha ou desenvolvimento de técnicas de segmentação e análise de imagens alternativas para melhorar a precisão da interface, enquanto mantém a característica de tempo real. Por exemplo, incorporar algum método de descarte de quadros; Melhorar a segmentação adicionando novas técnicas; etc;
- Adição de comandos, por exemplo, baseados nos lábios, piscadas ou movimentação do olhar;
- Os testes realizados mostraram que, para alguns usuários, a adição de uma calibração da quantidade de movimento pode ser necessária para reconhecer um comando pelas sobrancelhas. Cada pessoa possui sua própria "capacidade de movimento" e isso pode não ser um parâmetro fixo para todos os casos;
- Desenvolver uma solução que permita mudar a direção do robô enquanto ele está se movimentando.

Referências

ALMEIDA, C. R. *Desenvolvimento do Protótipo de uma Cadeira de Rodas Comandada por Voz*. Dissertação (Mestrado) — Universidade Federal de Sergipe, São Cristóvão - SE, Janeiro 2007. Citado na página 14.

ARDUINO; BANZI, M.; CUARTIELLES, D.; IGOE, T.; MARTINO, G.; MELLIS, D. *Arduino Uno*. Acessado em 15/07/2015. Disponível em: <http://arduino.cc/en/Main/arduinoBoardUno>. Citado na página 36.

BASTOS-FILHO, T.; FERREIRA, A.; CAVALIERI, D.; SILVA, R.; MULLER, S.; PéREZ, E. Multi-modal interface for communication operated by eye blinks, eye movements, head movements, blowing/sucking and brain waves. *IEEE 2013 ISSNIP Biosignals and Biorobotics Conference (BRC)*, p. 1–6, 2013. Citado 3 vezes nas páginas 14, 15 e 16.

BASTOS-FILHO, T. F.; CHEEIN, F. A.; MULLER, S. M. T.; CELESTE, W. C.; CRUZ, C. de la; CAVALIERI, D. C.; SARCINELLI-FILHO, M.; AMARAL, P. F. S.; PEREZ, E.; SORIA, C.; CARELLI, R. Towards a new modality-independent interface for a robotic wheelchair. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, XX, p. 567–584, 2013. Citado 2 vezes nas páginas 15 e 16.

BEAUCHEMIN, S. S.; BARRON, J. L. The computation of optical flow. *ACM Computation Surey.*, v. 27, p. 433–466, 1995. Citado 2 vezes nas páginas 26 e 27.

BETTADAPURA, V. Face expression recognition and analysis: The state of the art. *CoRR*, abs/1203.6722, 2012. Citado na página 17.

BOUGUET, J.-Y. Pyramidal implementation of the lucas kanade feature tracker: Description of the algorithm. *Intel Corporation Microprocessor Research Labs*, 2000. Citado 2 vezes nas páginas 29 e 30.

BRADSKI, G. R.; DAVIS, J. W. Motion segmentation and pose recognition with motion history gradients. *Fifth IEEE Workshop on Applications of Computer Vision*, v. 13, p. 238–244, 2002. Citado 3 vezes nas páginas 31, 32 e 33.

BUSIN, L.; VANDENBROUCKE, N.; MACAIRE, L. Color spaces and image segmentation. *Advances in imaging and electron physics*, Elsevier, v. 151, p. 65–168, 2009. Citado na página 23.

CAMPION; GUY; CHUNG; WOOJIN. Wheeled robots. *Springer Handbook of Robotics*, Springer, p. 391–410, 2008. Citado 2 vezes nas páginas 19 e 35.

CAMPION, G.; BASTIN, G.; D'ANDRÉA-NOVEL, B. Structural properties and classification on kinematic and dynamic models of wheeled mobile robots. *Nelineinaya Dinamika - Russian Journal of Nonlinear Dynamics*, Udmurt State University, v. 7, p. 733–769, 2011. Citado 2 vezes nas páginas 19 e 35.

CARLSON, T.; DEMIRIS, Y. Increasing robotic wheelchair safety with collaborative control: Evidence from secondary task experiments. *IEEE International Conference on Robotics & Automation*, p. 5582–5587, 2010. Citado na página 16.

CARLSON, T.; DEMIRIS, Y. Collaborative control for a robotic wheelchair: Evaluation of performance, attention, and workload. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 42, p. 876–888, 2012. Citado na página 16.

DFROBOT. *Turtle-2WD Mobile Platform - Instruction Manual*. 2013. Citado 2 vezes nas páginas 7 e 35.

ESCOBEDO, A.; RIOS-MARTINEZ, J.; SPALANZANI, A.; LAUGIER, C. Context-based face cotnrol of a robotic wheelchair. *IROS Workshop on Navigation and Manipulation Assistance for Robotic Wheelchairs*, 2012. Citado na página 15.

FARIA, P. M.; BRAGA, R. A. M.; VALGôDE, E.; REIS, L. P. Interface framework to drive and intelligent weelchair using facial expressions. *IEEE ISIE 2007 - International Symposium on Industrial Electronics*, p. 1791–1796, 2007. Citado na página 16.

FERNANDES, B. de P.; SILVA, V. A. S. da; PISTORI, H. Protótipo de um simulador para cadeiras de rodas guiadas por expressões faciais: Estudos preliminares. *III Congresso Catarinense de Software Livre - SOLISC*, 2005. Citado na página 16.

FIA, P.; HU, H. H.; LU, T.; YUAN, K. Head gesture recognition for hands-free control of an intelligent wheelchair. *Industrial Robot: An International Journal*, p. 60–68, 2007. Citado 2 vezes nas páginas 16 e 17.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, v. 55, n. 1, p. 119–139, ago. 1997. Citado na página 22.

GAJWANI, P. S.; CHHABIRA, S. A. Eye motion tracking for wheelchair control. *International Journal of Information Technology and Knowledge Management*, p. 185–187, 2010. Citado na página 16.

GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006. ISBN 013168728X. Citado na página 26.

HAN, J.-S.; BIEN, Z. Z.; KIM, D.-J.; LEE, H.-E.; KIM, J.-S. Human-machine interface for wheelchair control with emg and its evaluation. *25th Annual International Conference of the IEEE EMBS*, p. 1602–1608, 2003. Citado 2 vezes nas páginas 14 e 15.

IBGE. *Censo Demográfico 2010 - Características gerais da população, religião e pessoas com deficiência*. 2010. Disponível em: <ftp://ftp.ibge.gov.br/Censos/Censo_Demografico_2010/Caracteristicas_Gerais_Religiao_Deficiencia/caracteristicas_religiao_deficiencia.pdf>. Citado na página 14.

INTEL. *Open Source Computer Vision*. Acessado em 15/07/2015. Disponível em: <http://opencv.org/>. Citado na página 33.

JU, J. S.; SHIN, Y.; KIM, E. Y. Inteligent wheelchair (iw) interface using face and mouth recognition. *IUI '09 Proceedings of the 14th international conference on Intelligent user interfaces*, p. 307–314, 2009. Citado na página 37.

KUNO, Y.; SHIMADA, N.; SHIRAI, Y. A robotic wheelchair based on the integration of human and environmental observations. *IEEE Robotics & Automation Magazine*, p. 26–36, 2003. Citado na página 16.

KUPETZ, D. J.; WENTZELL, S. A.; BUSHA, B. F. Head motion controlled power wheelchair. *Proceedings of the 2010 IEEE 36th Annual Northeast Bioengineering Conference*, p. 1–2, 2010. Citado 2 vezes nas páginas 15 e 16.

LEVINE, S. P.; BELL, D. A.; JAROS, L. A.; SIMPSON, R. C.; KOREN, Y.; BORENSTEIN, J. The navchair assistive wheelchair navigation system. *Rehabilitation Engeneering, IEEE Transactions*, v. 7, p. 443–451, 1999. Citado na página 15.

LOPES, A. C.; PIRES, G.; VAZ, L.; NUNES, U. Wheelchair navigation assisted by human-machine shared-control and a p300-based brain computer interface. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, p. 2438–2444, 2011. Citado na página 14.

LUCAS, B. D. *Generalized Image Matching by the Method of Differences*. Dissertação (Mestrado) — Carnegie-Mellon niversity, Brasília, July 1984. Citado na página 28.

LUCAS, B. D.; KANADE, T. An iterative image registration technique with an application to stereo vision. *Proceedings of Imaging Understanding Workshop*, p. 121–130, 1981. Citado na página 28.

MANOGNA, S.; VAISHNAVI, S.; GEETHANJALI, B. Head movement based assist system for physically challenged. *2010 4th International Conference on Bioinformatics and Biomedical Engineering (iCBBE)*, p. 1–4, 2010. Citado na página 15.

MARTINS, B.; VALGôDE, E. *Interface Multimédia para Cadeiras de Rodas Inteligente.* Dissertação (Trabalho de Conclusão de Curso) — Faculdade de Engenharia da Universidade do Porto, 2006. Citado na página 16.

MARTINS, G. de A.; DOMINGUES, O. *Estatística Geral e Aplicada*. 4. ed. [S.l.]: Atlas, 2012. Citado na página 54.

MATSUMOTO, Y.; INO, T.; OGASAWARA, T. Development of intelligent wheelchair system with face and gaze based interface. *IEEE International Workshop on Robot and Human Interactive Communication*, p. 262–267, 2001. Citado 2 vezes nas páginas 16 e 17.

MAZO, M. An integral system for assisted mobility. *IEEE Robotics & Automation Magazine*, p. 46–56, 2001. Citado 2 vezes nas páginas 15 e 16.

MCROBERTS, M. *Arduino Básico*. 1. ed. São Paulo: Novatec Editora Ltda., 2012. Citado 2 vezes nas páginas 35 e 36.

MOON, I.; LEE, M.; RYU, J.; MUN, M. Intelligent robotic wheelchair with emg-, gesture-, and voice-based interfaces. *International Conference on Intelligent Robots and Systems*, p. 3453–3458, 2003. Citado 2 vezes nas páginas 14 e 15.

OLIVEIRA, P. M. de; FLORES, F. C.; ALMEIDA, N. M. de. A command generation framework to support an electric wheelchair interface. 2015. Citado 2 vezes nas páginas 19 e 60.

OLIVEIRA, P. M. de; FLORES, F. C.; ALMEIDA, N. M. de. A real-time eyebrow segmentation and tracking technique to support an electric wheelchair interface. *A aparecer em Proceedings of 2015 IEEE Region 8 16th International Conference on Computer as a Tool (EuroCon 2015)*, 2015. Citado 2 vezes nas páginas 19 e 60.

ORTIGOZA, R. S.; MARCELINO-ARANDA, M.; ORTIGOZA, G. S.; GUZMAN, V. M. H.; MOLINA-VILCHIS, M. A.; SALDANA-GONZALEZ, G.; HERRERA-LOZADA, J. C.; OLGUIN-CARBAJAL, M. Wheeled mobile robots: a review. *Latin America Transactions, IEEE (Revista IEEE America Latina)*, IEEE, v. 10, n. 6, p. 2209–2217, 2012. Citado na página 34.

OTSU, N. A Threshold Selection Method from Gray-level Histograms. *IEEE Transactions on Systems, Man and Cybernetics*, v. 9, n. 1, p. 62–66, 1979. Citado 3 vezes nas páginas 23, 25 e 41.

PAJKANOVIC, A.; DOKIC, B. Wheelchair control by head motion. *Serbian Journal of Electrical Engineering*, v. 10, p. 135–151, 2013. Citado na página 15.

PANTIC, M.; MEMBER, S.; ROTHKRANTZ, L. J. M. Automatic analysis of facial expressions: The state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 22, p. 1424–1445, 2000. Citado na página 17.

PARANHOS, T. C.; KIYOKY, K.; FILHO, W. de B. V. Estudo sobre interfaces de comando de cadeira de rodas motorizadas para tetraplégicos. *IV Encontro de Ciência e Tecnologia da Faculdade UnB - Gama*, p. 240–243, 2012. Citado na página 15.

PERES, S. M.; FLORES, F. C.; VERONEZ, D.; OLGUíN, C. J. M. Libras signals recognition: a study with learning vector quantization and bit signature. *IEEE Ninth Brazilian Symposium on Neural Networks*, 2006. Citado na página 26.

PLASCENCIA, A. *Sensor Fusion for Autonomous Mobile Robot Navigation*. Tese (Doutorado) — Aalborg Universitet, 2008. Citado na página 48.

RASCANU, G. C.; SOLEA, R. Electric wheelchair control for people with locomotor disabiliteis using eye movements. *2011 15th International Conference on System Theory, Control, and Computing (ICSTCC)*, p. 1–5, 2011. Citado na página 16.

SCIAVICCO, L.; VILLANI, L. *Robotics: modelling, planning and control*. [S.l.]: Springer, 2009. Citado na página 34.

TOMARI, M. R. md; KOBAYASHI, Y.; KUNO, Y. Development of a smart wheelchair system for a user with severe motor impairment. *International Symposium on Robotics and Intelligent Sensors*, p. 538–546, 2012. Citado na página 15.

VIOLA, P. A.; JONES, M. J. Rapid object detection using a boosted cascade of simple features. In: . [S.l.]: IEEE Computer Society, 2001. p. 511–518. Citado 3 vezes nas páginas 19, 21 e 22.

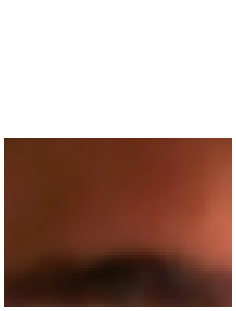
WEI, L.; HU, H. A multi-modal human machine interface for controlling an intelligent wheelchair using face movements. *IEEE International Conference on Robotics and Biometics*, p. 2850–2855, 2011. Citado na página 16.

YANCO, H. A. Wheelesley: A robotic wheelchair system: Indoor navigation and user interface. *Lecture Notes in Computer Science*, v. 7, p. 256–268, 1998. Citado 2 vezes nas páginas 15 e 16.

YU, H.-Y.; CHEN, J.-J.; KUO, C.-H. Toward constructing a user-environment interface for a smart light-mobile wheelchair. *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, p. 403–407, 2012. Citado na página 16.

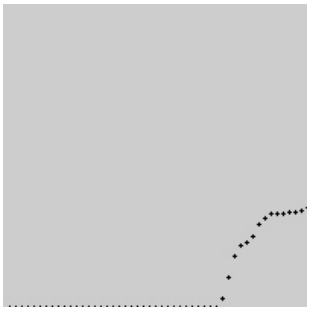
Apêndices

Apêndice A

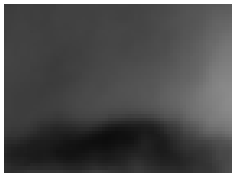


(a) Original

(b) Segmentação

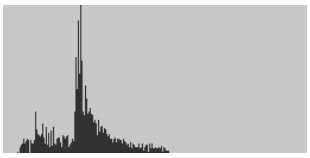


(c) Assinatura de Bits



(d) Banda L

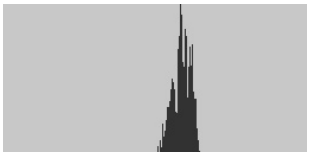
(e) L binarizada



(f) Histograma L

(g) Banda a

(h) a binarizada



(i) Histograma a

(j) Banda b

(k) b binarizada

(l) Histograma b

Figura 32 – Amostra 1. (a) Recorte original; (b) Segmentação resultante; (c) Mapa de Assinatura de Bits; (d) Banda L; (e) Banda L binarizada; (f) Histograma da Banda L; (g) Banda a; (h) Banda a binarizada; (i) Histograma da Banda a; (j) Banda b; (k) Banda b binarizada; (l) Histograma da Banda b;

(a) Original (b) Segmentação (c) Assinatura de Bits

(d) Banda L (e) L binarizada (f) Histograma L

(g) Banda a (h) a binarizada (i) Histograma a

(j) Banda b (k) b binarizada (l) Histograma b

Figura 33 – Amostra 2. (a) Recorte original; (b) Segmentação resultante; (c) Mapa de Assinatura de Bits; (d) Banda L; (e) Banda L binarizada; (f) Histograma da Banda L; (g) Banda a; (h) Banda a binarizada; (i) Histograma da Banda a; (j) Banda b; (k) Banda b binarizada; (l) Histograma da Banda b;

(a) Original (b) Segmentação (c) Assinatura de Bits

(d) Banda L (e) L binarizada (f) Histograma L

(g) Banda a (h) a binarizada (i) Histograma a

(j) Banda b (k) b binarizada (l) Histograma b

Figura 34 – Amostra 3. (a) Recorte original; (b) Segmentação resultante; (c) Mapa de Assinatura de Bits; (d) Banda L; (e) Banda L binarizada; (f) Histograma da Banda L; (g) Banda a; (h) Banda a binarizada; (i) Histograma da Banda a; (j) Banda b; (k) Banda b binarizada; (l) Histograma da Banda b;

(a) Original (b) Segmentação (c) Assinatura de Bits

(d) Banda L (e) L binarizada (f) Histograma L

(g) Banda a (h) a binarizada (i) Histograma a

(j) Banda b (k) b binarizada (l) Histograma b

Figura 35 – Amostra 4. (a) Recorte original; (b) Segmentação resultante; (c) Mapa de Assinatura de Bits; (d) Banda L; (e) Banda L binarizada; (f) Histograma da Banda L; (g) Banda a; (h) Banda a binarizada; (i) Histograma da Banda a; (j) Banda b; (k) Banda b binarizada; (l) Histograma da Banda b;

(a) Original (b) Segmentação (c) Assinatura de Bits

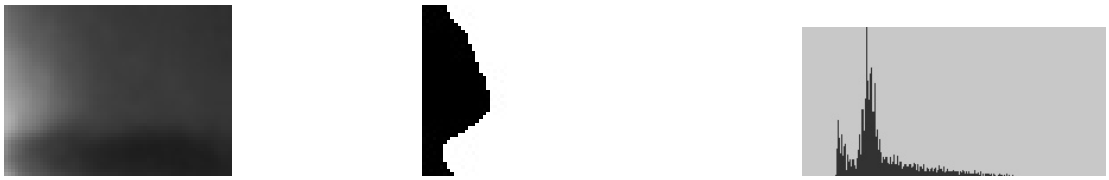
(d) Banda L (e) L binarizada (f) Histograma L

(g) Banda a (h) a binarizada (i) Histograma a

(j) Banda b (k) b binarizada (l) Histograma b

Figura 36 – Amostra 5. (a) Recorte original; (b) Segmentação resultante; (c) Mapa de Assinatura de Bits; (d) Banda L; (e) Banda L binarizada; (f) Histograma da Banda L; (g) Banda a; (h) Banda a binarizada; (i) Histograma da Banda a; (j) Banda b; (k) Banda b binarizada; (l) Histograma da Banda b;

(a) Original (b) Segmentação (c) Assinatura de Bits

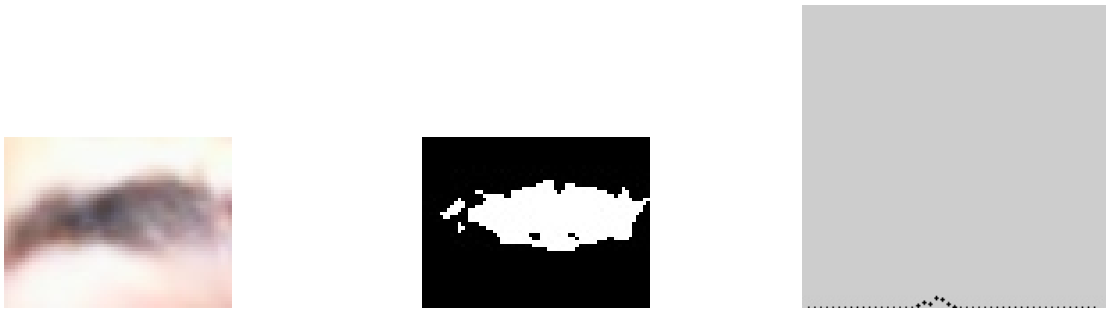


(d) Banda L (e) L binarizada (f) Histograma L

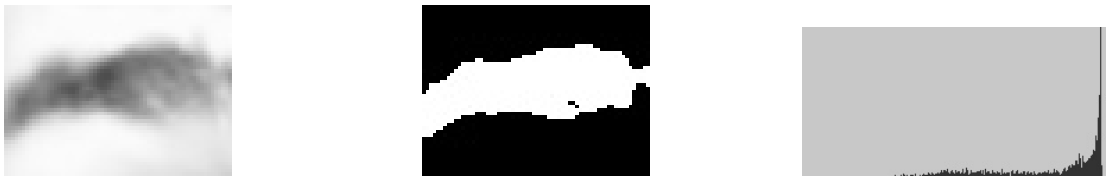
(g) Banda a (h) a binarizada (i) Histograma a

(j) Banda b (k) b binarizada (l) Histograma b

Figura 37 – Amostra 6. (a) Recorte original; (b) Segmentação resultante; (c) Mapa de Assinatura de Bits; (d) Banda L; (e) Banda L binarizada; (f) Histograma da Banda L; (g) Banda a; (h) Banda a binarizada; (i) Histograma da Banda a; (j) Banda b; (k) Banda b binarizada; (l) Histograma da Banda b;



(a) Original (b) Segmentação (c) Assinatura de Bits

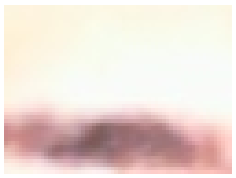


(d) Banda L (e) L binarizada (f) Histograma L

(g) Banda a (h) a binarizada (i) Histograma a

(j) Banda b (k) b binarizada (l) Histograma b

Figura 38 – Amostra 7. (a) Recorte original; (b) Segmentação resultante; (c) Mapa de Assinatura de Bits; (d) Banda L; (e) Banda L binarizada; (f) Histograma da Banda L; (g) Banda a; (h) Banda a binarizada; (i) Histograma da Banda a; (j) Banda b; (k) Banda b binarizada; (l) Histograma da Banda b;



(a) Original

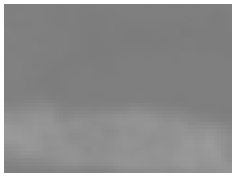
(b) Segmentação

(c) Assinatura de Bits

(d) Banda L

(e) L binarizada

(f) Histograma L



(g) Banda a

(h) a binarizada

(i) Histograma a

(j) Banda b

(k) b binarizada

(l) Histograma b

Figura 39 – Amostra 8. (a) Recorte original; (b) Segmentação resultante; (c) Mapa de Assinatura de Bits; (d) Banda L; (e) Banda L binarizada; (f) Histograma da Banda L; (g) Banda a; (h) Banda a binarizada; (i) Histograma da Banda a; (j) Banda b; (k) Banda b binarizada; (l) Histograma da Banda b;

(a) Original (b) Segmentação (c) Assinatura de Bits

(d) Banda L (e) L binarizada (f) Histograma L

(g) Banda a (h) a binarizada (i) Histograma a

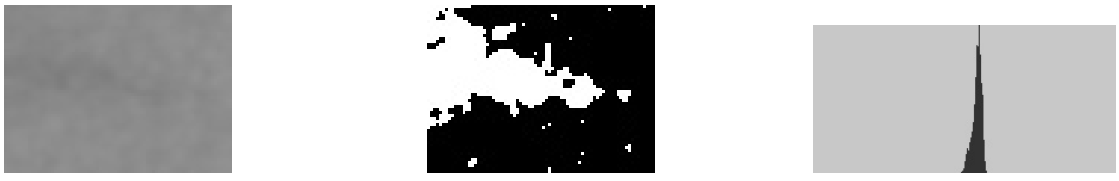
(j) Banda b (k) b binarizada (l) Histograma b

Figura 40 – Amostra 9. (a) Recorte original; (b) Segmentação resultante; (c) Mapa de Assinatura de Bits; (d) Banda L; (e) Banda L binarizada; (f) Histograma da Banda L; (g) Banda a; (h) Banda a binarizada; (i) Histograma da Banda a; (j) Banda b; (k) Banda b binarizada; (l) Histograma da Banda b;

(a) Original (b) Segmentação (c) Assinatura de Bits

(d) Banda L (e) L binarizada (f) Histograma L

(g) Banda a (h) a binarizada (i) Histograma a



(j) Banda b (k) b binarizada (l) Histograma b

Figura 41 – Amostra 4. (a) Recorte original; (b) Segmentação resultante; (c) Mapa de Assinatura de Bits; (d) Banda L; (e) Banda L binarizada; (f) Histograma da Banda L; (g) Banda a; (h) Banda a binarizada; (i) Histograma da Banda a; (j) Banda b; (k) Banda b binarizada; (l) Histograma da Banda b;