Centro Federal de Educação Tecnológica de Minas Gerais
Campus Divinópolis
Graduação em Engenharia Mecatrônica

Matheus Lara Pereira

Projeto e construção de um sistema embarcado para monitoramento de processos industriais utilizando visão computacional

Divinópolis
2017

Matheus Lara Pereira

Projeto e construção de um sistema embarcado para monitoramento de processos industriais utilizando visão computacional

Relatório final de Trabalho de Conclusão de Curso apresentado à comissão avaliadora do curso de Graduação em Engenharia Mecatrônica como parte dos requisitos exigidos para a obtenção da aprovação na disciplina de TCC II.
Áreas de integração: Computação, Eletrônica.

Orientador: Prof. Me. Jean Carlos Pereira

Divinópolis
2017

Centro Federal de Educação Tecnológica de Minas Gerais
CEFET-MG / Campus Divinópolis
Curso de Engenharia Mecatrônica

Monografia intitulada "*Projeto e construção de um sistema embarcado para monitoramento de processos industriais utilizando visão computacional"*, de autoria do graduando Matheus Lara Pereira, aprovada pela banca examinadora constituída pelos seguintes professores:

_____

Prof. M. Sc. Jean Carlos Pereira - CEFET-MG / Campus Divinópolis - Orientador

_____

Prof. M. Sc. Fernando Thomé de Azevedo Silva/ Campus Divinópolis

_____

Prof. M. Sc. Josias Gomes Ribeiro Filho - CEFET-MG / Campus Divinópolis

_____

Prof. Dr. Lúcio Santos Patrício
Coordenador do Curso de Engenharia Mecatrônica
CEFET-MG / Campus Divinópolis

Agradecimentos

Agradeço,
aos meus pais, Railda e José Pereira, e aos meus irmãos, Daniela e Tiago, que sempre se esforçaram e me apoiaram para que eu tivesse um estudo de qualidade. Ao meu orientador pelo suporte, correções e incentivos. Aos meus amigos, que estiveram presentes ao longo desses cinco anos muito felizes e produtivos. A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado. Por fim, a minha instituição e ao corpo docente por ter me dado à chance e todas as ferramentas que permitiram chegar hoje ao final desse ciclo de maneira satisfatória.

Resumo

O monitoramento de variáveis de processo é fundamental para gerar melhores resultados referentes a rentabilidade, desempenho, segurança e desenvolvimento em uma unidade de produção. A necessidade da obtenção, conversão e registro dessas variáveis é primordial para acompanhamento eficaz do processo. Em diversas empresas, essa etapa é relativamente lenta, visto que demanda uma inspeção presencial e periódica. Em paralelo, existem dispositivos de verificação visual que podem, juntamente com a visão computacional, dispensar o indivíduo da tarefa presencial, permitindo sua realização remotamente. A utilização de ferramentas de inspeção visual proporciona uma otimização na detecção dos fins desejados, um inspecionamento ininterrupto e uma maior segurança. Portanto o monitoramento inteligente é capaz de captar e processar variáveis desejadas de modo a retornar informações condizentes e importante para o processo. Estes fatores justificam a necessidade da realização de um projeto com as características deste trabalho, que consiste no projeto e desenvolvimento de um sistema embarcado para monitoramento de variáveis de processos, utilizando a ferramenta visão computacional. As atividades realizadas foram para a realização do projeto são de grande valia, visto que os resultados apresentados puderam retornar informações da utilização dos conceitos de identificação de perfis para monitoramento, processamento de imagem, aplicação de filtros, *webserver*, assim como conhecimento das interfaces de programação. Podendo assim, ser possível afirmar que as variáveis de um monitoramento visual contêm muitas informações que outros meios de realizar esta ação não possuem. O presente documento apresenta o projeto e desenvolvimento do dispositivo embarcado capaz de monitorar pessoas em ambientes específicos, sendo possível realizar a verificação dos dados através de um dispositivo móvel.

Palavras-chave: Monitoramento, Visão Computacional, Acessibilidade, Segurança, IOT.

Abstract

The monitoring of process variables is fundamental to generate better results regarding profitability, performance, safety and development in a production unit. The need of obtainment, conversion and record of these variables is primordial for effective process monitoring. In several companies, this stage is relatively slow, since it demands an on-site and periodic inspection. In parallel, there are visual verification devices that can, along with the computer vision, dismiss the individual from the face-to-face task, allowing them to be performed remotely. The use of visual inspection tools provides an optimization in the detection of the desired ends, an uninterrupted inspection and greater security. Therefore intelligent monitoring is able to capture and process desired variables in order to return information that is relevant and important to the process. These factors justify the need to carry out a project with the characteristics of this work, which consists of the design and development of an embedded system for monitoring process variables, using the computer vision tool. The activities for the realization of the project were of great value, since the presented results could return information of the use of the concepts of profile identification for monitoring, image processing, application of filters, webserver, as well as knowledge of the programming interfaces. Thus, it can be stated that the variables of a visual monitoring contain a lot of information that other means of performing this action do not have. This document presents the design and development of the embedded device capable of monitoring people in specific environments, and it is possible to perform the verification of the data through a mobile device.

Key-words: Monitoring, Computer Vision, Accessibility, Security, IOT.

Sumário

Lista de Figuras **ix**

Lista de Tabelas **x**

Lista de Acrônimos e Notação **xi**

1 Introdução **1**
1.1 Definição do Problema . . . 2
1.2 Motivação . . . 3
1.3 Objetivos do Trabalho . . . 3
1.4 Metodologia . . . 4
1.5 Estado da Arte . . . 6
1.6 Organização do Documento . . . 7

2 Fundamentos **9**
2.1 Revisão de Literatura . . . 9
2.2 Processamento de imagem . . . 12
2.2.1 Imagem digital . . . 14
2.2.2 Resolução . . . 19
2.2.3 Ruídos . . . 21
2.3 Visão Computacional . . . 22
2.3.1 OpenCV . . . 22
2.3.2 Segmentação . . . 23
2.3.3 Reconhecimento de Padrões . . . 24
2.3.4 Filtros . . . 26
2.3.5 Desfocagem . . . 27
2.4 Luz e Cor . . . 27
2.4.1 Sistema Visual Humano . . . 30
2.4.2 Espaço de cor . . . 30
2.4.3 Histograma de cores . . . 32
2.5 Sistemas embarcados . . . 33
2.5.1 Raspberry pi . . . 33
2.6 Comunicação Wireless . . . 34
2.6.1 Bluetooth . . . 34
2.6.2 Wi-fi . . . 35

3 Desenvolvimento **36**
3.1 Projeto Conceitual . . . 36
3.1.1 Análises do dispositivo embarcado . . . 36
3.1.2 Interação com a interface de programação no Sistema Operacional *Windows* 10 . . . 38
3.2 Métodos de identificação de perfis . . . 39
3.3 Interface do usuário . . . 40
3.3.1 Criação do Aplicativo . . . 40
3.3.2 Webpage . . . 41
3.3.3 Servidor *Web* Dinâmico . . . 42
3.4 Transição do algoritmo para o sistema embarcado . . . 45
3.4.1 Capacidade computacional . . . 46
3.4.2 Disposição das camadas algorítmicas para identificação do perfil . . . 46
3.5 Projeto Estrutural, Materiais e Montagem . . . 48
3.6 Análise do protótipo viável . . . 49
3.7 Orçamento . . . 51

4 Resultados parciais e finais **52**
4.0.1 Aplicação de técnicas de visão computacional . . . 52
4.1 Interface do Usuário . . . 55
4.2 Análise computacional dos algoritmos na plataforma embarcada . . . 56
4.3 Acoplamento Câmera . . . 59
4.4 Protótipo viável . . . 64

5 Considerações Finais **67**
5.1 Conclusões . . . 67
5.2 Propostas de continuidade . . . 68

A Códigos **70**
A.1 Instalação da biblioteca OpenCV no sistema operacional Windows . . . 70
A.2 Método Skin detection . . . 72
A.3 Código de contabilização de círculos . . . 75

Referências **76**

Lista de Figuras

1.1 Fluxograma para projeto e desenvolvimento do sistema 4
1.2 Camadas algorítmicas para identificação do perfil. 6

2.1 Modelo do primeiro sistema embarcado, utilizado pela NASA para a viagem do homem a Lua 10
2.2 Aplicação de filtros em imagens realizada por Clerck Maxwell 10
2.3 Resultado do experimento de Clerck Maxwell 10
2.4 Primeiro microprocessador criado pela INTEL 11
2.5 Tópicos do processamento digital 14
2.6 Dados de uma imagem raster 15
2.7 Representação de uma imagem digital raster de um olho humano 15
2.8 Representação de uma imagem em escala de cinza e sua respectiva binarização em níveis de limiares 17
2.9 Representação de imagem colorida, escala de cinza e binária 18
2.10 Demonstração da ocorrência de falseamento de sinal 18
2.11 Demonstração da ocorrência de falseamento de sinal 19
2.12 Imagens com o mesmo tamanho e diferentes resoluções 20
2.13 Imagens com a mesma resolução e tamanhos diferentes 20
2.14 Vizinhança de um Grid regular 20
2.15 Aplicação de ruído gaussiano 21
2.16 Efeitos do ruído impulsivo em imagem colorida e em escala de cinza 22
2.17 Representação de uma máscara genérica com o posicionamento dos valores utilizados nas equações de Prewitt e Sobel 24
2.18 Representação da segmentação por corte 25
2.19 Representação da atuação do método Background Subtraction 26
2.20 Espectro eletromagnético 27
2.21 Experimento realizado por Isaac Newton para comprovar sua Teoria do prisma 28
2.22 Decomposição do espectro de cores 28
2.23 Processo aditivo de cores primárias. 29
2.24 Representação cubo do modelo RGB 31
2.25 Representação de alguns modelos de cores e seus respectivos parâmetros RGB 31
2.26 Representação de uma imagem digital 32
2.27 Histograma dos canais RGB 33
2.28 Elementos básicos de um sistema embarcado 34

2.29 Ilustração de um Raspberry Pi 3B 34

3.1 Ilustração do módulo de câmera acoplado ao Raspberry Pi 3B 37
3.2 Arquitetura de rede. 43
3.3 Registro do servidor ao ser acessado por um hospedeiro 44
3.4 Registro do servidor ao ser acessado por dois hospedeiros 44
3.5 Ambiente de operação acessado remotamente. 45
3.6 Representação do suporte de acoplamento da câmera. 48
3.7 Representação do suporte para o Raspberry. 49
3.8 Representação do Protótipo viável. 50

4.1 Resultado do teste de contabilização de círculos. 53
4.2 Resultado do efeito do filtro blur em uma imagem. 53
4.3 Resultado da aplicação do método skin detection. 54
4.4 Resultado do design do ícone do aplicativo de utilização do usuário. 55
4.5 Resultado do ícone na tela do dispositivo móvel. 56
4.6 Página inicial do aplicativo. 57
4.7 Página que contém informações do aplicativo. 58
4.8 Página de acesso ao monitoramento. 59
4.9 Resposta do servidor ao aplicativo na ausência da *webpage*. 60
4.10 Processamento do sistema embarcado sem nenhum atividade. 60
4.11 Análise computacional dos métodos de Subtração de cena e contorno de área. 61
4.12 Inserção de ruído no ambiente para análise da estrutura condicional. 61
4.13 Processamento do sistema embarcado sem nenhum atividade. 62
4.14 Processamento do sistema embarcado sem nenhum atividade. 62
4.15 Processamento do sistema embarcado utilizando a requisição frequente de dados do servidor. 63
4.16 Resultado do protótipo de acoplamento da câmera 63
4.17 Resultado da instalação do sistema embarcado 64
4.18 Compartilhamento de rede, *smartphone* e sistema embarcado 64
4.19 Resultado do teste realizado para análise da comunicação entre o sistema embarcado e o aplicativo. 65
4.20 Resultado do teste com a variável da condição entre as camadas dois e três. 65

A.1 Janela para preenchimento das especificações do sistema. 71
A.2 Parâmetros para escolha das extensões a serem realizadas à biblioteca. . . . 71
A.3 Configuração para gerar os complementos da biblioteca OpenCv. 72

Lista de Tabelas

2.1 Representação da quantidade de bytes para uma imagem NxN com 2^n tons de cinza 13
2.2 Comparação de parâmetros das comunicações Wireless 35

3.1 Configurações relevantes do Raspberry Pi 3B 37
3.2 Características da câmera. 38
3.3 Orçamento estimado do projeto. 51

Lista de Acrônimos e Notação

EMBRAPA	Empresa Brasileira de Pesquisa Agropecuária
ISES	Instituto de Socioeconomia Solidária
IBMEC	Instituto Brasileiro de Mercado de Capitais
NASA	National Aeronautics and Space Administration
AGC	Apollo Guidance Computer
Ph.D	Philosophiae Doctor
MIT	Massachusetts Institute of Technology
INTEL	Integrated Eletronics
CI	Circuito Integrado
PA	Ponto de Acesso
SDK	Software Development Kit
IDE	Integrated Development Environment
ISM	Industrial, Scientific e Medical
WPAN	Wireless Personal Area Network

Capítulo 1

Introdução

Atualmente, há uma infinidade de aplicações de sistemas embarcados para os quais há diversos conceitos, mas apenas uma finalidade: realizar um conjunto de tarefas pré-definidas. BARR (1999) define um sistema embarcado (do inglês, *embedded system*, ou ainda, sistema embutido) como uma combinação de *software* e *hardware* projetada para desempenhar uma tarefa específica. BERGER (2002) aborda o conceito implementando que sistemas embarcados são dedicados a tarefas específicas, enquanto *desktops* são considerados plataformas de computação genérica. Diante disso, um sistema embarcado deve ser capaz de realizar determinada tarefa a ele empregada.

Perante as diversas aplicabilidades destes dispositivos, pode destacar a capacidade de serem acoplados a maiores sistemas, com a finalidade de monitoramento e obtenção de informações. Esta característica é de grande relevância, pois, obter informações se torna importante, a medida do quão grave a mudança de estado pode afetar o ambiente ao redor. Nessa situação, a capacidade de ouvir, visualizar, sentir ou analisar as variáveis de um local são de essencial magnitude para obter sinais de que algo não está normal. Sendo assim, é importante conhecer os valores das variáveis físicas do ambiente a ser monitorado. Para tal, utilizam-se sensores, que são dispositivos sensíveis a alguma forma de energia do ambiente que pode ser - luminosa, térmica, cinética -, relacionando informações sobre uma grandeza que precisa ser medida, THOMAZINI; ALBUQUERQUE (2005).

Uma forma de monitoramento eficaz é a utilização de câmeras para verificar visualmente o processo. Estas, juntamente com técnicas de visão computacional, podem ampliar a gama de variáveis a serem monitoradas, como: detecção de bordas; reconhecimento de objetos; visão estereoscópica ; detecção de movimento; entre outras. A aplicação desta técnica de monitoramento na indústria é revelante, pois, de acordo com BRADSKI; KAEHLER (2008), a visão computacional é a transformação de dados, de uma câmera fixa ou de vídeo, em uma decisão ou em uma nova representação. Todas essas transformações são feitas para alcançar algum objetivo particular, ou seja, tornar o dispositivo capaz de

ganhar alguma compreensão de mundo. Nota-se que um problema em dispositivos embarcados com captação visual é a transmissão das imagens, isto torna mais complexo o envio do sinal, pois possui maior número de dados a serem transmitidos e isso pode acarretar em um atraso na recepção do sinal. Outro fator que se torna uma problemática destes sistemas é a possibilidade de perda de dados, portanto, pode haver perda da imagem e o tratamento pode ser lesado. Contudo, estes problemas podem ser sanados fazendo o tratamento das imagens no próprio sistema embarcado, não havendo a necessidade de envio das imagens, e sim, o envio de sinais de comando para o dispositivo móvel.

As indústrias contemporâneas, chamadas indústrias 4.0, são empresas que envolvem inovações tecnológicas dos campos da automação. A alteração destas foi uma maneira de obter uma melhor visibilidade e visão sobre as operações da empresa. Uma grande mudança que acontecerá no mercado causada pela indústria 4.0 será a confiabilidade da produção e interação máquina-máquina, de forma que a tecnologia deverá se desenvolver continuamente. De acordo com GILCHRIST (2016), a Indústria 4.0 e o IIOT (*Internet Industrial Of Things*) são frequentemente apresentados a um nível elevado por consultores que estão apresentando de uma perspectiva de negócio aos clientes executivos.

Devido aos fatos apresentados, verifica-se uma viabilidade do desenvolvimento do projeto. Isso se justifica pela demanda por rápido acesso a informações nas indústrias contemporâneas e um monitoramento eficaz, tornando capaz a aquisição e tratamento de dados de forma praticamente instantânea, bem como possibilitando a tomada de decisões em tempo real. Pode-se também destacar que o projeto contem um sistema com tratamento da imagem na plataforma embarcada, ou seja, o vídeo não será enviado, e sim, tratado no sistema embarcado e o mesmo enviará sinais via *wireless* para o dispositivo móvel. Por fim, o projeto é uma possibilidade de rastreamento e monitoramento remoto de todos os processos por meio dos sistemas embarcados espalhados ao longo da planta.

1.1 Definição do Problema

O monitoramento visual em diversos ambientes passa por delimitações, tanto na identificação dos parâmetros a serem monitorados quanto no retorno de dados consistentes. Nos deparamos frequentemente com noticiários referentes a acidentes domésticos envolvendo crianças e piscina de uma residencia, a situação de desatenção em que a criança está próxima de uma área de risco e os pais não estão por perto e por descuido pode acontecer algum acidente. Outra situação envolvendo o mesmo contexto só que no âmbito industrial é a presença de um operador em uma região de perigo por algum descuido ou falta de informação.

Acidentes desta natureza acontecem pois os dispositivos de monitoramento acessíveis

(em relação a preço) não realizam o reconhecimento de formas, apenas fazem com que os dados obtidos pela câmera sejam gravados sem nenhum retorno ao usuário. Os dispositivos de baixo custo que retornam para os usuários quando há uma movimentação nas proximidades da região de risco, muitas vezes retornam informações falso-positivas, que ocorrem quando há a movimentação de objetos, pessoas ou animais em que o usuário do dispositivo não precisa ser notificado.

1.2 Motivação

A motivação para a realização deste trabalho é a falta de dispositivos de baixo custo que o meio social e industrial necessitam. A utilização de dispositivos embarcados para monitoramento, utilizando-se da visão computacional, pode sanar uma imensa quantidade de problemas na indústria, de forma muito mais acessível do que o monitoramento utilizado em algumas linhas de produção.

Já em meio social, o dispositivo pode agilizar tarefas diárias quanto evitar acidentes domésticos. Sendo assim, há uma grande demanda de mercado para que o trabalho seja realizado.

1.3 Objetivos do Trabalho

São objetivos deste trabalho:

Objetivo geral

Desenvolver um sistema de monitoramento utilizando a ferramenta visão computacional, a fim de identificar e computar a quantidade de pessoas em um ambiente, realizando o levantamento de dados à uma interface presente em um dispositivo móvel.

Objetivos específicos

Para obtenção dos resultados esperados, deve-se realizar os seguintes objetivos específicos:

- desenvolver o algorítimo de identificação visual de objetos;
- aplicação de filtros para captação da forma desejada;
- definição e utilização de um protocolo de comunicação industrial sem fio;
- desenvolver a interface gráfica em um dispositivo móvel para o levantamento de dados ao usuário;

- efetuar a comunicação entre o dispositivo de sensoreamento e a interface gráfica presente no dispositivo móvel;
- construir uma estrutura mecânica para instalação do sistema embarcado;
- instalação do dispositivo embarcado a fim de simular um ambiente industrial;
- desenvolver o sistema com um baixo custo.

1.4 Metodologia

PEREIRA (2016) ressalta que o desenvolvimento de um projeto mecatrônico deve ser dividido em seis fases. Duas dessas foram de fundamental importância para o desenvolvimento deste trabalho. De acordo com o autor, a primeira fase é o estudo do projeto. Esta busca obter conhecimento para o desenvolvimento do mesmo, como: Condições de operação; Características do sistema físico envolvido e; Especificações do projeto. A segunda fase consiste no projeto conceitual, esta, permite realizar um esboço dentro de uma arquitetura ideal de processo, analisando as ideias de conceitos do projeto, além de sua complexidade. À vista disso, criou-se um fluxograma, figura 1.1, para organização das três principais partes: Projeto, Processamento de imagem e Visão computacional.

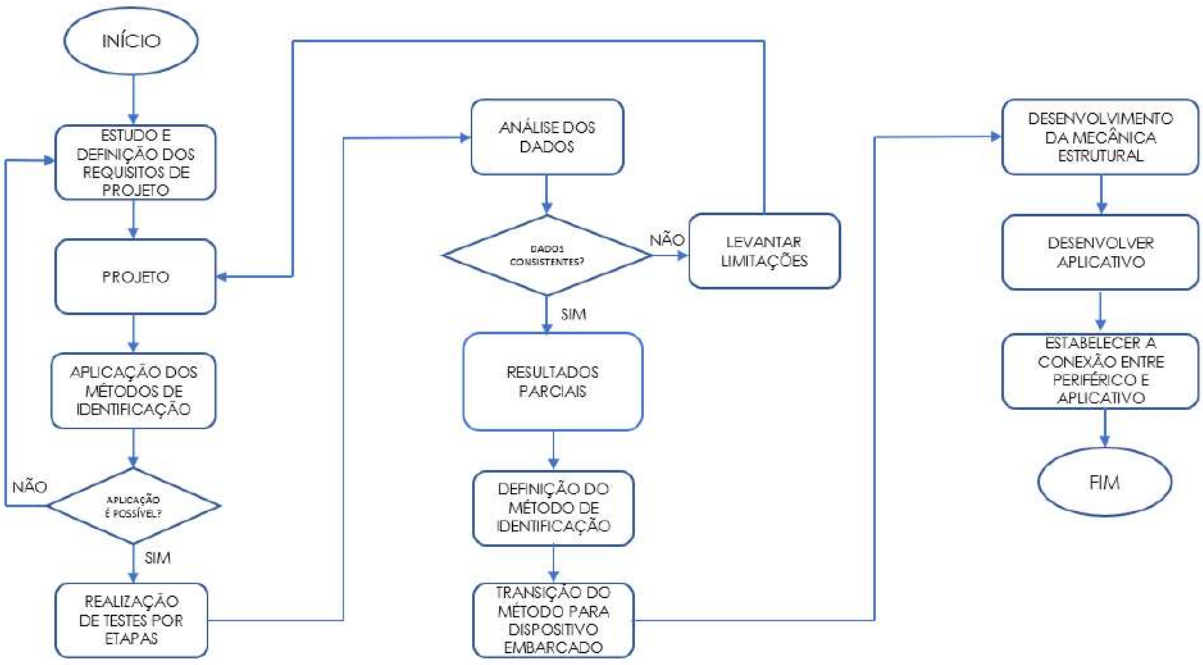


Figura 1.1: Fluxograma para projeto e desenvolvimento do sistema

Portanto para o desenvolvimento do trabalho proposto, foram levantadas as condições de operação do sistema como: posicionamento da câmera, condições de luminosidade do

ambiente, perfis que serão monitorados, angulação e área da região de interesse para o processamento, análise de possíveis ruídos que poderão interferir na captação e processamento da imagem. Estes parâmetros foram de fundamental importância para o desenvolvimento do projeto físico e determinação dos materiais que serão utilizados, assim como, a definição do posicionamento do dispositivo embarcado, escolha do modelo da câmera e do microcontrolador. O levantamento destes requisitos funcionais foi de fundamental importância para definir o quão bem o sistema desenvolve suas funções, ou seja, seu requisito de desempenho, que deve ser capaz de realizar o monitoramento de perfis através do processamento de vídeo, realizando-se assim a identificação de pessoas em um determinado ambiente com a ausência da variância luminosa e um posicionamento físico. Realizada estas tarefas pôde-se cumprir as etapas de 'Estudo e definição dos requisitos de projeto' e 'projeto'.

Dando sequência, conforme pode ser analisado nos resultados, realizou-se algumas amostragens utilizando diferentes métodos para inicializar o processo de identificação de objetos, abordando diversos conceitos. Estas, reconhecidas, analisadas e processadas através de algoritmos de identificação de perfis. O dispositivo de captação de imagens foi posicionado em diversos ambientes para analisar as variáveis influentes no processo de identificação, como: Iluminação; Angulação; Posicionamento, a fim de analisar o quão influentes estas variáveis podem alterar os resultados desejados. A realização destas atividades foi de fundamental importância para a coleta dos dados de diferenciação de pessoas e objetos, sendo possível realizar a contagem em um ambiente. Ao final destas atividades, pôde-se obter os resultados parciais presentes no fluxograma, figura 1.1, transpassando pelas etapas de 'Aplicação dos métodos de identificação', 'Realização dos testes por etapas' e 'Análise dos dados'.

Com a disposição dos dados coletados após os testes dos métodos e funções, criou-se uma estruturação algorítmica subdividida em camadas, conforme a figura 1.2.

Portanto, foi determinado o método de identificação utilizando-se três funções da visão computacional, dispostas em camadas, o algoritmo foi transcrito para o sistema embarcado. As camadas de algoritmo, juntamente com a transição da mesma para a plataforma embutida, estão descritas detalhadamente na seção de desenvolvimento.

Na etapa de projeto, definiu-se os periféricos a serem utilizados para a construção do sistema. Desta forma, pôde-se criar uma estrutura mecânica para o sistema embarcado. Antes da construção, realizou-se um dimensionamento necessário para dar confiabilidade ao sistema, logo isso, desenvolveu-se um protótipo estrutural e, em seguida, foi construído no processo de prototipagem rápida utilizando uma impressora 3D. Este é descrito na seção desenvolvimento.

Dando sequência ao fluxograma, realizou o desenvolvimento do aplicativo como ambi-

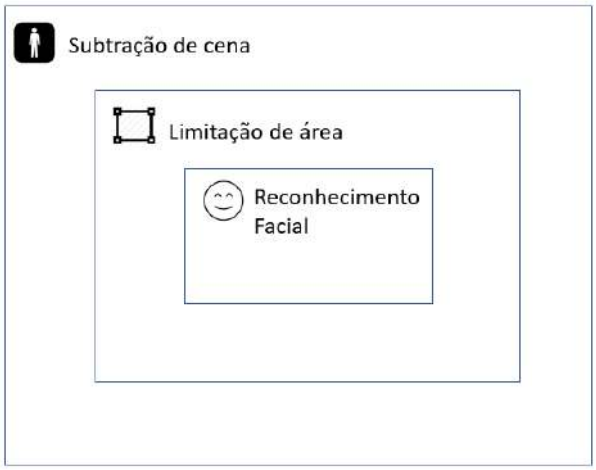


Figura 1.2: Camadas algorítmicas para identificação do perfil.

ente de operação do usuário, este, elaborado com a finalidade de tornar a navegação do usuário da melhor forma possível.

Ao fim da etapa de desenvolvimento do aplicativo, estabeleceu-se a conexão entre a plataforma embarcada e com o mesmo. Esta realizada através de um servidor *web* dinâmico que disponibiliza um ponto de acesso para o monitoramento da variável via interface. Tanto a criação do aplicativo quando sua conectividade com a plataforma embarcada pode ser analisada na seção desenvolvimento.

As atividades foram realizadas nos laboratórios de Protótipos e Automação do CEFET-MG, Campus V. O Laboratório de Protótipos utilizado para a construção da estrutura do projeto, assim como o Laboratório de Automação utilizado para estudos e utilização dos equipamentos para análise dos sinais presentes no trabalho.

1.5 Estado da Arte

Atualmente pode-se encontrar diversas pesquisas abordando aplicações da visão computacional na identificação de perfis. Estas, devido a grande expansão e a vasta gama de aplicações da visão computacional.

Dentre as pesquisas, pode-se destacar o projeto de AUQUIDA (2016), que desenvolveu um sistema de detecção de pessoas que estão com a ausência de capacetes de segurança em determinados ambientes, utilizando-se de visão artificial. O sistema possui a finalidade de detectar automaticamente pessoas que não estão utilizando o equipamento de segurança. O algoritmo é composto por uma série de etapas que vai da aquisição da imagem até segmentação da forma. O autor ressalta também que o algoritmo foi desenvolvido em um computador com processador dual core.

Outro sistema que pode ser destacado é o de captura e análise de movimentos utilizando-

se de visão computacional. PINHEIRO (2008), descreve que o sistema foi desenvolvido para análise das técnicas de medição do movimento humano e assim reconstruir a trajetória do movimento no espaço. O sistema foi desenvolvido para ser operado em um computador capaz de realizar o processamento das imagens capturadas.

Tomando em vista o ramo industrial, também há pesquisas e investimentos na área de visão computacional, no entanto, estas associadas a dispositivos embarcados. Segundo EMBRAPA (2016) as empresas Embrapa Instrumentação, Qualcomm Incorporated e o Instituto de Socioeconomia solidária (ISES) firmaram parceria para desenvolvimento tecnológico para drones com o objetivo de apoiar agricultores no Brasil. De acordo com a fonte "Os sistemas de bordo desenvolvidos para os drones - que combinarão a expertise da Embrapa em agricultura, algoritmos de processamento de imagem ao processador Qualcomm® Snapdragon e avançadas tecnologias móveis - têm como missão coletar, processar, analisar e transmitir informações das lavouras em tempo real para os agricultores e agentes ambientais ".

Segundo NYTIMES (2017), a empresa Intel investe 15 bilhões na compra da *Mobileye* cuja a tecnologia de visão digital ajuda a veículos autônomos a navegar com segurança pelas ruas. As expertises das empresas, Intel no desenvolvimento de circuitos integrados e *Mobileye* desenvolvimento de visão computacional e inteligência artificial para veículos autônomos, serão mescladas de forma a desenvolver um sistema embarcado com poder computacional tomando decisões corretas para a maior variedade de situações.

Observando as linhas de pesquisas e investimento, pode-se afirmar que os sistemas em desenvolvimento para monitoramento de perfis são realizados para serem utilizados em computadores e não em dispositivos embarcados. Nota-se também, que há pesquisas na identificação de padrões com o recurso da visão computacional, além de conter interesse de empresas do ramo tecnológico em investir em estudos e desenvolver produtos de monitoramento visual de forma embarcada.

Sendo assim, sistemas embarcados voltados para monitoramento de perfil também devem ser desenvolvidos para que seja possível fiscalizar os ambientes inacessíveis para processamento com computadores.

1.6 Organização do Documento

O presente documento está dividido em cinco capítulos, acrescidos de referências e anexos. O primeiro capítulo é referente à contextualização do projeto. Esta contextualização apresenta a definição do problema, objetivos, metodologia e uma breve abordagem a respeito das atuais pesquisas a respeito do tema.

O segundo capítulo contém os fundamentos teóricos para o desenvolvimento e com-

preensão do projeto. Os conceitos abordados são aqueles fundamentais para levantar um embasamento teórico e poder determinar as melhores técnicas e materiais para desenvolvimento do trabalho.

No terceiro capítulo está presente o desenvolvimento das atividades realizadas até o instante. O desenvolvimento realizado seguindo todas as etapas da metodologia também presente no documento. Para que o leitor possa compreender melhor as causalidades e justificativas do uso das técnicas e materiais.

O quarto capítulo contempla os resultados parciais obtidos na primeira etapa do projeto, juntamente com suas análises e discussões a respeito dos dados obtidos.

O quinto capítulo apresenta as considerações finais, estas a respeito das atividades e resultados obtidos. Neste capítulo também estão presentes propostas de continuidade e cronograma.

Capítulo 2

Fundamentos

Este capítulo apresenta todos os marcos históricos relevantes para a realização deste projeto assim como os conceitos fundamentais para aprofundamento da pesquisa sobre o mesmo.

2.1 Revisão de Literatura

Antes da década de 60 haviam estudos a respeito de sistemas embarcados e visão computacional. No entanto, foi nesta década que pesquisadores e engenheiros começaram a notar a enorme potencialidade destes dispositivos.

Em 1960 foi desenvolvido por Charles Stark Draper o primeiro grande sistema embarcado, o *Apollo Guidance Computer (AGC)*, Figura 2.1. O sistema era responsável por controlar todas as espaçonaves Apollo que levaram o Homem à Lua por diversas vezes nas décadas de 60 e 70. Este foi capaz de propiciar através de um sistema que ao invés de processadores utilizava-se portas NOR o poder computacional para a navegação espacial até a Lua. O AGC possuía uma interface composta por teclado numérico e *display* eletroluminescentes de forma a ser operado pelos astronautas, retornando dados da navegação e a execução de sua programação EMBARCADOS (2016).

Por outro lado, até a mesma década, a visão computacional não havia um embasamento expressivo. Os conceitos que eram conhecidos eram dos processos de fotografia, quando em 1861, Clerk Maxwell, capurou tês imagens em preto e branco e aplicou-se três modelos de filtros - vermelho, verde e azul - projetando cada uma dessas imagens, conforme figura 2.2. Em sequência, Maxwell realizou a mesclagem das respectivas imagens obtendo-se então uma figura colorida, Figura 2.3. Este processo foi tão revolucionário e eficiente que até hoje é utilizado, pois as cores vermelha, verde e azul são cores aditivas primárias.

Foi então que em 1960 a visão computacional começava a ser idealizada por Larry Roberts, Ph.D no *Massachusetts Institute of Technology (MIT)*, que escreveu sobre as possibilidades de extrair informações geométricas 3D de uma vista em perspectiva 2D de

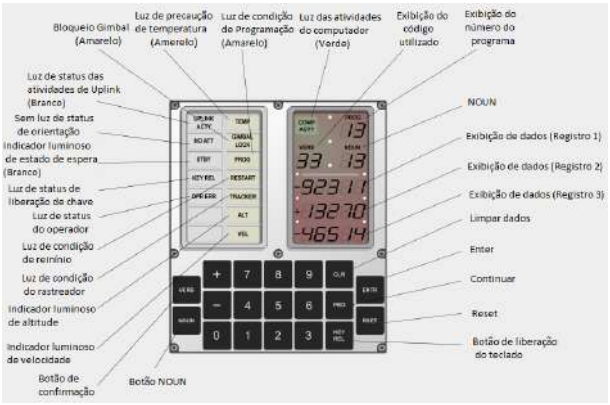


Figura 2.1: Modelo do primeiro sistema embarcado, utilizado pela NASA para a viagem do homem a Lua - Fonte: AGC (2008).

Figura 2.2: Aplicação de filtros em imagens realizada por Clerck Maxwell 1861.

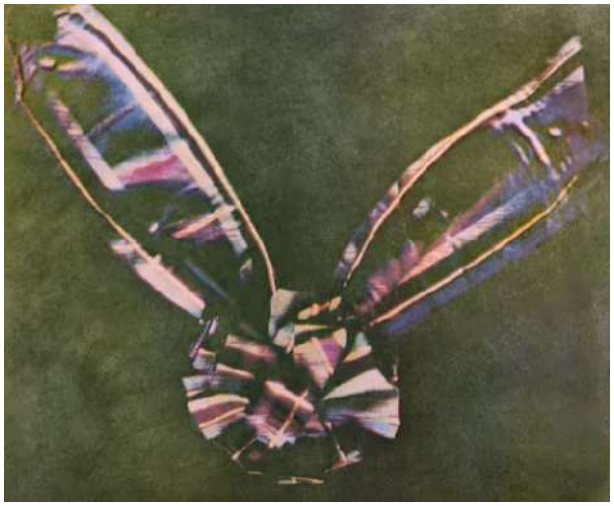


Figura 2.3: Junção das imagens com filtros para obtenção de uma imagem colorida realizado por Clerck Maxwell em 1861.

blocos. Assim foi notado as possíveis aplicabilidades de tais informações, pesquisadores do MIT e de outras instituições do mundo seguiram os estudos no contexto da visão computacional dos blocos HUANG (1996).

Enquanto a pesquisa perante os conceitos da visão computacional estavam em alta, em 1961 foi produzido o primeiro sistema embarcado em massa. O computador de orientação D-17 da *Autonetics* foi desenvolvido para o míssil Minuteman, foi feito a partir da lógica de transistor e tinha um disco rígido para a memória principal. Sete anos após, em 1968,

B. Noyce e G. Moore formaram a *Integrated Eletronics (Intel)*. Com o nascimento da *Intel*, começou o grande investimento em pesquisa para desenvolvimento dos chamados Circuitos Integrados (CI), que são potenciais tecnologias para o avanço dos dispositivos embarcados. E assim, um ano após a criação da *Intel*, em 1969, foi idealizado o primeiro sistema incorporado em um carro, a injeção de combustível do *Volkswagem* 1600 foi controlada por um microprocessador EMBARCADOS (2016).

Em 1971, a Intel lança o seu primeiro microprocessador, o 4004, uma unidade de processamento central de 4 bits. Este foi um grande marco pois foi o primeiro microprocessador em um chip simples, assim como o primeiro disponível comercial, Figura 2.4. Diante destes fatos, inicializou-se então o investimento para desenvolvimento de novos microprocessadores com melhor capacidade e espaço físico, por empresas como: Zilog, Motorola, MOS *Technology*, Texas *Instruments*, Rockwell, RCA, Data General EMBEDDED (2008).

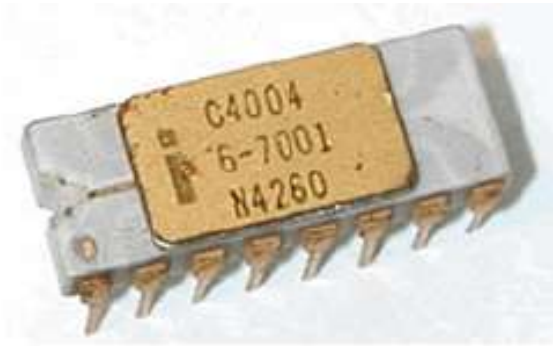


Figura 2.4: Primeiro microprocessador criado pela INTEL - Fonte: EMBEDDED (2008).

Durante a ascensão dos dispositivos embarcados nos anos de 1960, pesquisadores e engenheiros de diversas partes do mundo desenvolviam pesquisa referentes a visão computacional. Até que os pesquisadores perceberam que era necessário enfrentar imagens do mundo real, assim, muitas pesquisas eram necessárias nas chamadas tarefas de visão de 'baixo nível', como na detecção de segmentação. Algoritmos de processamento de imagens de baixo níveis eram aplicados em imagens 2D para obter o "esboço primário", a partir dos quais um esboço de 2,5D da cena é obtido usando o método *binocular stereo*.

Foi assim que em 1978, David Marr no *MIT*, tomou uma abordagem ascendente para a compreensão da cena. Marr apresentou algoritmos de processamento de imagem de alto nível para modelar as representações dos objetos na cena. Este marco foi um dos mais influentes na visão por computador, conhecido pelos pesquisadores como "Do paradigma criado por Marr, não podem expulsar-nos"HUANG (1996).

Na mesma década, Bryce Bayer, um cientista americano que trabalha para a *Kodak*, apresentou a captura de imagens de cores vivas para a fotografia digital, chamando a invenção de filtro *Bayer* PHASE (2016).

Em 1980, os microcontroladores e microprocessadores passaram por uma mudança em seus *designs*. O microprocessador passou por uma otimização em sua velocidade de processamento e no tamanho de sua memória, enquanto no microcontrolador houve uma melhora em sua potência e em seu tamanho físico. Vários componentes externos presentes nos dispositivos embarcados foram integrados em um microchip de forma a serem empregados à um processador, e assim houve a propalação dos dispositivos móveis EMBARCADOS (2016).

Na mesma proporção que haviam investimentos nos sistemas embarcados, em 1981 as câmeras inteligentes para aplicações industriais eram introduzidas, baseadas em *mouse óptico*, desenvolvido por Richard Lyons na empresa Xerox. Este foi o primeiro dispositivo de imagem e unidade de processamento embutida em um sistema compacto PHASE (2016).

À vista de toda a evolução de *hardware* para os dispositivos embarcados, em 1984 Fujio Masuoka inventa a memória *flash (NOR e NAND)* enquanto trabalhava na Toshiba. O desenvolvimento desta memória foi importante para armazenamento de informações, por se tratar de um a memória não volátil que pode ser eletricamente apagado e reprogramado. Sendo que os dispositivos embarcados executam atividades pré-programadas, este tipo de memória é fundamental EMBEDDED (2008).

Diante de todo o investimento e pesquisa nos sistemas embarcados, este termo começou a ser utilizado no Brasil em 2007, quando o mesmo foi definido em um site de busca e, assim, sistema embarcado passou a ser utilizado por engenheiros da área. Antes de serem definidos assim eram citados como eletrônicos de uma forma geral, não sendo possível diferenciar a complexidade e os poderes computacionais de um sistema embarcado dos eletrônicos sem a característica de computar.

Atualmente a junção dos dois sistemas se propaga em todos os meios. Essa proliferação do monitoramento inteligentes capaz de avaliar o meio que está incluso em seu espaço amostral retoma um conceito do início deste capítulo que ressalta que o investimento na pesquisa e desenvolvimento neste conjunto pode ser revolucionário para os sistemas tecnológicos.

2.2 Processamento de imagem

Conforme MARENGONI; STRINGHINI (2009) os recursos de visão computacional, em maioria, carecem de pré-processamento realizado pelo processamento de imagem. Para a realização da extração dos dados de uma determinada imagem, esta, em alguns casos, deve ser convertida para algum formato específico, tamanho correspondente ou aplicação de filtros para remoção de ruídos provenientes da aquisição da imagem.

Tabela 2.1: Representação da quantidade de bytes para uma imagem NxN com 2^n tons de cinza

n N	1	2	3	4	5	6	7	8
32	128	256	512	512	1024	1024	1024	1024
64	512	1024	2048	2048	4096	4096	4096	4096
128	2048	4096	8192	8192	16384	16384	16384	16384
256	8192	16384	32768	32768	65536	65536	65536	65536
512	32768	65536	131072	131072	262144	262144	262144	262144
1024	131072	262144	393216	524288	655360	786432	917504	1048576

De acordo com MARQUES FILHO; NETO (1999) para o sistema de processamento de imagens consiste nas seguintes etapas:

- Aquisição: É responsável por converter uma imagem em representação numéria para ser armazenada, neste processo dois elementos são fundamentais, o primeiro é um equipamento capaz de captar uma faixa de energia, que reproduz como sinal de saída o mais correspondente ao sinal elétrico detectado. O segundo é a conversão do sinal elétrico analógico em dado digital, na representação de bit 0 ou 1. O sinal analógico obtido é submetido a uma discretização espacial, chamado de amostragem, e uma discretização em amplitude, chamada de quantização, para estar na formatação necessária para o processamento computacional.

- Armazenamento: Consiste em conservar os dados obtidos pela aquisição. A armazenagem pode ser dividida em: Armazenamento de curta duração, é acessado quando a imagem é utilizada em várias etapas do processamento; Armazenamento de massa, utilizado para operações de imagens relativamente rápidas e; Arquivamento de imagem, usado quando há a necessidade de acesso a imagem futuramente. Parâmetros de processamento são fundamentais para determinar a unidade de armazenamento dos dados, pois quanto maiores as especificações do processo de digitalização deve-se determinar N,M e n adequados, do ponto de vista de qualidade de imagem e *byte* de armazenamento. A tabela 2.1, demonstra a estimativa do número de *bytes* necessários para armazenar uma imagem NxN com 2^n tons de cinza, calculados como: N x N x $n/8$.

- Processamento: Esta etapa compreende métodos expressos sob forma algorítmica realizada via *software*. Pode ser tratado fatores como: Domínio de espaço; Filtragem no domínio espacial; Filtros estáticos; Filtros Gaussianos; Filtro Passa-Alta; Domínio de frequência. A utilização de *hardware* para processamento de imagem é utilizado em situações nas quais há certas limitações do computador principal.

- Saída: Consiste em fornecer os dados de forma a serem transmitidos ou exibidos. A transmissão de imagens digitalizadas pode ser realizada utilizando-se de redes de computadores e protocolos de comunicação, uma questão relevante na transmissão de grande quantidade de *bytes* à uma longa distância torna-se um desafio devido a interferências e atraso nas informações. A exibição também é uma opção essencial de saída para o processamento de imagem.

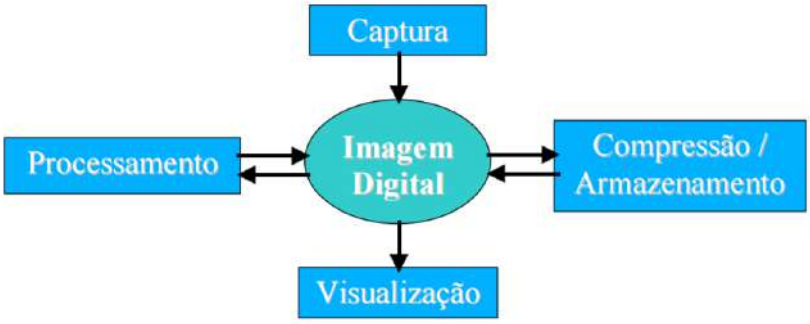


Figura 2.5: Tópicos do processamento digital - Fonte: SCURI (1999)

2.2.1 Imagem digital

Em concordância com MARQUES FILHO; NETO (1999), uma imagem digital consiste em uma função f(x,y) discretizada tanto em questão espacial quanto em amplitude. Sendo assim, imagens digitais são representadas como matrizes cuja a representação de um ponto na imagem corresponde à uma determinada linha e coluna da matriz, e a tonalidade correspondente à este ponto é intensificada pelo valor presente na posição da combinação de linha e coluna da matriz, Figura 2.7.

Para MATHER; KOCH (2011) imagens digitais são representadas logicamente como uma matriz de linhas e colunas, e acrescenta que os *arrays*[1] de dados de uma imagem estão incluídos na classe geral de 'dados *raster*' [2]. O autor explica que o valor de dados individuais não é explicitamente associado a uma determinada posição, a localização de cada valor de dados, denominado *pixel*, está implícito pela sua posição na matriz, Figura 2.6.

A Figura 2.6 demonstra o posicionamento dos *pixels* em uma imagem *raster*. A posição de cada célula de grade pode ser calculada se houver o conhecimento do espaçamento de *pixel* horizontal e o vertical.

A matriz de valores de *pixel* é mantida em uma área especial da memória do computador, nomeada 'memória gráfica'. Esta região está normalmente localizada na placa de

[1]Estrutura de dados que armazena uma coleção de elementos de tal forma que cada um possa ser identificado por, pelo menos, um índice ou uma chave.

[2]Matriz de pontos que representa geralmente uma grade retangular de pixel ou pontos de cor.

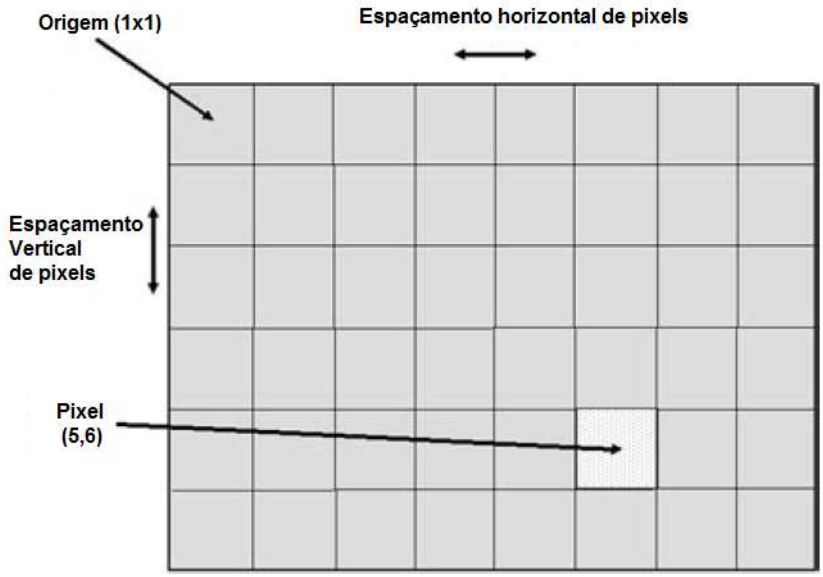
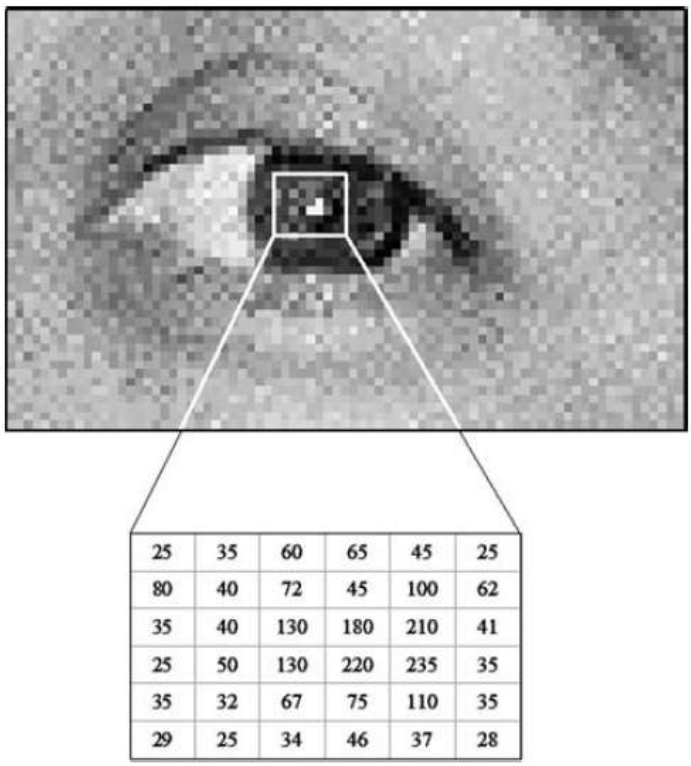


Figura 2.6: Dados de uma imagem raster com representação do sistema de coordenadas (linha,coluna) - Fonte:MATHER; KOCH (2011).



25	35	60	65	45	25
80	40	72	45	100	62
35	40	130	180	210	41
25	50	130	220	235	35
35	32	67	75	110	35
29	25	34	46	37	28

Figura 2.7: Representação de uma imagem digital raster de um olho humano, mostrando a correspondência entre os níveis de cinza dos pixels que compõem a imagem e a representação matricial de nível de cinza de pixel na memória gráfica do computador - Fonte:MATHER; KOCH (2011).

vídeo e não faz parte da memória de acesso aleatório (RAM) do computador. A Figura 2.7, por ser uma imagem em escala de cinza, apenas uma matriz de números é necessária

para armazenar os valores de pixel, cada um dos quais podem assumir valores de 256 níveis de brilho variando de 0 (preto) e 255 (branco) em que os valores próximos de 127 são meados de cinza.

Uma imagem digital em escala de cinza tem apenas uma componente para denominar a coloração do *pixel* correspondente, enquanto imagens digitais coloridas necessita de três componentes, sendo estes os níveis das cores primárias de luz (Vermelho, Verde e Azul) em cada posição de pixel MATHER; KOCH (2011).

Ainda conforme o autor, uma imagem colorida é produzida utilizando-se três matrizes de varredura, esta possuem valores de *pixel* que representam os níveis das três cores primárias da luz. Os níveis 0 a 255 representam o intervalo de cada cor primária de 0 (preto) a 255 (intensidade máxima de vermelho, verde ou azul). Desta maneira, diferentes combinações das R,G e B produzem as cores do espectro.

Imagem Binária

De acordo com DAWSON-HOWE (2014) uma imagem binária é aquela que há apenas um único bit por *pixel*, portanto é exibida apenas com as cores preto e branco, que são determinados por limiares de uma imagem em escala de cinza. As imagens binárias podem ser criadas a partir de muitos tipos de imagens, como imagens de intensidade, imagens de gradiente, imagens de diferença, dentre outras. Para a realização da transição de imagens em níveis de cinza para binária, deve-se determinar que cada célula da matriz corresponda ao valor de 0 ou 1 dependendo se valor da sua intensidade está acima ou abaixo do limiar determinado. Esta operação é frequentemente utilizada para alguns objetos que pretende descartar o fundo, mais especificamento os *pixels* do objeto são representados em 1 e o fundo em 0, conforme Figura 2.8.

A vantagem de se utilizar imagens binarizadas é a redução do custo computacional utilizado para realizar o processamento da imagem, sendo que cada ponto de intensidade possui apenas um *bit*.

Imagem em escala de cinza

As imagens em escala de cinza geralmente possuem 8 bits por *pixel*. O processamento em escala de cinza é mais fácil que o de imagens coloridas devido aos *pixels* correspondentes a imagem em escala de cinza possuírem apenas um parâmetro enquanto as imagens coloridas os *pixels* possuem 3 dimensões para determinação da intensidade de cada canal, tornando o processamento mais complexo, e portanto, possuindo maior custo computacional. Consequente, há muitos anos, a visão por computador foi baseada em imagens em nível de cinza, devidamente à essa premissa de que as imagens em níveis de cinza são menores e menos complexas sendo possível seu entendimento, DAWSON-HOWE (2014).

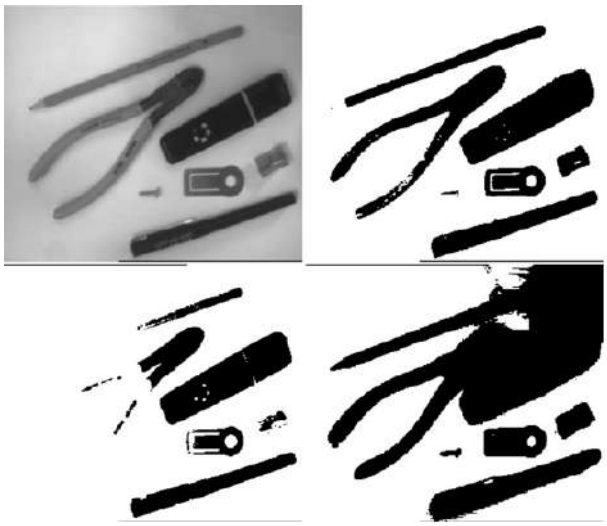


Figura 2.8: Representação de uma imagem em escala de cinza e sua respectiva binarização em níveis de limiares. A imagem superior, à direita, representa a taxa de limiar determinada corretamente. Inferior, à esquerda, representa a taxa de limiar baixa. Inferior, à direita, representa a taxa de limiar alta - Fonte: DAWSON-HOWE (2014).

KUEHNI (2001) descreve que imagens em escala de cinza possuem uma série de graus representando uma escala de cores acromática, com passos visualmente equidistantes entre as classe vizinhas, e complementa dizendo que uma imagem em escala de cinza em que os passos diferem em 10% em reflectância luminosa, não é uma escala perceptualmente uniforme.

Imagem Colorida

As imagens coloridas, também chamadas de multiespectrais, possuem diversos canais diferenciando-se das imagens em escala de cinza que são monocromáticas. As imagens com mais de um espectro, policromáticas, representam a luminância e a crominância dentro da cena. Estas informações podem ser representadas por diversas formas, mas todos requerem múltiplos canais de dados. Portanto, imagens coloridas necessitam de maior estrutura para armazenamento e são mais complexas para serem processadas do que as imagens binarizadas e imagens em escala de cinza. Esta maior dificuldade de processamento é devido possuírem três canais para serem tratados enquanto os outros modelos de imagem possuem apenas um. No entanto, a grande vantagem das imagens coloridas é que a cor fornece informações úteis que podem ajudar com muitos processos, conforme pode ser visto na Figura 2.9, DAWSON-HOWE (2014).

Amostragem

MARENGONI; STRINGHINI (2009) diz que imagens digitais são formadas através da amostragem de uma imagem contínua em elementos discretos. Este processo consiste

Figura 2.9: À esquerda, a representação detalhada de uma imagem colorida. Ao centro, a mesma imagem em escala de cinza. À direita, a imagem binarizada com limiar de intensidade igual a 40.

em converter a imagem analógica em uma matriz no formato M por N pontos, cada ponto caracterizado como um *pixel*.

$$f(x,y) = \begin{pmatrix} \text{f(0,0)} & \text{f(0,1)} & \text{...} & \text{f(0,N-1)} \\ \text{f(1,0)} & \text{f(1,1)} & \text{...} & \text{f(1,N-1)} \\ \text{:} & \text{:} & \text{.:.} & \text{:} \\ \text{f(M-1,0)} & \text{f(M-1,1)} & \text{...} & \text{f(M-1,N-1)} \end{pmatrix}$$

A taxa de amostragem é a quantidade de amostras do sinal contínuo no tempo essencial para obter uma digitalização contendo todos as informações necessárias da imagem, Figura 2.10.

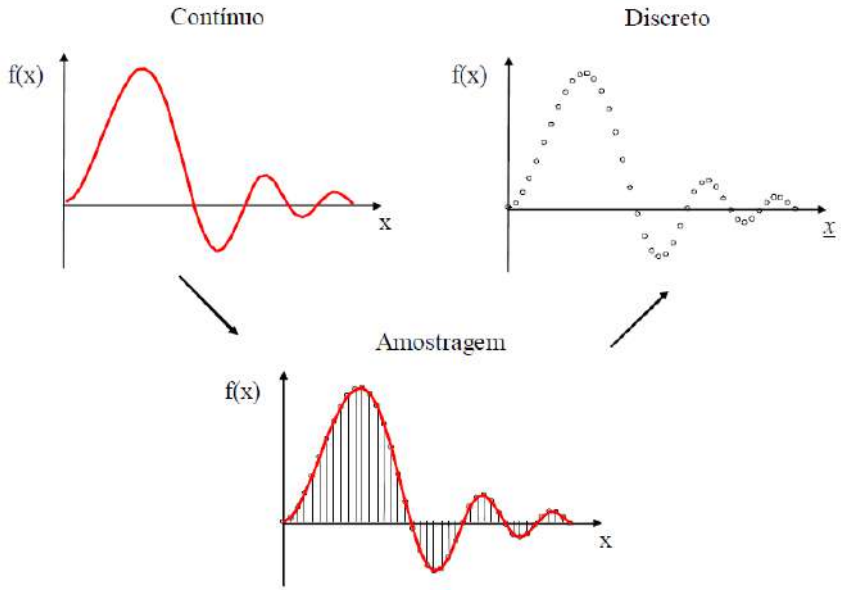


Figura 2.10: Demonstração da ocorrência de falseamento de sinal - SCURI (1999).

A determinação incorreta da taxa de amostragem do sinal pode acarretar na perda de informações necessárias, chamado de *Aliasing*. SCURI (1999) descreve matematicamente *Aliasing* como uma má definição da taxa de amostragem, ou seja, a frequência de amostragem está abaixo da frequência da função contínua, que no caso o sinal desejado.

Portanto, a inconsistência de dados resulta no falseamento do amostrado, este fenômeno pode ser visto na figura 2.11.

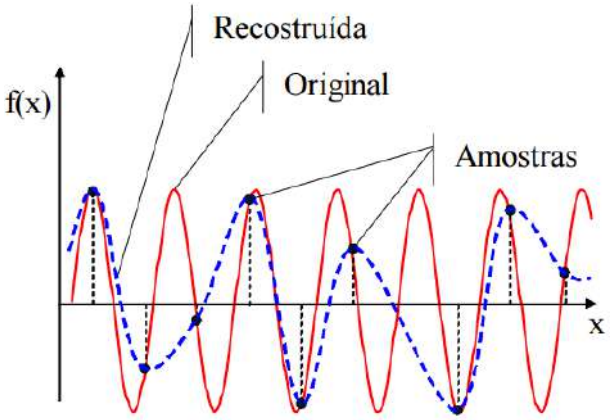


Figura 2.11: Demonstração da ocorrência de falseamento de sinal - Fonte: SCURI (1999).

Para PRITHA; JESLET (????) três questões são relevantes para o processo de amostragem: *i)* como as amostras em tempo discreto serão obtidas a partir do sinal de tempo contínuo; *ii)* como reconstruir o sinal de tempo contínuo de um sinal discreto; *iii)* quais condições pode-se recuperar o sinal de tempo contínuo.

2.2.2 Resolução

Resolução é descrito por AZEVEDO; CONCI (2003) como a quantidade de pontos individuais de uma imagem (*pixels*) um monitor é capaz de comportar em seus eixos horizontal e vertical. Para processamento de imagem a resolução é fundamental para correção de alguns fatores.

Para SCURI (1999) resolução trata-se de da razão entre o número de *pixels* e o tamanho da imagem real, equação 2.1.

$$resolucao = \frac{pixels}{tamanhoreal} \tag{2.1}$$

O autor relata que a unidade de medida da resolução é medida em pontos por polegada - DPI (*dots per inche*) - ou por centímetros - DPC.

A relação presente a equação 2.1, pode ser vista através da Figura 2.12 em que retrata a relação *resolução-pixels* quando mantem-se o tamanho fixo e varia-se a resolução.

Já a figura 2.13 retrata a relação *tamanho-pixels* quando mantem-se a resolução fixa e varia-se o tamanho.

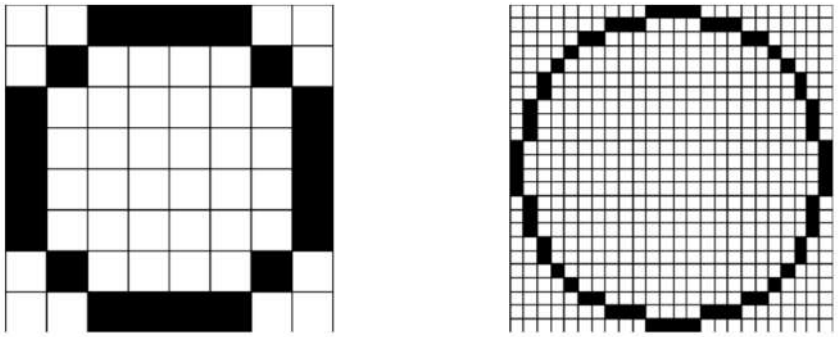


Figura 2.12: Imagens com o mesmo tamanho e diferentes resoluções - Fonte: SCURI (1999).

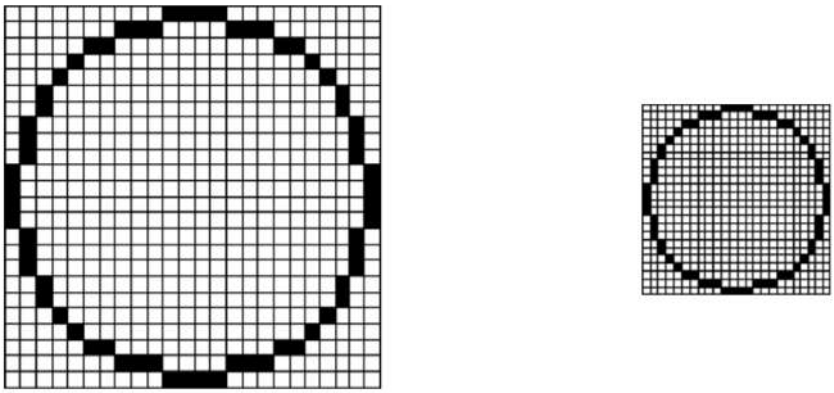


Figura 2.13: Imagens com a mesma resolução e tamanhos diferentes - Fonte: SCURI (1999).

Pixel

"*Pixel*, palavra criada a partir da expressão em inglês *picture element*, é uma unidade básica da imagem que pode ser controlada individualmente e que contém informações sobre cores e intensidade" - descreve AZEVEDO; CONCI (2003). Deve-se encarar essa unidade básica da imagem como uma unidade lógica e não uma unidade física, pois, seu tamanho físico é definido pela resolução da imagem.

SCURI (1999) vai um pouco além, definindo-os como quadrados preenchidos com a cor respondente à sua intensidade, em que seu conjunto, *pixel* e vizinhança, formam uma grade (*grid*) uniformemente espaçada, conforme Figura 2.14.

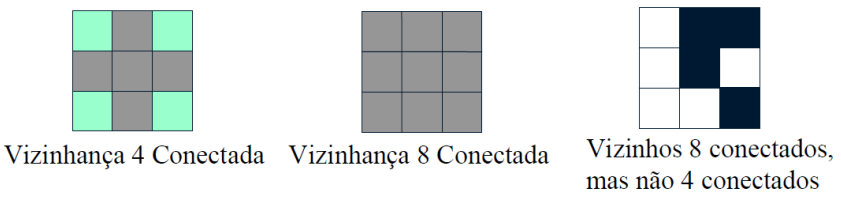


Figura 2.14: Vizinhança de um *Grid* regular - Fonte: SCURI (1999).

2.2.3 Ruídos

MARENGONI; STRINGHINI (2009) ruídos em imagens são provenientes de várias fontes, dentre elas pode-se destacar: o tipo de sensor utilizado, a iluminação do ambiente, as condições climáticas no momento da captura da imagem, a posição relativa entre o objeto de interesse e a câmera. Sendo assim, o ruído não é proveniente apenas da interferência do sinal na captação da imagem, mas também, dos fatores influentes do ambiente, que dificultam na compreensão ou verificação de objetos

Conforme DAWSON-HOWE (2014) a medida mais comum de ruído é a relação sinal/-ruído. Este, para uma imagem $f(i,j)$ é definida pela equação 2.2.

$$\frac{S}{N_{ratio}} = \frac{\sum_{(i,j)} f^2(i,j)}{\sum_{(i,j)} v^2(i,j)} \tag{2.2}$$

Em que $v(i,j)$ é o ruído. Para análise de processamento de imagem pode-se ressaltar dois modelos de ruídos - Ruído Gaussiano e Ruído Impulsivo.

De acordo com DAWSON-HOWE (2014), o ruído Gaussiano é uma perfeita estimativa do ruído real. O ruído $v(i,j)$ modelado usando a distribuição Gaussiana em torno de um termo médio (μ), normalmente esse termo é zero, com um valor estimado de desvio padrão (σ). Um modelo de ruído gaussiano pode ser visto na Figura 2.15.

Figura 2.15: Imagens de cor (esquerda, cima) e escala de cinza (esquerda, baixo). Estas com ruído gaussiano com um termo médio de 0 e com desvio padrão de 20 - Fonte: DAWSON-HOWE (2014).

Para imagens com cores a relação sinal/ruído para as que possuem ruídos são de 43,3. Já para imagens em escala de cinza a relação é de 40,3.

O ruído impulsivo, ainda de acordo com o autor, é a corrupção com *pixels* ruidosos individuais em que possui o brilho diferente da vizinhança. Este é um tipo de ruído de

impulso em que é saturado e afeta a imagem, resultando em um *pixel* corrompido com brancos e pretos puros, podendo ser acompanhado na Figura 2.16.

Figura 2.16: Imagens de cor (esquerda, cima) e escala de cinza (esquerda, Baixo). Estas, a direita, com 10% de ruído impulsivo - Fonte: DAWSON-HOWE (2014).

Para imagens com cores a relação sinal/ruído para as que possuem ruídos de impulso são de 7,5, para imagens em escala de cinza são 6,7.

2.3 Visão Computacional

Visão computacional preocupa-se em realizar a integração automática de uma ampla gama de processos e representações usadas na percepção visual e no processamento cognitivo. A utilização de computação e algoritmos para reconhecimento de imagens busca gerar descrições inteligentes e úteis de cenas e sequências visuais dos objetos que estão presente na cena, realizando-se assim operações nos sinais recebidos por dispositivos capazes de realizar a captação dos sinais visuais. O objetivo da visão computacional é simplificar as imagens, realizando seu tratamento na etapa de processamento de imagem, até que o final da resolução de um determinado problema DATTA; MUNSHI (2016)

HUANG (1996) retrata a visão computacional no ponto de vista da ciência biológica em que apresenta modelos computacionais do sistema visual humano. E do ponto de vista da engenharia, em que visa construir sistemas autônomos que possam desempenhar algumas tarefas que o sistema visual humano pode realizar, de forma a até superá-los.

2.3.1 OpenCV

De acordo com BRADSKI; KAEHLER (2008) a *Open Source Computer Vision* surgiu de uma iniciativa da *Intel Research* para avançar com aplicações intensivas de CPU. Deste

modo, foi criado uma biblioteca de visão computacional, chamada OpenCV, com o objetivo de avançar na pesquisa da visão computacional, fornecendo não só código aberto, mas também otimizado para infra-estrutura de visão básica. Além de disseminar o conhecimento da visão, fornecendo uma infra-estrutura comum que os desenvolvedores poderiam construir. Esta biblioteca é programada em linguagens C e $C++$ e funciona nos sistemas operacionais *Linux, Windows e MAC OS X*. A OpenCV possui um desenvolvimento ativo em interfaces para *Python, Ruby, Matlab* e outras linguagens.

Ainda de acordo com o autor, a biblioteca contém mais de 500 funções que abrangem muitas áreas de visão, incluindo inspeção de produtos, imagem médica, segurança, interface do usuário, calibração de câmera, estéreo visão e robótica. Como a visão computacional e a aprendizagem mecânica frequentam a mesma mão, a OpenCV também contém uma biblioteca de aprendizagem automática (MLL) de uso geral. O MLL é altamente útil para as tarefas de visão que estão no cerne da missão do OpenCV, mas é suficiente para ser usado para qualquer problema de aprendizagem da máquina.

Segundo MARENGONI; STRINGHINI (2009), a biblioteca está dividida em cinco grupos de funções: Análise de movimento e rastreamento de objetos; Análise estrutural; Processamento de imagens; Reconhecimento de padrões e calibração de câmera e reconstrução 3D. O autor ainda ressalta que o código está disponível em código fonte e seus executáveis otimizados para os processadores Intel. Por isso, ao executar um programa OpenCV, uma DDL (*Dynamic Linked Library*) é que automaticamente reconhece o tipo de processador, por sua vez, carrega a DDL otimizada para este.

2.3.2 Segmentação

MARENGONI; STRINGHINI (2009) salienta que a divisão da imagem em regiões guiada por característica da cena, consiste no processo de segmentação. A eficiência em questão de detalhamento da segmentação é dependente do quão a imagem está detalhada, ou seja, do quão grande é sua resolução. A openCV contém diversas técnicas para realizar a segmentação de uma imagem, pode-se destacar duas: A Segmentação por detecção de borda e a Segmentação por corte.

- Segmentação por detecção de borda: Esta técnica aborda a mudança no nível de intensidade dos *pixels*, caracterizando-se assim uma borda. Portanto, é identificado essa variação abrupta de intensidade dos *pixels* e verifica-se o comportamento dos *pixels* da vizinhança, caso forem semelhantes realiza-se a conexão entre eles formando uma borda ou contorno identificando o objeto. A biblioteca OpenCV realiza essa identificação da variação das intensidades dos pontos através da primeira e/ou segunda derivada, matematicamente, utiliza-se os operadores de gradiente, os mais comuns são: *Prewitt*, equação 2.3, e *Sobel*, equação 2.5.

$$g(x) = \frac{\partial f(x,y)}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) \tag{2.3}$$

$$g(x) = \frac{\partial f(x,y)}{\partial y} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7) \tag{2.4}$$

$$g(x) = \frac{\partial f(x,y)}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \tag{2.5}$$

$$g(x) = \frac{\partial f(x,y)}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \tag{2.6}$$

Estes dois métodos utilizam a definição de uma máscara que assemelham-se às variações, e em seguida realiza-se a convolução[3] da imagem pela máscara. A Figura 2.17 apresenta um modelo de máscara e os operadores de *Prewitt* e *Sobel*.

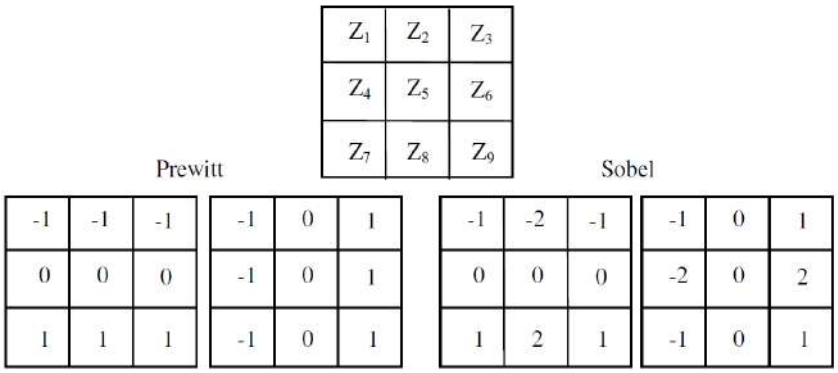


Figura 2.17: Representação de uma máscara genérica com o posicionamento dos valores utilizados (cima) nas equações de Prewitt e Sobel. Abaixo, à esquerda, as máscaras dos operadores horizontal e vertical de Prewitt e a direita as máscaras do operador de Sobel - Fonte:MARENGONI; STRINGHINI (2009).

- Segmentação por corte: Esta segmentação divide a imagem por meio dos valores de intensidade ou propriedades desses valores, ou seja, a segmentação por corte é realizada através dos valores extremos de intensidade da imagem, e assim utilizando-se do histograma de cor realiza a segmentação a partir de uma faixa, realizando tratamentos no histograma.

2.3.3 Reconhecimento de Padrões

ARAUJO (2010) define reconhecimento de padrões como algoritmo capaz de distinguir objetos (ou padrões) em categorias, podendo ser utilizado para o aprendizado da

[3]Em processamento de sinal, é um operador linear que, a partir de duas funções dadas, resulta numa terceira que mede a área subentendida pela superposição delas em função do deslocamento existente entre elas.

Figura 2.18: Representação da segmentação por corte, imagem original à esquerda. Imagem binarizada por segmentação à direita - Fonte: MARENGONI; STRINGHINI (2009).

máquina. O autor relata que o processo de reconhecimento de padrões é dividido e cinco etapas: Sensoreamento; Segmentação; Extração de características; Classificação e; Pós-processamento. Já HAYKIN (2001) descreve que reconhecimento de padrões é um processo em que um sinal captado é concedido à uma classe dentre diversas classes.

Método HaarCascade

Conforme a documentação presente em OPEN (2015) O processo de detecção de objetos utilizando-se de classificadores de cascata baseados nos recursos de Haar é um dos métodos mais eficazes na detecção de objetos proposto por Paul Viola e Michael Jones. Este método é uma abordagem baseada em aprendizado de máquina em que a função em cascata é treinada a partir de muitas imagens positivas e negativas, sendo que é utilizado para detectar objetos em outras imagens.

O método HaarCascade pode ser utilizado para a identificação de objetos assim como a identificação de rostos através de pontos característicos. O algoritmo precisa de muitas imagens com faces, chamadas de imagens positivas, e imagens sem faces, chamadas de imagens negativas para que o classificador seja treinado. O primeiro recurso selecionado parece se concentrar na propriedade de que a região dos olhos é geralmente mais escura que a região do nariz e bochechas. A segunda característica depende da propriedade de que os olhos são mais escuros do que a ponte do nariz.

Para analisar as características de todas as imagens de treinamento, para cada recurso é encontrado o melhor limiar que classificará as faces para positivo e negativo, podendo haver erros de classificação, sendo minimizados através dos recursos utilizados. Cada imagem recebe um peso, após cada classificação, os pesos das imagens mal classificadas são aumentados, então novas taxas de erros são calculadas.O classificador final é uma soma ponderada desses classificadores fracos.

Método Background Subtraction

OPEN (2015) descreve o método *Background Subtraction*(subtração de fundo) como uma técnica comum e amplamente utilizada para gerar uma máscara de primeiro plano, ou seja, uma imagem binária que contém os *pixels* pertencentes a objetos em movimentos na cena. Como o nome descreve, a subtração de fundo calcula a máscara de primeiro plano realizando a subtração entre o quadro atual e um modelo de fundo, contendo a parte elástica da cena ou, mais em geral, tudo o que pode ser considerado como fundo dado as características da cena observada, a atuação do método pode ser observada através da Figura 2.19.

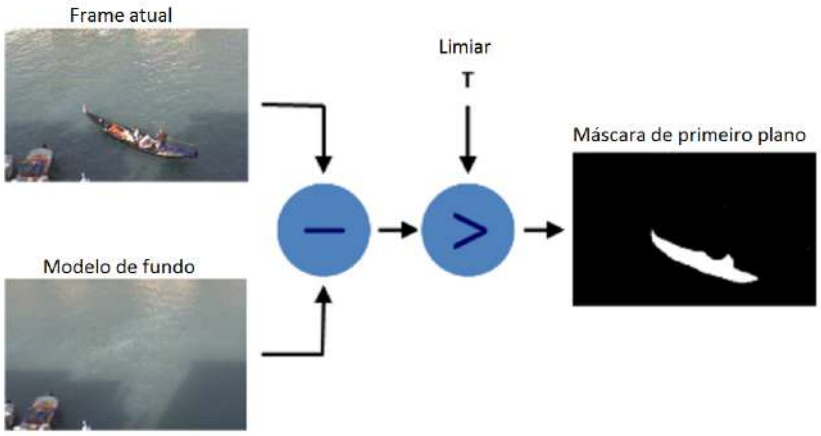


Figura 2.19: Representação da atuação do método Background Subtraction - Fonte: OPEN (2015).

A modelagem da cena de fundo é realizada de duas formas: Inicialização de fundo e; Atualização de fundo. Na primeira etapa, um modelo inicial de fundo é obtido, enquanto na segunda etapa esse modelo é atualizado para se adaptar à possíveis mudanças na cena.

2.3.4 Filtros

OPEN (2015) apresenta um conteúdo a respeito de filtros relatando que podem ser implementados através de funções e classe algorítmicas resultando em filtragens lineares ou não lineares de uma imagem 2D. Para cada local de *pixel* na imagem de origem, sua vizinhança é considerada e usada para calcular a resposta correspondente.

O filtro linear, é a soma ponderada dos valores de cada *pixel*, em que em operações morfológicas, são os valores mínimos e máximos, e assim por diante. A resposta, após os cálculos, é armazenada no *array* da imagem de destino no mesmo local, portanto, a imagem de saída será do mesmo tamanho que a imagem de entrada.

2.3.5 Desfocagem

OPEN (2015) relata que a desfocagem de imagem é realizada convolvendo a imagem com um *kernel* de filtro de passagem baixa, sendo muito útil para remoção de filtros, portanto, as arestas são borradas um pouco nessa operação.

Método Gaussian Blurring

OPEN (2015) retrata que o desfoque Gaussiano é feito utilizando-se a função *GaussianBlur()*. Neste método, deve-se especificar a largura e altura do*Kernel*, que deve ser aleatório e positivo. Deve-se também, especificar o desvio padrão em direção X (σx) e Y (σy), caso o valor de ambos forem dados como zeros, eles são calculados utilizando-se o tamanho de *kernel*. A *sintaxe* do método pode ser observada pela linha de código a seguir.

```
blur = cv2.GaussianBlur(img,(5,5),0)
```

Método Median Blurring

OPEN (2015) relata que a função *cv2.medianBlur()* leva a mediana de todos os *pixels* na área do *kernel* e o elemento central é substituído por esse valor médio, este método possui grande eficácia contra ruídos impulsivos. A sintaxe pode ser observada através da linha de código a seguir.

```
median = cv2.medianBlur(img,5)
```

2.4 Luz e Cor

Para HIRSCHLER (2009), luz é uma constituição de várias ondas eletromagnéticas emitidas de um ponto de origem a uma velocidade de 300000 km/s no vácuo. Pode-se dizer que a captação de cores compreendida pelo olho humano possui comprimentos de ondas entre a faixa de 380nm e 760nm, Figura 2.20.

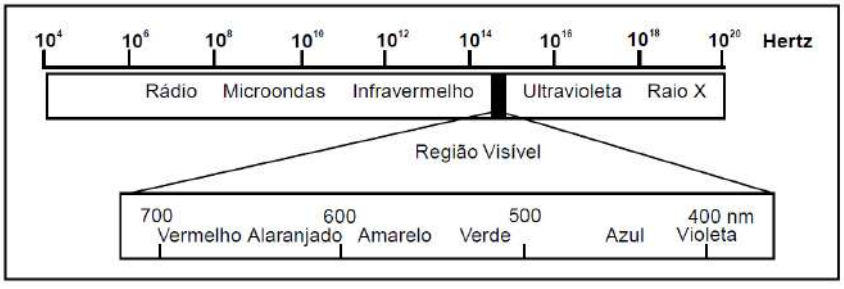


Figura 2.20: Espectro eletromagnético - Fonte: AZEVEDO; CONCI (2003).

Utilizando-se da luz solar, que é compostas por diversos comprimentos de onda, em que cada um resulta em uma cor, Issac Newton em seu experimento denominado Teoria do Prisma aplicou um faixe de radiação solar (luz branca) e obteve-se então um espectro [*Spectrum*][5] de cores visíveis, Figura 2.21, da mesma forma que acontece em uma arco-íris. Pode-se então comprovar que o aglomerado de cores diferentes resultam na luz branca, STEFEN (2008).

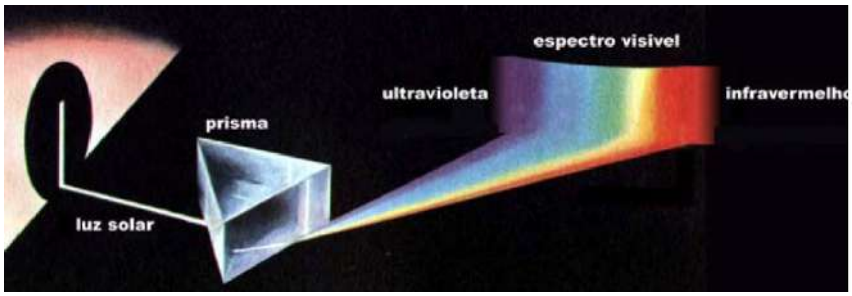


Figura 2.21: Experimento realizado por Isaac Newton para comprovar sua Teoria do prisma, que descreve que um feixe de luz branca é composto por um espectro de cores - Fonte: STEFEN (2008).

Foi observado que houve uma dispersão da luz policromática [6] quando ultrapassada pelo prisma resulta numa sequência de cores de radiações monocromáticas [7], Figura 2.22

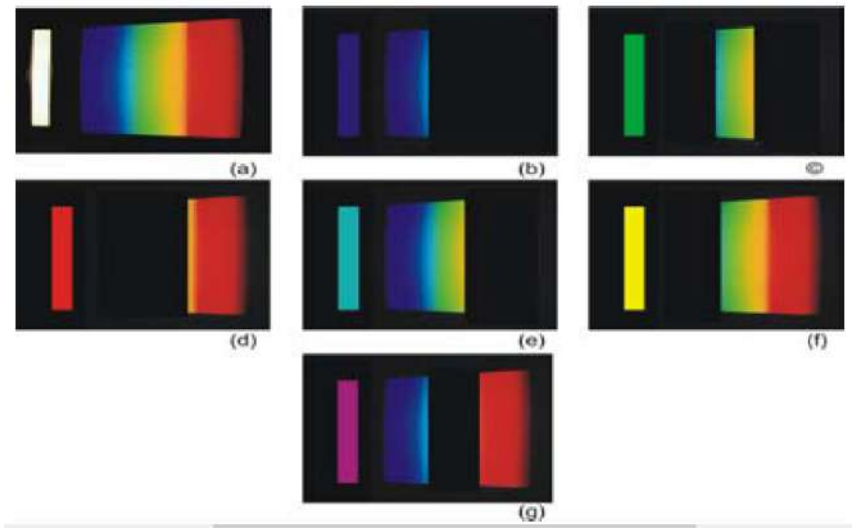


Figura 2.22: Decomposição do espectro de cores - (a) luz branca com seu respectivo espectro; (b) luz azul, (c) luz verde; (d) luz vermelha; (e) luz ciano; (f) luz amarela e (g) luz magenta.

Para dar fundamento em sua teoria, Isaac Newton distinguiu-se um novo conceito de

[5]Em latim, no original. Newton foi o primeiro a usar a palavra *spectrum* com o significado de um segmento de cores formada por um feixe de luz branca após atravessar um prisma, em seu experimento denominado Teoria do Prisma de Newton.

[6]Combinação de dois ou mais comprimentos de onda.

[7]Composta por apenas um comprimento de onda.

cor primária e cor composta. A luz que pode ser segmentada em duas ou mais fragmentos do prisma é denominada luz composta, enquanto a luz primária não. Não há um número finito de cores primárias visíveis que produza realmente todas as cores, mas pode-se produzir maior parte delas com três primárias escolhidas das extremidades e centro do espectro de luz. A escolha de três cores primárias é devido ao fato de que os olhos humanos possuírem três tipos de sensores para captação de cores diferentes, sensíveis a diferentes partes do espectro de luz visível, SILVA; ANDRADE MARTINS (????).

Outros conceitos de luz e cor que são relevantes para a computação gráfica são os conceitos de cores aditivas e subtrativas. AZEVEDO; CONCI (2003) relata que vermelho, verde e azul são cores primárias aditivas, o preto a ausência de qualquer outra cor (não havendo luz sendo transmitida). Já o branco é a combinação de todas elas, resultado da intensidade máxima de vermelho, verde e azul, Figura 2.23.

Figura 2.23: Processo aditivo de cores primárias.

A representação da cor de cada *pixel*(C) em uma imagem colorida pode ser representada matematicamente pela equação 2.7

$$C = r.R + g.G + b.B \tag{2.7}$$

Em que (R,G,B) são as três cores primárias e (r,g,b) são os coeficientes de intensidade de cada cor correspondente as cores primárias utilizadas.

Ainda de acordo com o autor, o sistema de cores subtrativas é altamente utilizado por indústrias de pintura devido a utilização das chamadas cores primárias subtrativas: *magenta*, amarelo, *ciano*, estas possuem o efeito de absorver parcialmente a luz branca.

2.4.1 Sistema Visual Humano

Conforme DATTA; MUNSHI (2016) a percepção visual é a habilidade humana de interpretar uma cena processando a informação que ali está contida pelo aparelho olho-cérebro. Este processo ativo do cérebro iniciasse na visão, e pode ser definido como o processo ou resultado de processos da construção da representação do ambiente na mente do observador. O sistema visual humano, assim, é consistido por duas partes. A primeira parte, os olhos atuam como sensores de imagens que captam luzes e as convertem em sinais elétricos ou neurais, através de processo químico. Estes sinais elétricos são transportados para o centro de processamento de imagens no cérebro e são processados construído uma réplica interna da cena sendo visualizada. A segunda parte, o cérebro atua de forma a processar a imagem e construir e manipular o modelo interno da cena.

Conforme AZEVEDO; CONCI (2003) bastonetes e cones são dois tipos de células presentes na retina do olho humano. Os bastonetes são sensíveis a baixa luminosidade e são capazes de diferenciar tons de cinza, além de serem capazes pela visão periférica. Já os cones, ao contrário dos bastonetes, são sensíveis a alto nível de brilho e responsáveis pela compreensão de cores.

Para que seja possível identificar o formato de objeto colorido, o observador deve voltar os receptores para que os cones estimule seus grupos sensíveis as cores azul, verde e vermelho. Assim, as células que se assemelham às cores da imagem são excitadas e enviam ao cérebro seus respectivos sinais nervosos quanto a imagem do objeto colorido é visualizada, STEFEN (2008).

2.4.2 Espaço de cor

Há distintas formas de definir o termo espaço de cores, sempre concordando com certas situações. Neste texto utilizou-se a expressão na representação gráfica das relações entre cores, para TKALCIC; TASIC (2003), a cor é a forma que o sistema visual humano compreende uma parte do espectro eletromagnético, podendo ser visível numa faixa de comprimento de onda de $300nm$ à $830nm$, não sendo possível ver todas as combinações de espectro, mas, tende-se a agrupar vários espectros em cores. Portando, ainda de acordo com o autor, um espaço de cor é uma notação através da qual pode-se especificar cores, ou seja, a percepção humana do espectro eletromagnético visível.

RGB

O espaço de cores RGB representa-se por um cubo de lados da base R e G, e altura B, correspondentes respectivamente a vermelho ($700nm$), verde ($546{,}1nm$) e azul (435,8nm), Figura 2.24. Estas, chamadas cores primárias, podem ser combinadas em várias proporções para obter outras no espectro visível humano. Neste contexto, cada cor no sistema

RGB é identificado por uma tripla ordenada (R,G,B) de números inteiro correspondentes a tonalidade e intensidade com $0 \leq R \leq 255$, $0 \leq G \leq 255$, $0 \leq B \leq 255$, correspondendo possíveis 16.777.216 cores (256x256x256), MARQUES FILHO; NETO (1999).

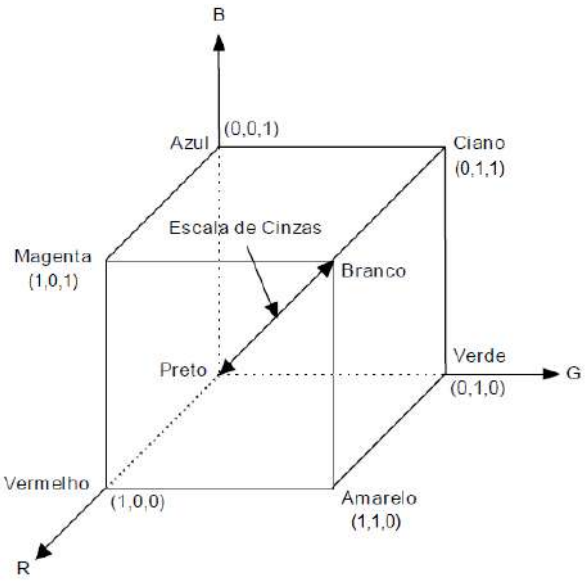


Figura 2.24: Representação cubo do modelo RGB com valores máximos de R,G e B normalizado na faixa de 0 a 1 - Fonte: LIST (2015).

Para conhecimento da obtenção de cores por distribuição do vetor RGB, a Figura 2.25 apresenta algumas cores por intensidade dos parâmetros R-G-B.

Nome	Cor	R	G	B	(R, G, B)
Abóbora		255	117	24	(255, 117, 24)
Água-marinha		127	255	212	(127, 255, 212)
Amarelo		255	255	0	(255, 255, 0)
Azul		0	0	255	(0, 0, 255)
Branco		255	255	255	(255, 255, 255)
Bronze		205	127	50	(205, 127, 50)
Canela		210	180	140	(210, 180, 140)
Dourado		218	165	32	(218, 165, 32)
Dourado escuro		184	134	11	(184, 134, 11)
Ouro		255	215	0	(255, 215, 0)
Preto		0	0	0	(0, 0, 0)
Rosa		255	192	203	(255, 192, 203)
Verde		0	255	0	(0, 255, 0)
Verde claro		144	238	144	(144, 238, 144)
Verde escuro		0	100	0	(0, 100, 0)
Vermelho		255	0	0	(255, 0, 0)
Vermelho escuro		139	0	0	(139, 0, 0)

Figura 2.25: Representação de alguns modelos de cores e seus respectivos parâmetros RGB - Fonte: COR (2008).

Dentre outros espaços de cores, pode-se citar alguns como: CMY, YUV e HSV. O modelo CMY é baseado em cores secundárias - Ciano, Magenta, Amarelo -,e é um esquema

de cores subtrativo. O modelo YUV é utilizado para sinais de televisão analógicos, sua composição é definida por luminância [8] (Y) juntamente com duas componentes de cor: Azul (U) e vermelho (V). Já o modelo HSV é inspirado na forma que o artista descreve e mistura cores.

2.4.3 Histograma de cores

Histogramas de cores são essenciais para o processamento de imagem. Eles são determinados a partir da intensidade de cada *pixel*, ou seja, o valor que cada *pixel* comporta. Dentre as diversas aplicações que os histogramas podem ser submetidos, para o autor, vale ressaltar a melhora da definição de uma imagem e a segmentação de imagens MARENGONI; STRINGHINI (2009).

Pode-se demonstrar o histograma de uma imagem (A) em escala de cinza, cujos os valores de intensidade estejam numa determinada faixa de valor (I_{min} e I_{max}) através da equação 2.8.

$$h(A_k) = n_k \tag{2.8}$$

Em que, A_k é o valor de intensidade k ($I_{min} < k < I_{max}$) da imagem A e n_k é a quantidade de *pixels* na imagem que possuem intensidade k. Obtendo-se a o histograma da imagem, possibilita-se a realização da normalização deste, através da representação de termos em porcentagem, conforme equação 2.9.

$$p(A_k) = \frac{h(A_k)}{n} = \frac{n_k}{n} \tag{2.9}$$

onde n é a quantidade de *pixels* da imagem.

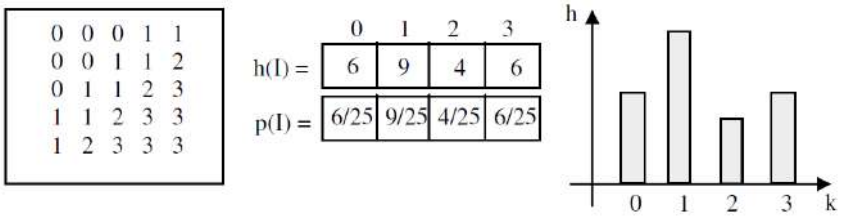


Figura 2.26: Representação de uma imagem digital (A) à esquerda, histograma da imagem em valores (h(A)) e normalizada (p(A)) ao centro e a representação gráfica à direita - Fonte: MARENGONI; STRINGHINI (2009).

Já AZEVEDO; CONCI (2003) ressalta os conceitos de histograma para imagens coloridas, o autor, relata que a distribuição de cores de uma imagem colorida caracteriza um diagrama de cor, em que, cada *pixel* é representado em um vetor 3D. Ou seja, para um

[8] Luminância: É a medida da densidade da intensidade de uma luz refletida numa dada direção.

determinado espaço de cor, no caso do RGB, há três histogramas, em que cada um representa um canal de cor, matematicamente o histograma de uma imagem colorida (I[x,y]) pode ser visto na equação 2.10.

$$(V) = (I_r[x,y], I_g[x,y], I_b[x,y]) \tag{2.10}$$

em que, x ϵ [1,X] e y ϵ [1,Y] com X e Y correspondendo respectivamente a altura e largura da imagem.

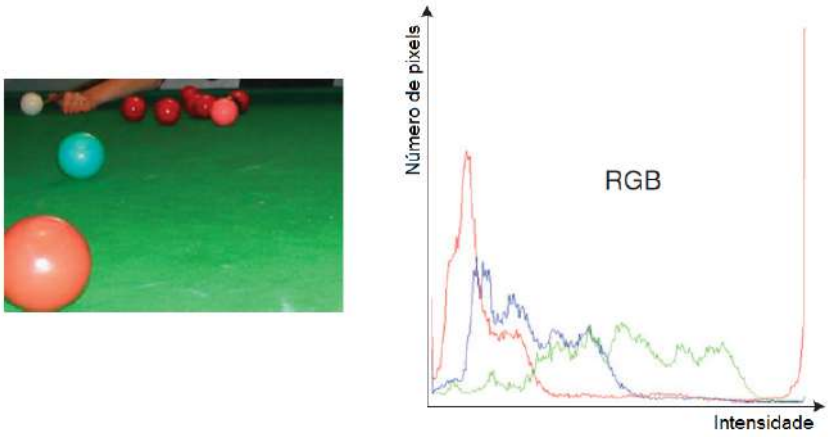


Figura 2.27: Histograma dos canais RGB - Fonte: DAWSON-HOWE (2014).

2.5 Sistemas embarcados

CUNHA (2007) descreve sistemas embarcados como dispositivos, sistemas ou equipamentos capazes de realizar atividades computacionais pré-programadas. O autor também registra que a estrutura básica de um sistema embutido, figura 2.28, é composta por: Conversor analógico-digital; CPU; Memória; Conversor digital-analógico; ambiente de programação (FPGA).

2.5.1 Raspberry pi

Raspberry PI foram originalmente concebidos para inspirar jovens programadores a aprimorar seus conhecimentos dos alunos de Cambridge. O dispositivo, Figura 2.29, é totalmente diferente de um computador, a começar pelo tamanho que é praticamente do tamanho de um cartão de crédito, o preço também se torna um atrativo para o equipamento de pequeno porte e grande capacidade computacional.

Um *chip* único contém a memória, unidade central de processamento e chip gráfico. O Raspberry Pi usa um chip projetado por ARM, a mesma empresa com base em Cambridge que projeta processadores usados em muitos *smartphones* e *tablets*. Sua unidade de

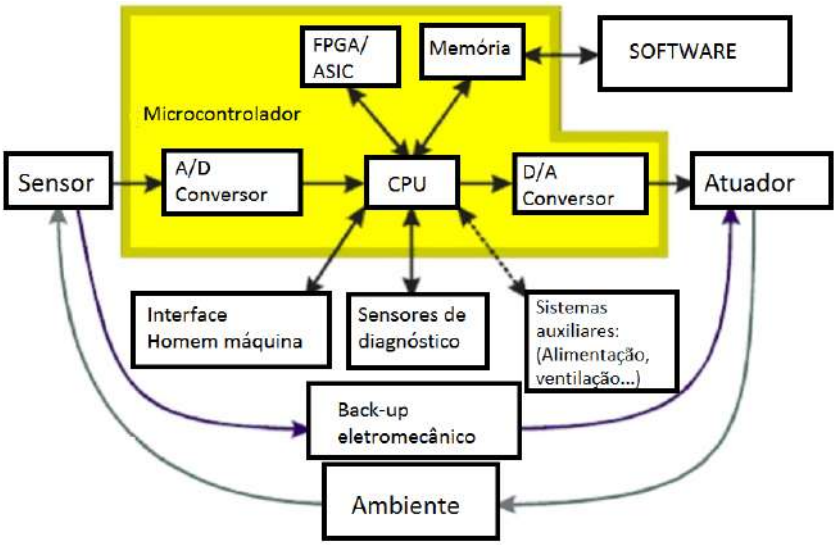


Figura 2.28: Elementos básicos de um sistema embarcado - Fonte CUNHA (2007).

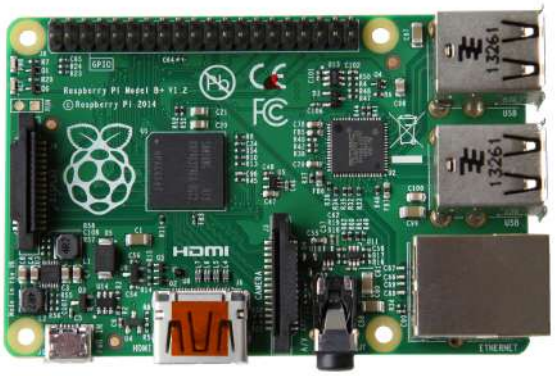


Figura 2.29: Ilustração de um Raspberry Pi 3B

processamento é equivalente a uma especificação elevada de um dispositivo móvel. Pois, pode executar jogos em 3D e executar vídeos de alta definição.

2.6 Comunicação Wireless

As chamadas comunicações sem fios são o segmento de mais rápido crescimento no setor de comunicações. Além da utilização em dispositivos móveis e residências, há diversas utilizações em meio industriais: redes de sensores sem fio, rodovias e fábricas automatizadas, casas e eletrodomésticos inteligentes e telemedicina. Há diversos meios de comunicação *Wireless*, dentre eles pode-se destacas: *Bluetooth*, Wi-Fi e ZigBee.

2.6.1 Bluetooth

Bluetooth, nome em homenagem ao rei da Dinamarca entre 940 e 985 DC, Harald I Bluetooth, que uniu a Dinamarca e Noruega, isso, devido ao fato que o *Bluetooth* propõe

Tabela 2.2: Comparação de parâmetros das comunicações Wireless

	Bluetooth	Wi-fi	ZigBee
Especificação IEEE	802.15.1	802.11a/b/g	802.15.4
Banda de frequência	2.4Ghz	2.4Ghz;5Ghz	868/915MHz;2.4GHz
Máxima taxa de sinal	1Mb/s	54Mb/s	250Kb/s
Alcance Nominal	10m	100m	10-100m
Nominal energia de transmissão	0-10dBm	15-20dBm	(-25)-0 dBm
Número de canais RF	79	14	1/10;16
Largura de banda do canal	1MHz	22MHz	0.3/0.6 MHz;2MHz
Tipo de modulação	GFSK	BPSK,QPSK	BPSK(+ASK)O-QPSK

unir dispositivos através de conexões de rádio, daí surge a inspiração para o nome. Estes raios fornecem conexão de curto alcance entre dispositivos sem fio, juntamente com redes rudimentares. O *bluetooth* é o principal meio de comunicação de baixo alcance, operando em uma faixa de frequência de 2,4GHz, portanto, pode ser utilizada sem licença. Esta comunicação fornece um canal de dados assíncronos em 723.2 Kbps. Este, também utiliza saltos de frequências para acesso múltiplo com um espaçamento de suporte de 1MHz, normalmente, até 80 frequências diferentes são usadas para uma largura de banda de 80MHz GOLDSMITH (2005).

LEE; SU; SHEN (2007) já descreve os termos técnicos desta comunicação, ressaltando que o *Bluetooth* também é conhecido como padrão IEEE 802.15.1 com base em um sistema de rádio sem fio projetado para curto alcance e dispositivos baratos para substituir canos para periféricos de computadores, tais como: *Mouses*, teclados, *joysticks* e impressoras. O autor descreve que essa gama de aplicativos são conhecidos como rede de área pessoal sem fio (WPAN).

2.6.2 Wi-fi

Wi-fi é um padrão de rede de área local desenvolvido pelo grupo de trabalho IEEE 802.11 e foi projetado para ser usado em ambientes fechados, para distribuir o acesso à Internet a um monte de computadores em uma casa ou em um escritório. LEE; SU; SHEN (2007) diz que Wi-Fi possui padrões para redes de área local sem fio (WLAN). Esta, permite que usuários naveguem na *Internet* à velocidades de banda larga quando conectados por um ponto de acesso (AP). A arquitetura do Wi-Fi consistem em vários componente que interagem para fornecer uma rede sem fio que suporte a mobilidade da estação de forma transparente para as camadas superiores.

Capítulo 3

Desenvolvimento

3.1 Projeto Conceitual

O projeto conceitual foi instituído através da definição das condições de operação do sistema. Estas, determinadas para o dispositivo embarcado posicionado em um ambiente na ausência da alteração das variáveis de posicionamento, luminosidade e outras que possam causar ruídos na captação dos dados e assim interferir no processamento. Isto garantiu a capacidade do dispositivo em monitorar as variáveis no campo de visão do sistema. O posicionamento fixo do sistema foi garantido através do desenvolvimento do acoplamento, isto impossibilitou a modificação da angulação e assim extinguiu variância na perspectiva da vista. A perspectiva da imagem foi corrigida no momento da instalação do dispositivo, sendo assim, enquanto não houver a troca de posicionamento não ocorrerá erros. Outro fator é a luminosidade, sendo que a identificação dos perfis é realizada por imagem, a alteração desta irá causar uma grande problemática, pois o dispositivo foi configurado para que esta variável seja fixa.

3.1.1 Análises do dispositivo embarcado

Diante dos microcontroladores estudados, definiu-se o que mais se encaixa nos parâmetros do projeto. O *Raspberry Pi 3B* é o ideal para o sistema, pois possui uma arquitetura que comporta bem as necessidades de processamento, além de possuir periféricos que facilitarão na aplicação e utilização de dispositivos.

Os requisitos que foram fundamentais para a escolha deste dispositivos estão presentes na Tabela 3.1.

Periféricos de comunicação

A presença dos periféricos de comunicação *Wireless*, *Bluetooth* foram de fundamental importância para a escolha. A presença destes, além da capacidade de utilizar o equipa-

Tabela 3.1: Configurações relevantes do Raspberry Pi 3B

	Raspberry PI 3B
Núcleos do CPU	4
Frequência CPU	1.2GHz
GPU Design	Cortex A53
Frequência GPU	400MHz
USB 2.0	4
HDMI	1200P60
Ethernet	100Mb
Wireless	802.11N BCM43438
Bluetooth	Bluetooth 4.1
OS Linux	Sim

mento como um servidor de dados, reduzem o custo da construção do projeto, abonando a necessidade da compra de seus módulos, além de possuir também a porta para comunicação *Ethernet*. Enquanto os outros sistemas mais acessíveis financeiramente não possuem esta diversidade de periféricos de comunicação, possuindo apenas um.

Entradas para câmera

Pode-se destacar as diversas possibilidades de utilização da câmera para o dispositivo. Estas são dispostas em: Câmera USB, módulos de câmera, câmeras *ethernet*. O módulo de câmera, figura 3.1, foi utilizado pois possui uma otimização em relação ao seu tamanho podendo deixar a estrutura do dispositivo embarcado compacta, além da variedade de modelos que possuem grande resolução e captação de um amplo espaço, outras características relevantes para a escolha do módulo estão presentes na tabela 3.2

Figura 3.1: Ilustração do módulo de câmera acoplado ao Raspberry Pi 3B

Tabela 3.2: Características da câmera.

Sensor	$5MP$
Vídeo	1080p a 30fps
Tamanho	3,67 x 2,74 mm
Lente	f = 3,6mm, f/2.9
Angulação	54 x 41 graus
Foco	1m ao infinito
CSI	Câmera de interface serial
Tamanho da placa	25 x 24 mm

Acessibilidade

A acessibilidade do *Raspberry* Pi 3B diante a documentação presente para sua utilização é um fator que influência na escolha da plataforma, além de conter um dispositivo disponível para utilização na instituição. Uma vez que não há unidades de outras plataformas para utilização disponíveis no CEFET-MG, além da documentação dos outros dispositivos ser escassa.

3.1.2 Interação com a interface de programação no Sistema Operacional *Windows* 10

Inicialmente, a interface de programação escolhida foi o *Software* Visual Studio IDE 2013, este escolhido pois possui um *Software development kit* (SDK) que se adéqua as necessidades do projeto. O conjunto de ferramentas de desenvolvimento do *Visual Studio IDE 2013* auxilia na criação de janelas de ferramentas, comando de menu, extensões de editor e projetos de *shell*. Além de comportar as extensões da biblioteca OpenCv 2.4.9 utilizada para a realização do trabalho.

Portanto, para interação com o *Visual Studio* IDE 2013 e a biblioteca *OpenCv* 2.4.9, realizou-se a implementação de alguns algoritmos.

Acesso à câmera

O acesso à câmera utilizando as funções da *OpenCV* foi realizado na intenção de trabalhar com imagens em tempo real, imagens essas, obtidas através da câmera do notebook ou outra câmera disponível, e assim se iterar com as funções disponíveis na biblioteca. Inicialmente incluí-se as bibliotecas necessárias para utilização das funções e classes para realização da tarefa.

Em seguida, determinou-se uma variável como matriz. Esta variável matriz receberá a imagem digital, conforme foi visto na fundamentação teórica, imagens digitais são matrizes numéricas que contém a intensidade de cada *pixel*. Para este acesso utilizou-se a função *VideoCapture cap* e a condição de acesso *cap.open(0)*.

Em seguida atribuiu-se os valores da variável matriz através da imagem captada pela câmera e criou-se uma janela para retorno dos dados juntamente com seus parâmetros: nome e dimensões.

Contagem de círculos

O desenvolvimento do algoritmo para reconhecimento e contagem de círculos foi realizado no intuito de obter conhecimento da utilização de laços de repetições e contabilização de formas identificadas. Portanto, realizou-se um teste comum para se iterar da capacidade de contabilizar formas. Utilizou-se imagens para que fossem identificados círculos. As imagens a serem analisadas foram armazenadas pela função *imread()*. Após esse armazenamento, a identificação dos círculos foi realizada utilizando-se a função *HoughCircles()*, esta função verifica o formato das formas geométricas presentes na imagem e identifica as formas circulares.

Vale ressaltar que neste teste utilizou-se o método *GaussianBlur()* para minimizar a interferência dos ruídos na imagem e assim facilitar a identificação das formas desejadas. Após a identificação, realizou-se a contagem das formas circulares identificadas na imagem, conforme figura 4.1.

3.2 Métodos de identificação de perfis

Após a interação com o ambiente de programação e a biblioteca para aplicação das técnicas, realizou-se alguns testes para a definição dos métodos mais eficazes para identificação de perfil.

Método Background Subtraction

Primeiramente foi utilizado a função *BackgroundSubtractorMOG()*. Esta, realiza a subtração de imagens e exibe a variação na cena anterior. Os testes aplicados com o método foram realizados para identificar suas limitações e falhas diante a identificação de um objeto específico. Este método é o que mais necessita de processamento pois sempre está atualizando a cena para que a comparação seja realizada.

Posicionou-se uma câmera no corredor do CEFET-MG campus V e realizou a obtenção do vídeo. Os dados coletados, puderam retornar a movimentação de uma pessoa no corredor da instituição através da variação dos *frames* realizada pela locomoção da pessoa.

Observou-se que este método possui falhas quando não há movimentações de objetos, ou seja, o *frame* anterior é igual ao *frame* atual. Este fator prejudica a identificação de perfis em ambientes estáticos ou em ambiente em que a variação da cena é pouca.

Método Skin Detection

Um dos métodos também analisado foi o *Skin Detection*, este é utilizado para tratamento da imagem, retornando a segmentação da mesma por um padrão de tonalidade setado. O método pôde retornar quase totalmente a área de interesse. O que dificultou a utilização do mesmo é a configuração dos parâmetros de tonalidades de pele, esta que é configurada através dos canais de cores da imagem e devem ser muito bem empregados para que assim possa identificar pessoas de todas as etnias.

Para facilitar este processo foi realizado a criação de uma barra com limiares de cores do canal, chamada de *Threshold*, esta facilita a variação da intensidade de cada canal de cor, permitindo verificar em real time a coloração que está sendo filtrada. Para tornar a visualização da intensidade das cores, realizou-se a transição da imagem para o espaço de cores HSV auxiliando assim no processo de calibração dos filtros de cores para segmentação.

Neste método pôde-se também observar os efeitos do filtro *Blur*, este aplicado para amenizar a intensidade das tonalidades das cores, para que assim seja possível verificar o perfil desejado e aplicar os filtros de cores para segmentação. Através dos testes realizados, variou-se a intensidade do filtro *Blur* e observou-se seus efeitos no método *Skin detection*. A variação do filtro influenciou diretamente na identificação da região de interesse, realizou-se a variação do parâmetro até encontrar um limiar que fosse satisfatório para o ambiente.

Método HaarCascades

O método *HaarCascades* foi o único dos métodos não foi possível ser implementado no sistema operacional *Windows 10*. As tentativas de sua utilização foram interrompidas devido ao fato deste método possuir um padrão de reconhecimento fixo, ou seja, há um arquivo .xml com as especificações de face. A utilização do arquivo possui algumas limitações de sistema que não foram corrigidas. No entanto, foram levantadas bibliotecas para que seja possível a aquisição de dados e assim corrigir essas limitações.

3.3 Interface do usuário

3.3.1 Criação do Aplicativo

Primeiramente realizou-se a criação do *design* do aplicativo de monitoramento para acessar os dados retornados pelo dispositivo móvel. Para esta etapa, utilizou-se o *software Adobe Experience Design*. Este, possui ferramentas que auxiliam na criação de interfaces para diversos tipos de dispositivos móveis.

A acessibilidade e o fácil entendimento da interface foram os requisitos de maior importância para a construção do *design*, contendo três telas: a tela inicial que é exibida assim que o aplicativo é selecionado e duas telas que são exibidas de acordo com o ícone selecionado, uma contendo informações do programa e a outra com o ambiente de exibição das variáveis retornadas pela plataforma embutida, nos propósitos do projeto.

Após a criação do *design* do aplicativo, realizou-se o desenvolvimento do mesmo. O *design* foi criado no ambiente de integração para desenvolvedores para dispositivos Android, *software Android Studio*. A linguagem utilizada para esta etapa foi *JAVA*. O algoritmo de acesso possui a presença de classes para realizar a solicitação de acesso ao *Raspberry*, atuante como servidor das páginas com o conteúdo.

Para o desenvolvimento do aplicativo no ambiente *Android Studio*, criou-se três classes para o realizar as tarefas com os propósitos do projeto.

- ActivityMain.xml: Classe responsável pelo *layout* e ações a serem tomadas quando o aplicativo é requisitado quando selecionado.
- AndroidManifest.xml: Classe responsável por permitir o acesso *web* do aplicativo, verificando a conectividade com a internet, alem de, definir o nome e ícone que estarão disponível na tela do usuário.
- MainActivity.java: Classe responsável por definir o endereço URL que o aplicativo aponta após ser selecionado, este endereço foi definido pelo *webserver* com o IP do Raspberry: 192.168.137.107 : 5000.

Portanto, a funcionalidade do aplicativo é dada por apontar um endereço URL disponibilizado pelo servidor do *Raspberry*, fazendo-se assim a obtenção das informações.

3.3.2 Webpage

O desenvolvimento das páginas para acessar o conteúdo foi realizado em duas linguagens: $HTML$ e CSS. Três páginas foram criadas para que fosse possível que o usuário navegasse pelo aplicativo.

- HOME: Esta página é exibida quando o usuário acessa o aplicativo. O seu *layout* foi desenvolvido para que ao primeiro contato com o usuário, fosse possível o mesmo identificar facilmente as atividades que poderiam ser realizadas, para isso, alocou-se ícones para acesso as outras funções do aplicativo. Cada ícone contendo uma âncora para apontar para o arquivo .html em questão.
- SOBRE: Esta página está associada a um ícone de um livro presente na página inicial. Esta, contém os dados sobre o funcionamento do programa e seus criadores. Além de conter um ícone para retornar a página HOME.

- PESQUISA: Esta página está associada ao ícone de uma lupa presente na página inicial. Esta, contém os dados que são retornados pela plataforma embarcada através do servidor *web* dinâmico. Portanto, a variável para monitoramento é obtida através do algoritmo de visão computacional, e assim, via servidor, disponibilizada para ser visualizada frequentemente pelo aplicativo. É de grande importância ressaltar que o valor da variável a ser monitorada é reescrito frequentemente no arquivo .HTML da página em questão.

Posteriormente a criação das *webpages* realizou-se um teste de acesso as mesmas. Sendo que quando o usuário realiza a solicitação do conteúdo, mas por algum motivo de ruptura dos arquivos, estes, não podem ser acessados. Sendo assim, retirou-se o arquivo *lupa.html* do diretório de buscas e verificou-se através da interface do usuário o retorno do aplicativo.

3.3.3 Servidor *Web* Dinâmico

Diante da possibilidade de transformar *Raspberry Pi* em um *WebServer*, realizou-se a configuração do mesmo para que fosse possível disponibilizar o conteúdo aos dispositivos conectados na mesma rede. Portanto, utilizando-se estrutura *FLASK*, criou-se um ponto de acesso capacitando o *Raspberry* tornar um servidor. A escolha desta estrutura é devido ao fato de transformar o *Raspberry Pi* em um servidor *web* dinâmico. Além desta funcionalidade, a estrutura Flask é capaz de suportar diversas extensões e ser programado em linguagem Python, estes foram os fatores cruciais para a sua escolha.

Para a utilização da estruturação desejada, realizou-se a configuração do servidor *web* dinâmico para uma arquitetura de rede estrela para o sistema, conforme pode ser ilustrada pela figura 3.2.

Nesta etapa definiu-se através de classes as ação a serem realizadas quando solicitado algum dado. No caso das *webpages* há um ponteiro associado a cada ícone que faz a requisição de um determinado arquivo *.html*, este dado é encaminhado para o servidor que retorna a solicitação para o cliente *web browser*.

Outra configuração realizada é a definição do endereço e porta a serem utilizados, o *host='0.0.0.0'* determina a utilização do endereço padrão do dispositivo, no caso a utilização de seu IP 192.168.137.107. A porta utilizada foi a 5000, a escolha da porta relacionada ao hospedeiro é devido ao fato da mesma estar disponível e assim facilitar o acesso aos pontos da camada de transporte dos dados, sendo assim o servidor faz a leitura da porta e relata quaisquer erro. A estruturação e configurações do servidor, podem ser vistas no código a seguir.

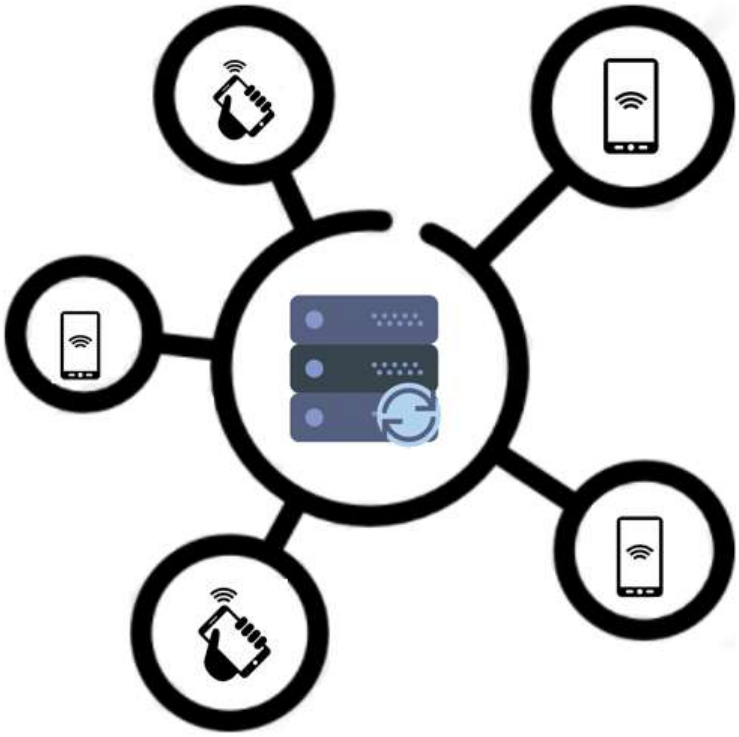


Figura 3.2: Arquitetura de rede.

```
from flask import Flask, render_template
import os
app = Flask(__name__)
app.route('/')
def main():
return render_template('home.html')
@app.route('/home.html')
def home():
return render_template('home.html')
@app.route('/lupa.html')
def lupa():
return render_template('lupa.html')
@app.route('/sobre.html')
def sobre():
return render_template('sobre.html')
if __name__ == "__main__":
app.run(host='0.0.0.0', port=5000, debug=True)
```

Posteriormente a configuração do servidor *web* dinâmico, utilizando-se da arquitetura Flask, foi possível disponibilizar o ponto de acesso ao conteúdo que é acessado pelo aplicativo. Este acesso foi verificado através de testes na conexão entre o hospedeiro e o conteúdo.

Inicialmente fez-se a conexão com a rede e realizou a requisição de acesso do hospedeiro ao servidor. Rapidamente, o servidor realizou a atividade registrando a operação,

contendo o Protocolo de Internet (IP), data e horário de acesso, a solicitação (GET) do conteúdo e o retorno do mesmo.

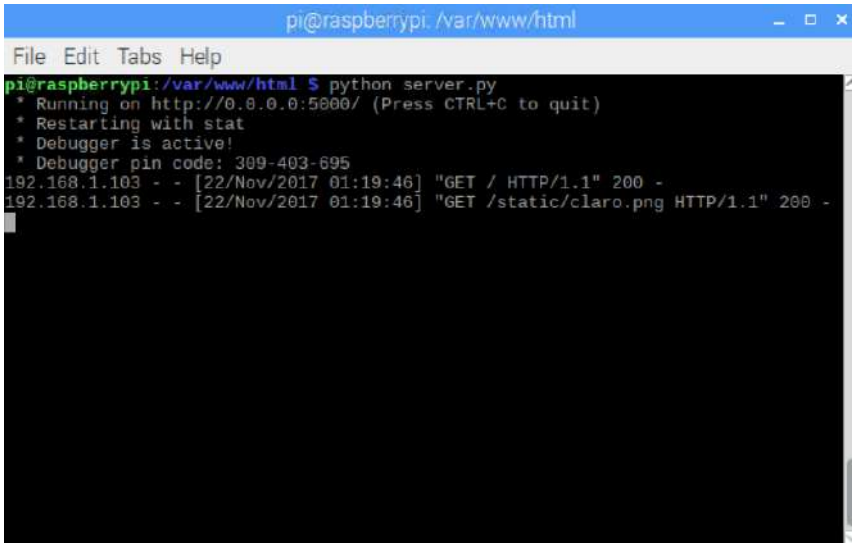


Figura 3.3: Registro do servidor ao ser acessado por um hospedeiro

A mesma operação foi realizada, no entanto, com dois hospedeiros solicitando informações ao servidor. Este, realizado para verificação de congestionamentos das vias de transição de dados. Observa-se, figura 3.4, que a navegação dos usuários não é afetada e que o conteúdo é retornado normalmente as solicitações.

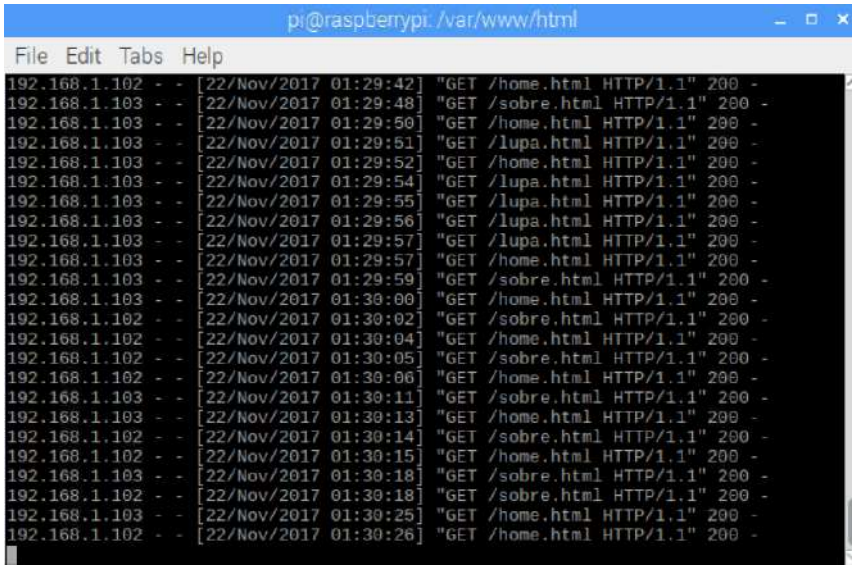


Figura 3.4: Registro do servidor ao ser acessado por dois hospedeiros

3.4 Transição do algoritmo para o sistema embarcado

A transição dos algoritmos para o periférico embarcado pode ser realizada de duas formas: Acesso remoto e programação diretamente no periférico. Por motivos de facilidade de operação, escolheu-se o acesso remoto utilizando-se o *software PUTTY* juntamente com o *software VNCView* para ter acesso a interface do dispositivo, figura 3.5.

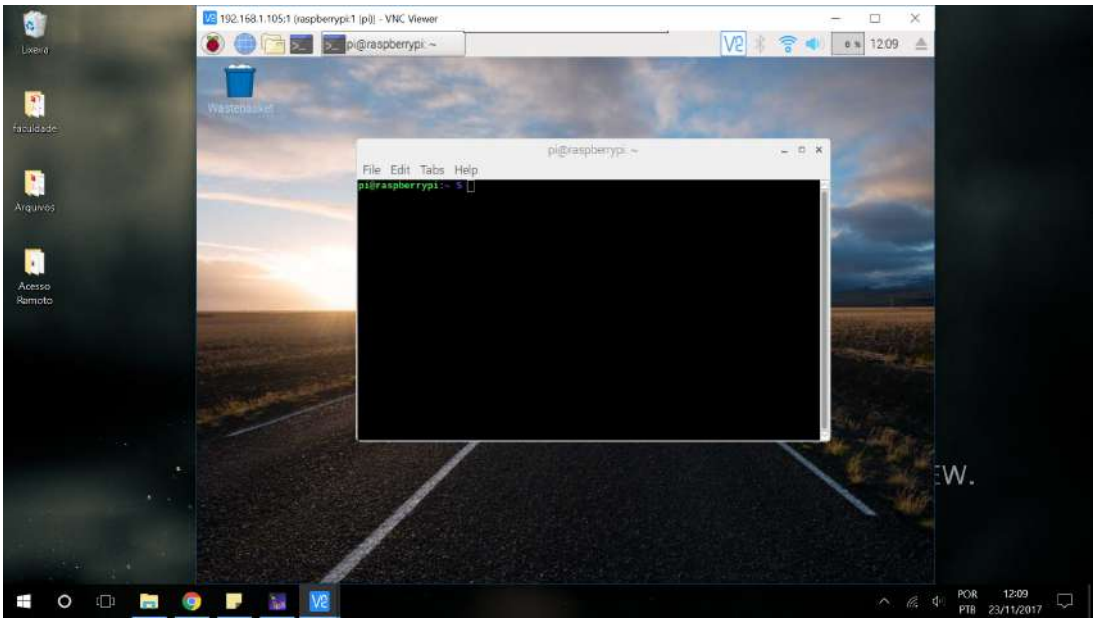


Figura 3.5: Ambiente de operação acessado remotamente.

Inicialmente, vale ressaltar que os algoritmos foram implementados para o sistema operacional *Windows* 10 foram desenvolvidos em linguagem, *C++*, enquanto no *Raspberry Pi* foram desenvolvidos em linguagem *Python*. A mudança de linguagem foi necessária devido a facilidade de se trabalhar com arquivos no formato python no sistema operacional do *Raspberry Pi*. O Sistema Operacional utilizado para a realização das atividades foi o *Raspbian Stretch*. A escolha deste, foi devido ao fato de ser a última versão disponível para instalação no periférico, sendo assim contendo as últimas atualizações.

Dando sequência as atividades, realizou-se a interação com o novo ambiente de operação, e sequencialmente, fez-se a instalação da biblioteca *OpenCV* para que fosse iniciado a transição dos métodos para a plataforma embarcada. A instalação da biblioteca da visão computacional foi realizada em um ambiente virtual. Isto, para particionar as operações realizadas no sistema embutido. A utilização da biblioteca em questão só é necessária em um dos algoritmo. Particionando-se o local de sua instalação, é possível otimizar o processamento do sistema embarcado.

Após as etapas de instalação da biblioteca de visão computacional, foi realizado a análise da capacidade computacional do dispositivo. Sendo que este foi um fator importante para a escolha do *Raspeberry*, pois para a realização das operações do projeto necessita de

um grande custo computacional, custo este que é suprido pela plataforma na realização de alguns métodos.

Portanto, a capacidade de suportar o sistema operacional *Linux* possibilita a instalação de uma biblioteca de programação e a utilização das técnicas e métodos da visão computacional.

3.4.1 Capacidade computacional

Posteriormente a instalação da biblioteca para aplicar as funções da visão computacional, realizou-se os testes para verificar o quão eficiente o dispositivo é para executar as tarefas desejadas. Portanto, realizou a compilação de cada código, referente a cada camada da identificação, para analisar o gasto computacional dos métodos. Os testes foram realizados na seguinte sequência: Subtração de cena e contorno de áreas e; Reconhecimento facial. A análise computacional dos métodos Subtração de cena e Contorno de áreas foi realizada juntamente, pois, são aplicações dependentes. Ou seja, para realizar o contorno de uma seção, deve-se haver a variação do ambiente. Portanto, quando a variação é identificada pelo método de subtração de cena, só assim, é possível aplicar o contorno. Nesta etapa de análise computacional também foi realizada a calibragem do termo condicional da segunda camada, ou seja, dos limiares da área do perfil a ser monitorado.

Após os testes dos métodos compostos pela camada de identificação de perfil serem realizados, fez-se, também, a configuração do dispositivo para tornar-se servidor *web* e foram obtidos os mesmos dados para análise. Por fim, realizou-se todas as atividades juntas e verificou-se a viabilidade de realizar o projeto no periférico e a sua capacidade de processamento para o propósito desejado.

3.4.2 Disposição das camadas algorítmicas para identificação do perfil

A disposição dos algoritmos, caracterizando-se as camadas de filtros, foi disposta para a redução de processamento do periférico, sendo que, a ativação da camada, somente ocorre se passada da camada anterior.

Primeira Camada

A primeira camada, subtração de cena, é responsável por detectar a variação no ambiente. Quando a camada é iniciada, realiza-se a captura do ambiente e a cena é armazenada.

Esta, é tomada como referência para os *frames* conseguintes, fazendo-se assim, com que qualquer variação do ambiente seja exibida.

Em um dos testes da seção 3.2 pôde-se observar um lapso na função *backgroundSubtractionMOG2()*. Como retratado na seção, quando o objeto fica estático, o mesmo não é exibido pois a função faz a subtração *frame* à *frame*. Esta questão é resolvida na primeira camada, pois no momento em que é setado uma cena referência, qualquer objeto que não esteja na cena é exibido.

Segunda Camada

Após a identificação de movimentação no ambiente, a segunda camada é ativada. Esta realiza o registro da área de variação da cena de referência. Sendo assim, a primeira camada retorna os *pixels* que sofreram variações significativas e a segunda camada realiza o contorno e cálculo da área desta variação.

Uma estrutura condicional foi implementada na segunda camada para que filtrasse ruídos como: quedas de objetos, movimentações causadas pelos ventos e outros fenômenos.

Esta estrutura condicional permite que a terceira camada somente seja ativada se a área de contorno estiver dentro de limiares de valores de áreas configurados juntamente ao procedimento de instalação do sistema embarcado.

Terceira Camada

A terceira camada, quando ativada, realiza a detecção facial da área filtrada pela camada de nível dois. Ao fim desta a variável de monitoramento é atualizada para que seja possível ser monitorada pelo aplicativo.

A variável de monitoramento recebe três valores de retorno:

- Ambiente vazio: Esta condição é enviada a variável quando não há movimentação no ambiente dentro das condições dos limiares, ou seja, a detecção de movimento na primeira camada e o filtro condicional da segunda camada não estão ativados.

- N Pessoas no Ambiente: Esta condição é enviada a variável quando a terceira camada é ativada, assim, realiza a contabilização das N's faces e enviada a variável.

  O modelo de variáveis disposta não pode retornar o intermediário da segunda camada e terceira. Sendo assim, realizou-se também a disposição das seguintes variáveis para realizar o retorno da variável na condição entre a segunda e terceira camada.

- Possível Pessoa no ambiente: Esta condição ?e enviada a variável quando há movimentação no ambiente dentro das condições dos limiares, no entanto, não há a identificação de face.

3.5 Projeto Estrutural, Materiais e Montagem

Buscando um posicionamento fixo e visando a estabilidade do sistema, realizou-se o desenvolvimento da estrutura mecânica de forma a posicionar o sistema em um local que pudesse capturar a área de interesse da melhor forma. A estrutura é composta por duas partes: Acoplamento para a câmera e; *Case* para o *Raspberry*, ambos projetados de forma compacta e que não intervisse na capacidade de operação dos mesmos.

Acoplamento Câmera

O acoplamento da câmera é composto por quatro partes. Estas possibilitam o posicionamento ideal da mesma, pois oferece uma mobilidade da lente da câmera, podendo assim ser posicionada corretamente para obtenção da região de interesse, o modelo da câmera pode ver visto na figura 3.6.

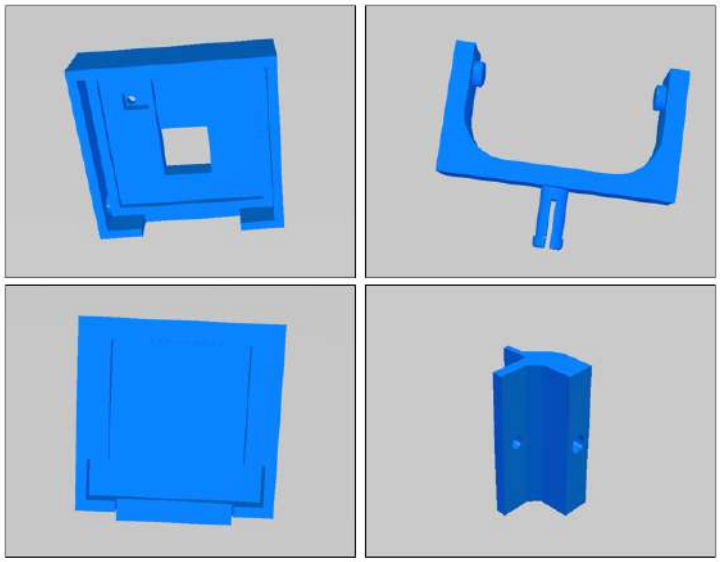


Figura 3.6: Representação do suporte de acoplamento da câmera.

Case Raspberry

A *case* para o *Raspberry* foi estruturada para que não atrapalhasse a circulação de ar no periférico e assim não dificultasse a refrigeração dos componentes. A estrutura pode ser vista na figura 3.7.

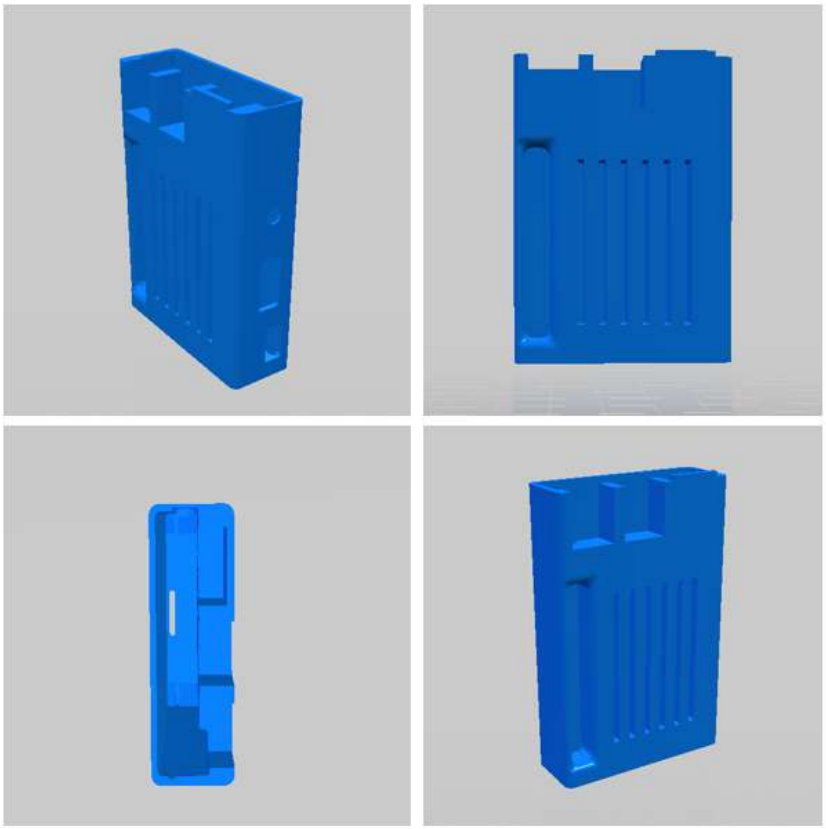


Figura 3.7: Representação do suporte para o Raspberry.

Após ser projetado, os acoplamentos foram impressos por uma impressora $3D$ utilizando-se técnica de prototipagem rápida e assim realizados testes de posicionamento e estabilidade. O material utilizado para o processo foi o filamento ABS (*Acrilonitrila butadieno estireno*). A escolha do mesmo, foi devido a sua elevada resistência a impacto e facilidade de produção para os dimensionamentos da peça. Ao fim da prototipagem do componentes do sistema, realizou-se assim a sua instalação para verificar suas características de estabilidade e posicionamento.

3.6 Análise do protótipo viável

O produto viável consiste no sistema final do projeto. Este, após a validação de seus componentes e métodos separadamente, foi preparado e submetido ao estudo de sua

viabilidade. A figura 3.8 demonstra a arquitetura final do projeto, como cada etapa foi disposta e as conexões necessárias para seu funcionamento.

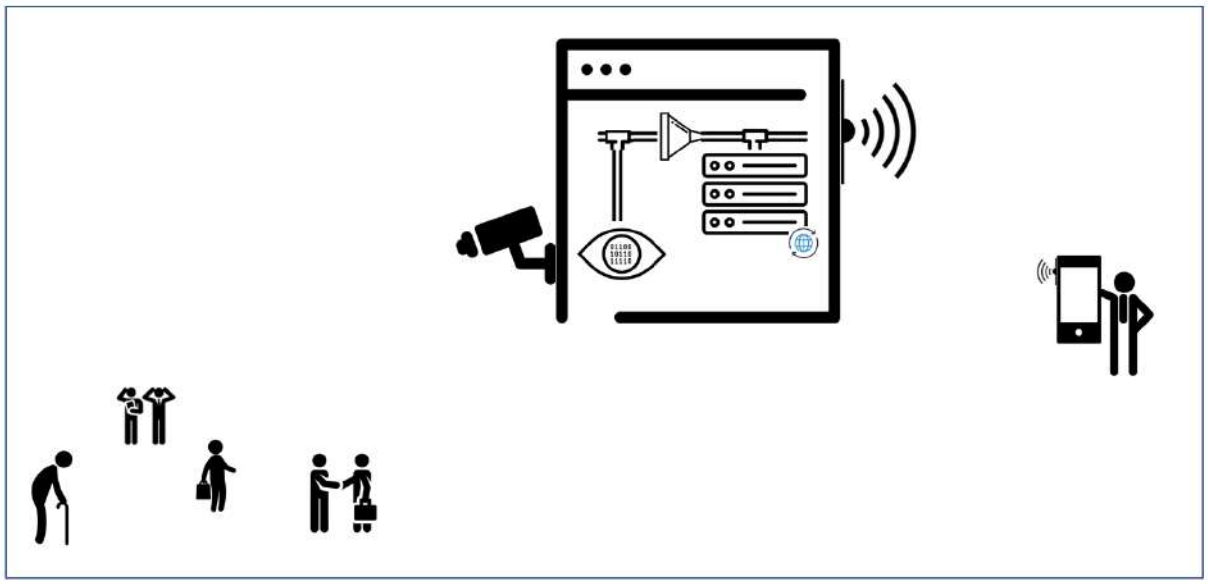


Figura 3.8: Representação do Protótipo viável.

Ao fim de todas etapas de projeto, desenvolvimento, análise e construção realizou-se a verificação de sua viabilidade quanto a tempo de resposta ao monitoramento e o custo computacional correspondente a implementação.

O dispositivo mobile utilizado para o teste foi o *Galaxy-S6-edge*. Vale ressaltar que para a utilização do aplicativo desenvolvido, o *smartphone* deve possuir um sistema operacional *Android*.

Primeiramente, realizou-se o posicionamento do sistema embarcado para fazer o monitoramento da área de interesse. Em sequência, conectou-se o *smartphone* a rede interna, para que a comunicação entre os dois pudesse ser estabelecida.

Posteriormente, a execução dos algoritmos foi realizada, primeiro o algoritmo de visão computacional e em seguida o do servidor.

Com o funcionamento da plataforma embarcada, selecionou-se o aplicativo para verificar a velocidade de retorno da variável ao usuário.

Com os resultados do desempenho do protótipo viável deu-se sequência ao mesmo procedimento de teste, no entanto, com ausência da terceira camada do algoritmo de reconhecimento de perfil.

3.7 Orçamento

Realizou-se o levantamento e dimensionamento dos periféricos necessários para o desenvolvimento do projeto, além dos

Tabela 3.3: Orçamento estimado do projeto.

Equipamento	**Valor**
Raspberry Pi 3B	R$189,00
Câmera	R$ 65,00
Servidor	Gratuito
Biblioteca	Gratuito
Aplicativo	Gratuito
Insumos	R$20,00
Total	R$274,00

Capítulo 4

Resultados parciais e finais

A partir deste capítulo são apresentadas as descrições dos projetos, testes e experimentos realizados, acompanhados dos respectivos resultados dos testes realizados para o desenvolvimento do trabalho.

4.0.1 Aplicação de técnicas de visão computacional

Contabilização de círculos

Pode-se observar através da Figura 4.1, que a contagem no número de círculos presentes na imagem foi realizada de forma correta. O retornado do contorno em vermelho e demarcação do centro com o ponto verde. Este método realizado para análise de atividades clássicas de identificação de forma geométrica específica e a contabilização da mesma. A realização da atividade é de grande valia pelo fato da contabilização das formas, esta que é uma das atividades do projeto final. Sendo assim, a prática dos laços de repetição ajudaram para o entendimento dos conceitos de contabilização.

Efeitos da função Blur

A Figura 4.2 mostra os efeitos da função *blur* com algumas variações de sua intensidade. A imagem superior esquerda é a original. Superior direita foi submetida a aplicação *blur* com intensidade *size* (2,2). Inferior esquerda demonstra o efeito *blur* com intensidade *size*(10,10). Inferior direita possui a aplicação do *blur* com intensidade *size*(20,20). A variação da intensidade da função foi realizada para ver o quão variante são os efeitos do filtro de forma a observar os limiares de tratamento da imagem. Foi possível observar a partir de qual intensidade pode haver interferência no entendimento de formas presentes nas imagens.

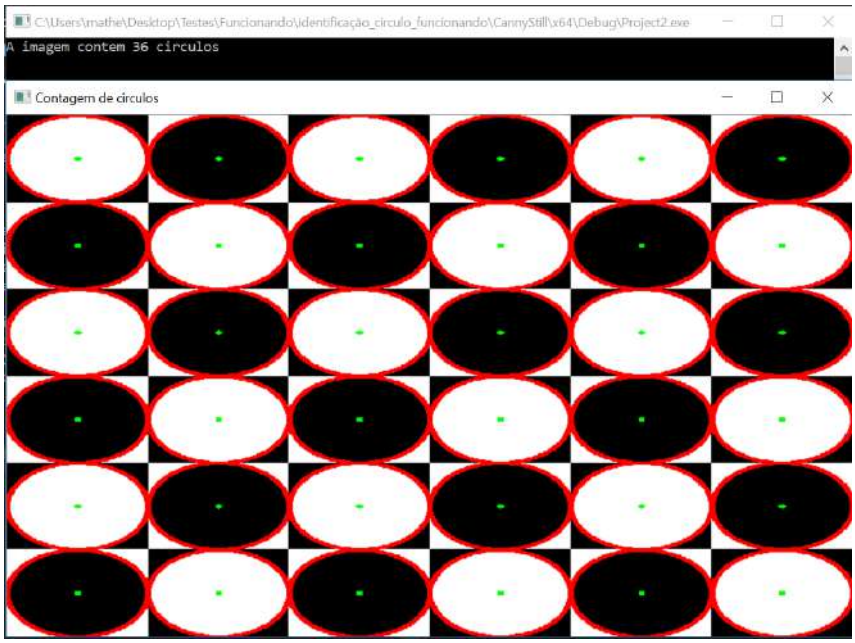


Figura 4.1: Resultado do teste de contabilização de círculos.

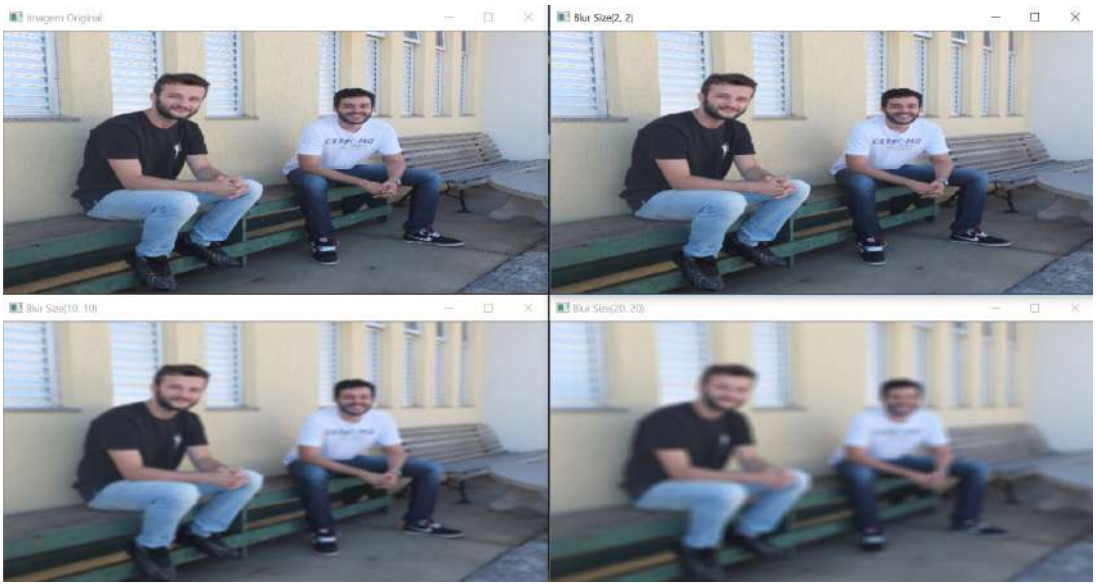


Figura 4.2: Resultado do efeito do filtro blur em uma imagem.

Skin detection

A Figura 4.3 demonstra o quão eficiente é a segmentação da função *inRange()* quando calibrada corretamente em um ambiente com parâmetros fixos. Juntamente com outros conceitos de troca de espaço de cor, contorno e segmentação podem realizar a identificação com perfeição. A identificação de perfis utilizando este método pode conter ótimos resultados quando submetido em um ambiente estável.

A figura está particionada em quatro imagens. A superior esquerda contém a aplicação do filtro *blur*, este aplicado para amenizar a intensidade das tonalidades das cores,

para que assim seja possível verificar o perfil desejado e aplicar os filtros de cores para segmentação. Na região superior direita da figura, contém a imagem com aplicação do filtro *blur* no espaço de cores HSV. A realização da transição do espaços de cores foi realizada para auxiliar a análise das intensidades de tonalidade, valores e saturação. Estes parâmetros foram utilizados para realizar a segmentação da imagem. A imagem inferior esquerda contém o contorno da segmentação, este realizado nos pontos de alta variação na intensidade dos *pixels* analisado em sua vizinhança, considerados como borda. A imagem inferior direita representa o resultado da segmentação final. Esta realizada utilizando filtros das tonalidades observadas na figura no espaço de cores HSV, portanto a segmentação foi realizada utilizando os limiares das cores vermelho, verde e azul.

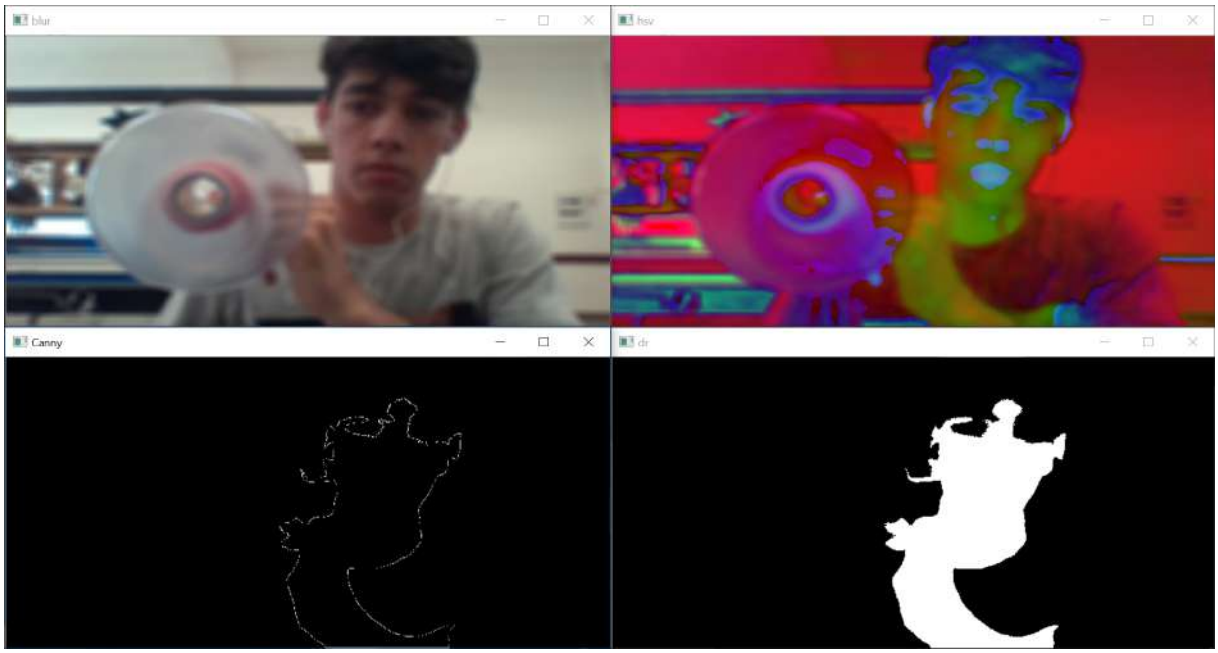


Figura 4.3: Resultado da aplicação do método skin detection.

Método Background Subtraction

Observa-se através da Figura 4.4 a identificação através da variação do *frame*, esta variação acontece devido a movimentação da pessoa pelo corredor. As imagens obtidas na figura foram tiradas do vídeos realizado no ambiente, portanto a variação dos *frames* pode ser analisada pelo método.

A imagem superior esquerda mostra o ambiente sem nenhuma pessoa, ou seja, sem variações na intensidade dos *pixels*. A imagem superior direita demonstra o instante que a pessoa entra no campo de visão da câmera. Assim, através da subtração de *frames* pôde-se observar a região de interesse, no caso do projeto em questão, a presença de um indivíduo.

A imagem inferior esquerda mostra o acompanhamento da região de interesse juntamente com a locomoção do indivíduo. Este fato é interessante pois ressalta o fato da atualização do chamado *frame* atual. Pode-se dizer então que na próxima captura o *frame* atual será o *frame* de comparação. Portanto a atualização destes é fundamental para realização do processo de subtração.

A imagem inferior direita, demonstra a pessoa no sentido contrário, podendo assim ser identificado em qualquer perfil, desde que seja realizada a movimentação na região de interesse.

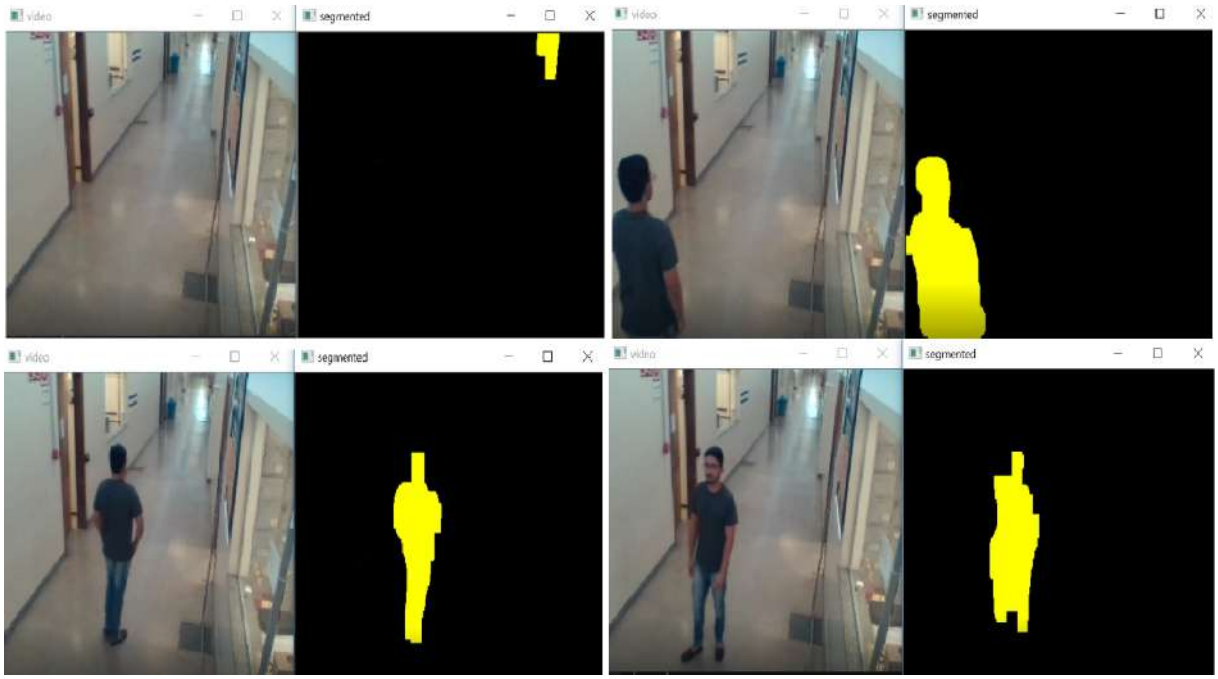


Figura 4.4: Resultado do design do ícone do aplicativo de utilização do usuário.

4.1 Interface do Usuário

Aplicativo

Os resultados da criação do aplicativo foram conforme o previsto. A interface possui um *design* limpo, permitindo que o usuário no momento do acesso consiga filtrar as informações de operação que podem ser realizadas. Em primeira instância o ícone do aplicativo, figura 4.5, é bem intuitivo, contendo o símbolo de uma 'lupa' seguido do nome do mesmo, 'Monitoramento'.

A escolha da coloração verde tanto para o ícone quanto para as páginas de navegação é devido aos conceitos de cromoterapia que indica que esta causa sensações de equilíbrio e harmonia.

As páginas de navegação pelo aplicativo são intuitivas, impedindo que o usuário tenha

Figura 4.5: Resultado do ícone na tela do dispositivo móvel.

dificuldades ao escolher o que deseja. A figura 4.6 demonstra a tela inicial quando o aplicativo é selecionado.

Já as figuras 4.7 e 4.8 demonstram, respectivamente, o conteúdo quando selecionado o ícone 'Livro' ou 'Lupa'.

Webpage

Para análise de erro de acesso ao conteúdo de uma *webpage*. Realizou-se a extração da página lupa.html do diretório de pesquisa e verificou a resposta do servidor ao aplicativo. Conforme pode ser analisado na figura 4.9 o servidor retorna ao aplicativo um alerta de erro gerado pela extensão do *Jinja2* da estrutura *flask*. No alerta consta que o arquivo em questão não foi encontrado e assim algumas funções não podem ser executadas.

4.2 Análise computacional dos algoritmos na plataforma embarcada

Com a realização dos testes das camadas algorítmicas e do servidor *web*, foi possível verificar o custo computacional da atuação de cada um e também em conjunto, caracterizado como o funcionamento final do sistema.

Figura 4.6: Página inicial do aplicativo.

Inicialmente, sem a operação de quaisquer atividade extra, o sistema comporta com as seguintes informações presentes na figura 4.10. Nota-se que a utilização do CPU é aproximadamente de 1% e a memória utilizada é de 148MB.

Subtração de cena e Contorno de Área

Ao realizar o teste de subtração de cena juntamente com o contorno de área, pôde-se observar a necessidade computacional na implementação dos dois métodos. A figura 4.11 mostra que há a necessidade de 30% do CPU e uma memória correspondente a 38MB. Na imagem pode-se observar também as áreas dos contornos presentes na imagem.

Com a obtenção dos valores da área dos contornos, realizou a instrumentação do método de forma a não exibir áreas que não sejam correspondentes a do perfil desejado. Sendo assim, áreas que não estejam dentro da margem, são consideradas ruídos. Como pode ser observado na figura 4.12, houve uma variação na posição da cadeira, para simular um ruído, este foi inferior a área do perfil desejado e assim ignorada.

A camada filtro de área implementada é de suma importância para evitar sinalizações de perfis indesejados. Estes podem ser, movimentação de algum objeto causada pelo vento, movimentação de um ventilador ou até mesmo a passagem de algum animal pelo

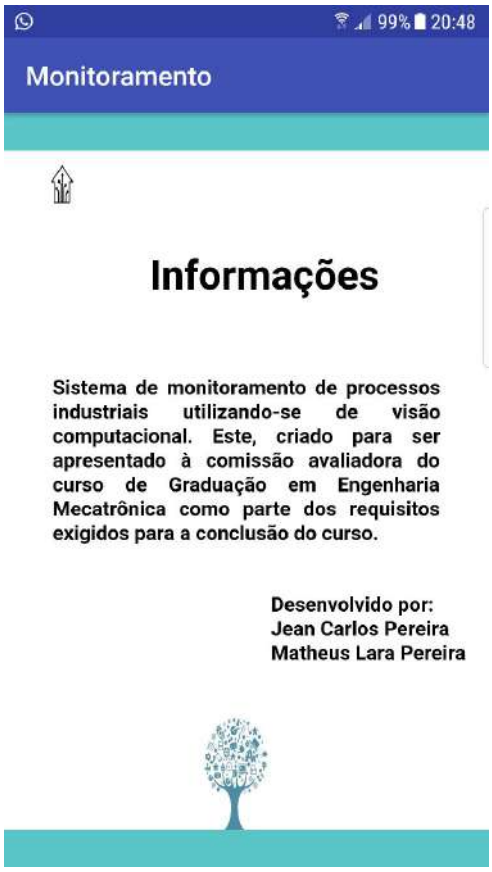


Figura 4.7: Página que contém informações do aplicativo.

ambiente.

Detecção Facial

O algoritmo de detecção facial possui seu funcionamento baseado em parâmetros disponíveis em um arquivo *.html*. Ao fazer a verificação, ele analisa pontos fundamentais para a verificação da imagem. Caso estes pontos correspondam aos parâmetros de consulta, ele identifica a face. Devido a essa consulta aos parâmetros é o algoritmo que possui maior consumo de processamento, pois necessita maior capacidade de computação para alcançar seu objetivo. A figura 4.13 demonstra o custo computacional deste algoritmo.

Servidor *web* dinâmico

Duas análises foram realizadas com o servidor dinâmico. A primeira foi verificada quando o servidor é disponibilizado, sem nenhum acesso para requisição de dados. A figura 4.14 mostra que não há alteração de processamento do CPU. No entanto, esta atividade requer uma memória de 17MB.

A outra análise, realizada quando o servidor é utilizado na sua ação máxima, ou seja, quando o usuário hospedeiro quer visualizar o retorno das variáveis. Esta ação solicita

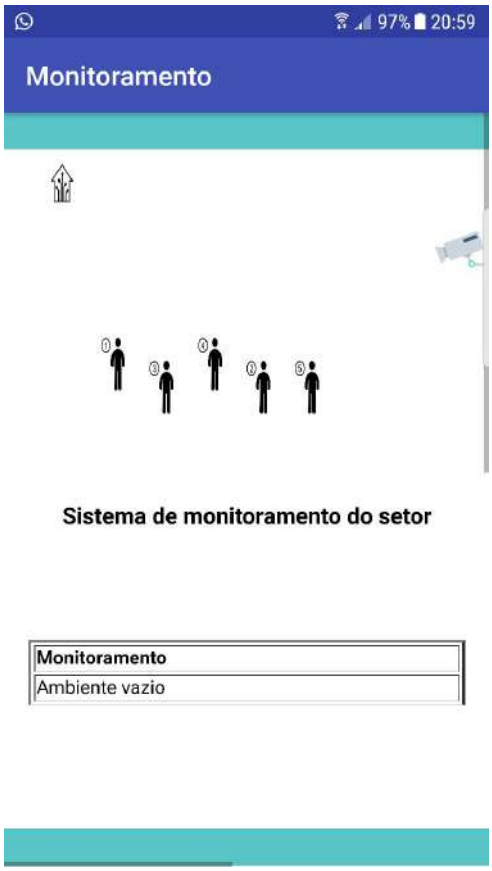


Figura 4.8: Página de acesso ao monitoramento.

frequentemente o valor da variável ao servidor, sendo assim, o trânsito de informações é frequente.

Observa-se pela figura 4.15 que esta ação requer tanto processamento do CPU quanto memória de operação. Nota-se que o processamento do CPU e a memória são, respectivamente, 5% e 25MB a mais do que o convencional.

4.3 Acoplamento Câmera

A figura 4.16 demonstra o resultado da impressão do acoplamento para o módulo da câmera utilizando-se a impressão 3D, caracterizando a prototipagem rápida.

A estrutura possui dois graus de liberdade possibilitando a configuração do posicionamento e angulação da lente da câmera de forma a capturar a região de interesse. Este posicionamento pode ser alterado conforme a necessidade do sistema, no entanto, parâmetros como os limiares de área devem ser restabelecidos caso haja a modificação do posicionamento da câmera. pois a perspectiva do ambiente muda com a alteração do sistema.

A atuação do servidor, controlando o acesso ao conteúdo é realizado de forma eficiente. A solicitação dos dados e a transição dos mesmo até o aplicativo são monitorados

Figura 4.9: Resposta do servidor ao aplicativo na ausência da *webpage*.

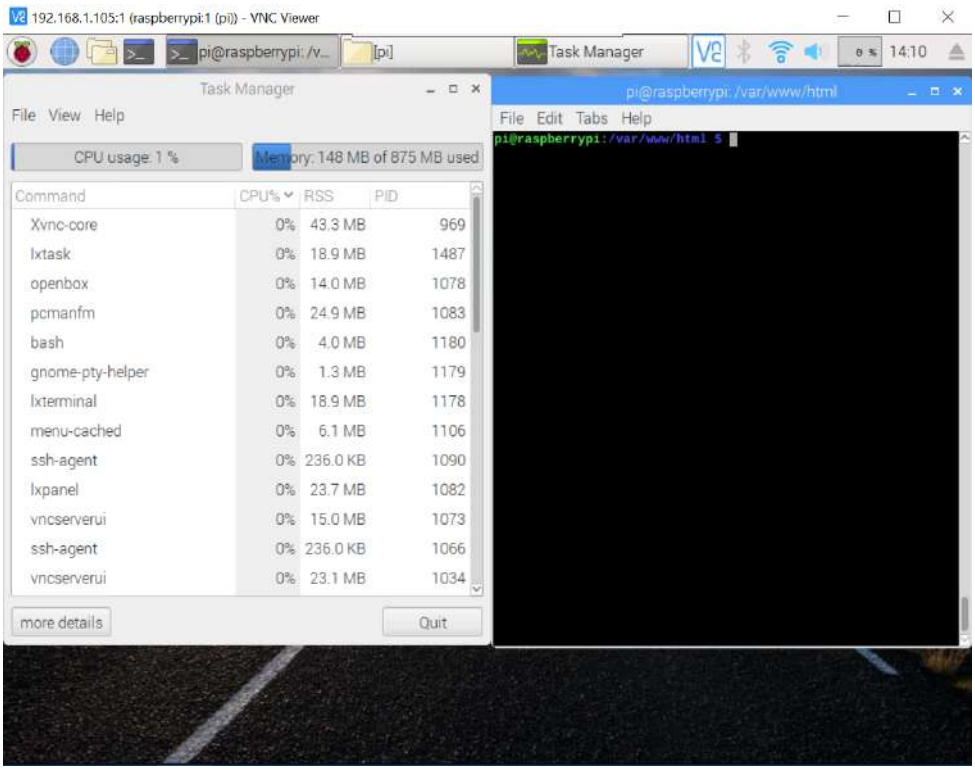


Figura 4.10: Processamento do sistema embarcado sem nenhum atividade.

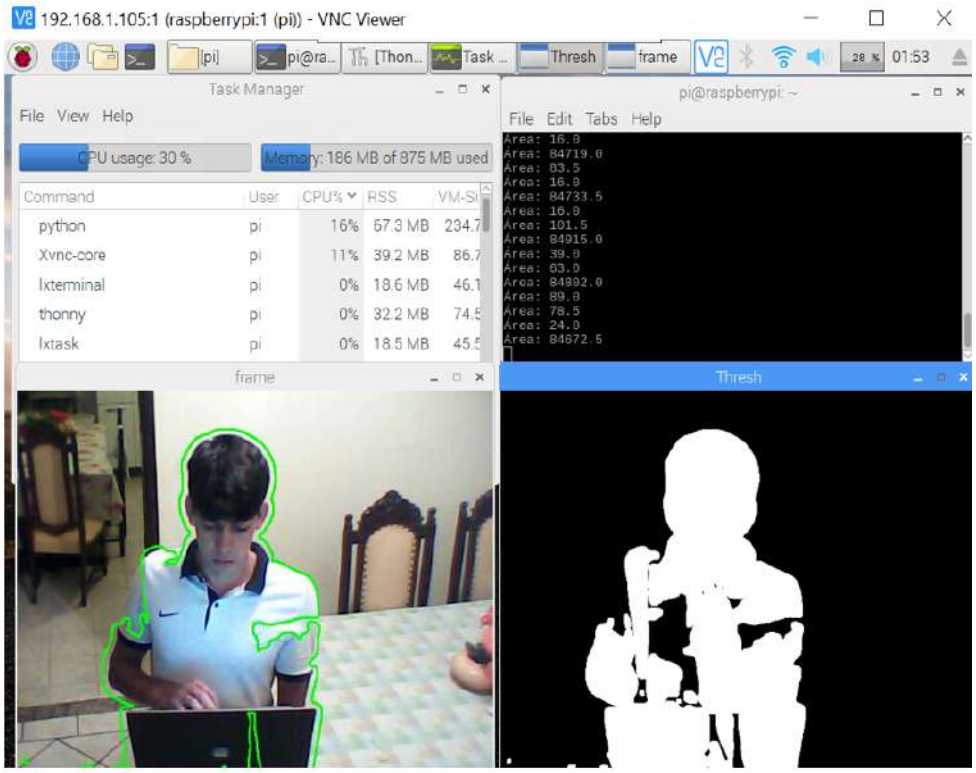


Figura 4.11: Análise computacional dos métodos de Subtração de cena e contorno de área.

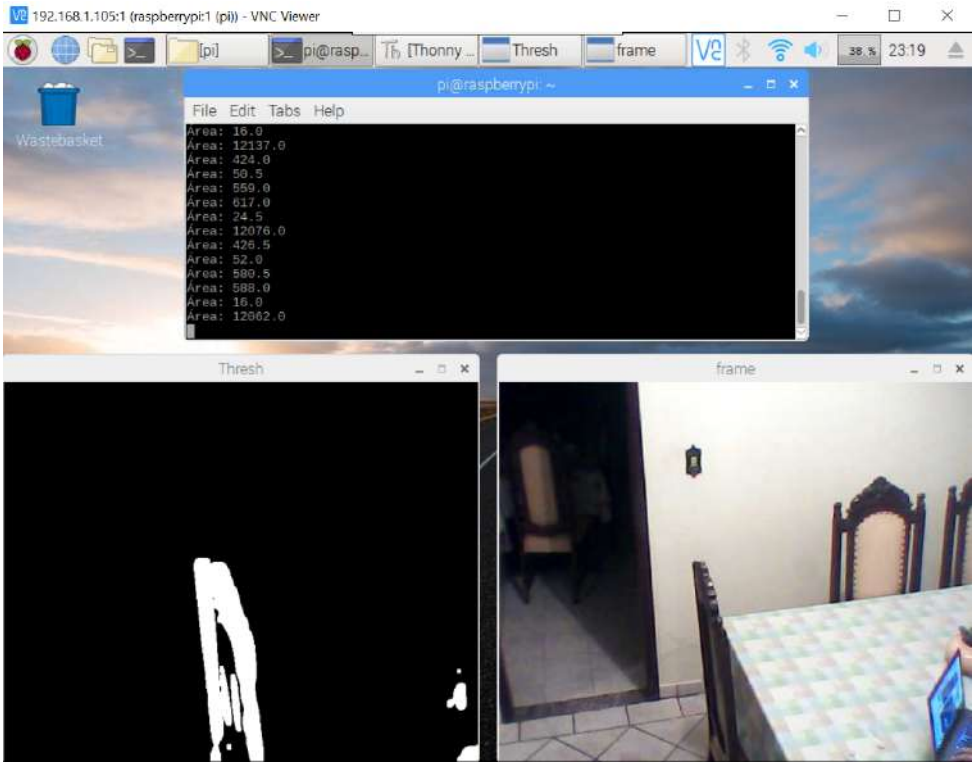


Figura 4.12: Inserção de ruído no ambiente para análise da estrutura condicional.

e manipulados pelo servidor de forma correta, garantido a fluidez das informações. A segurança, mesmo que apenas pela porta de acesso, está sendo empregada, visto que ao

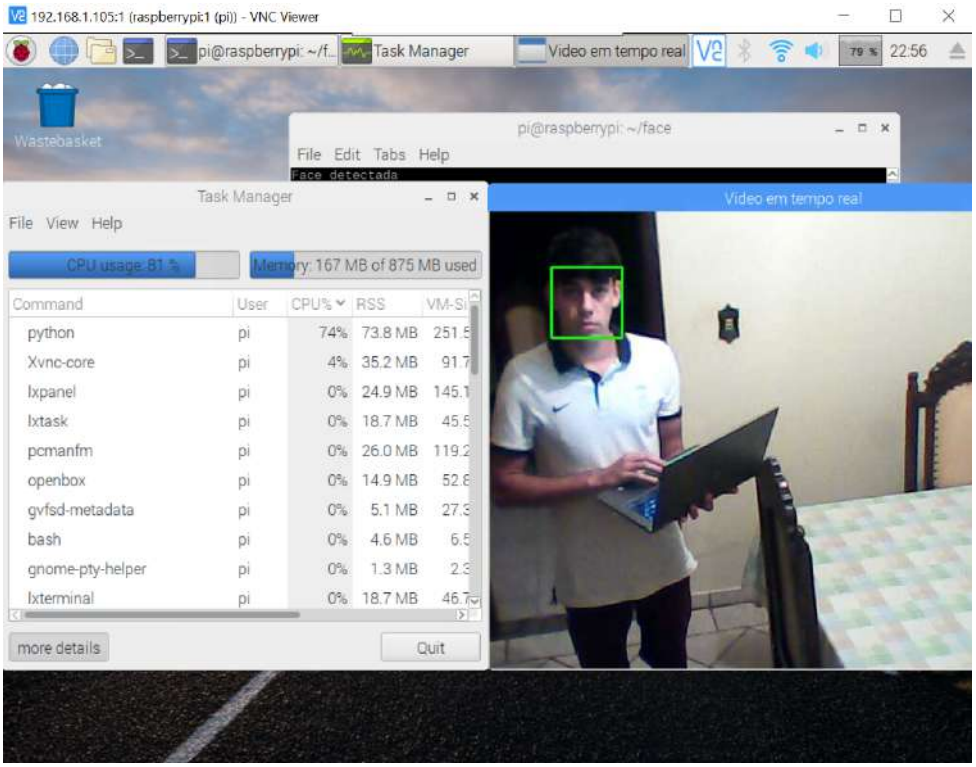


Figura 4.13: Processamento do sistema embarcado sem nenhum atividade.

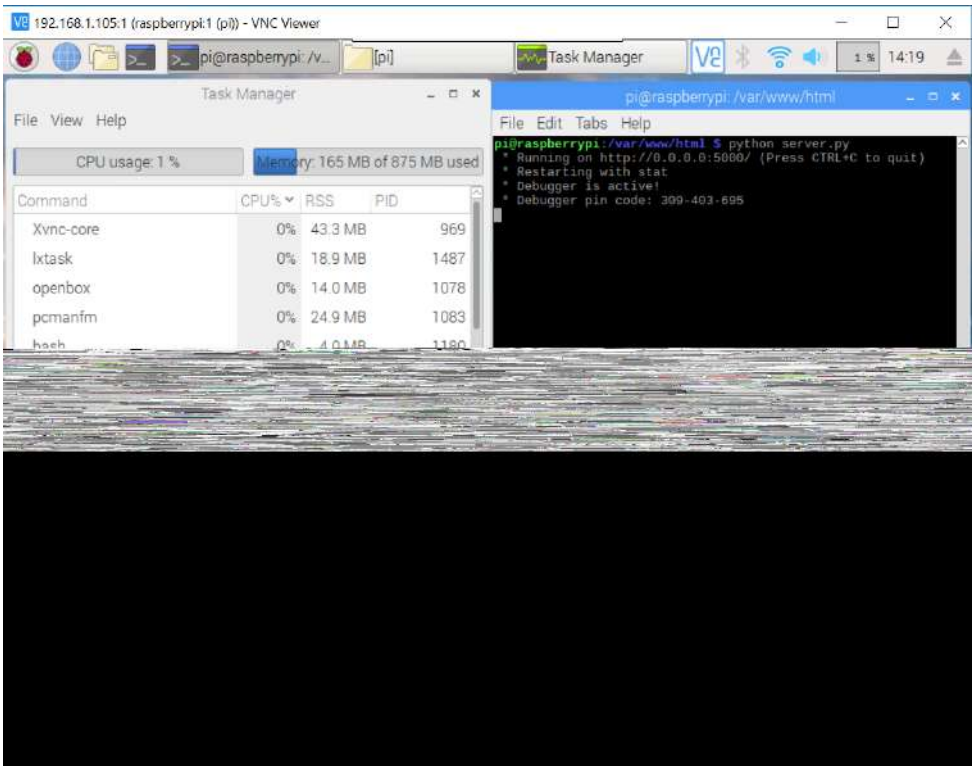


Figura 4.14: Processamento do sistema embarcado sem nenhum atividade.

tentar acessar o conteúdo a por um rede externa, o mesmo não é exibido.

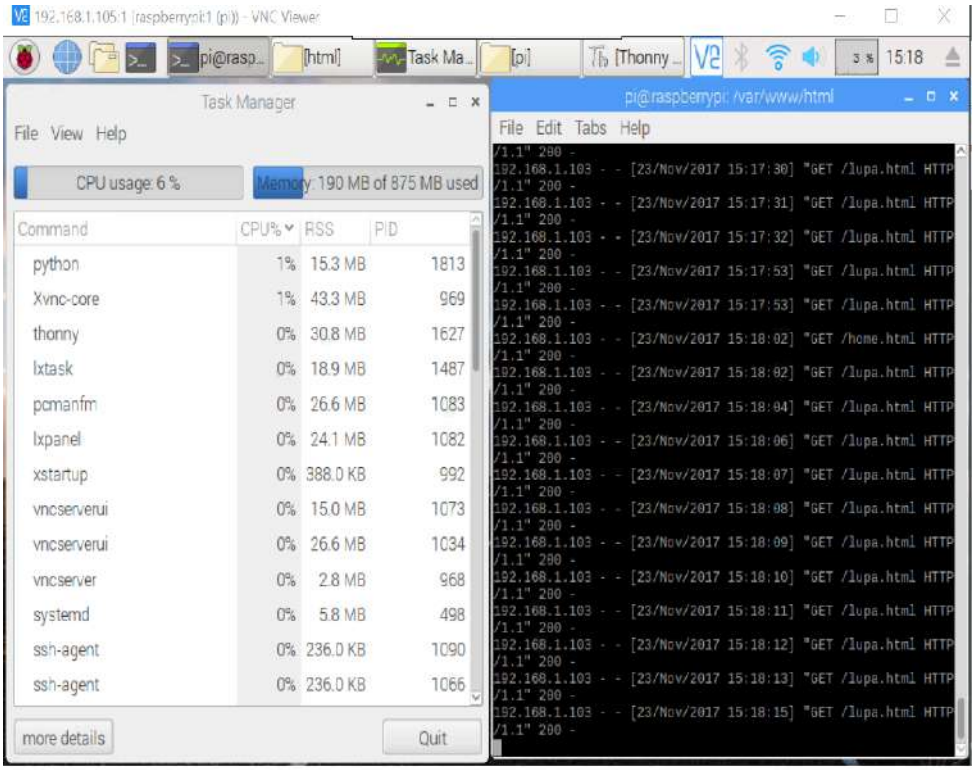


Figura 4.15: Processamento do sistema embarcado utilizando a requisição frequente de dados do servidor.

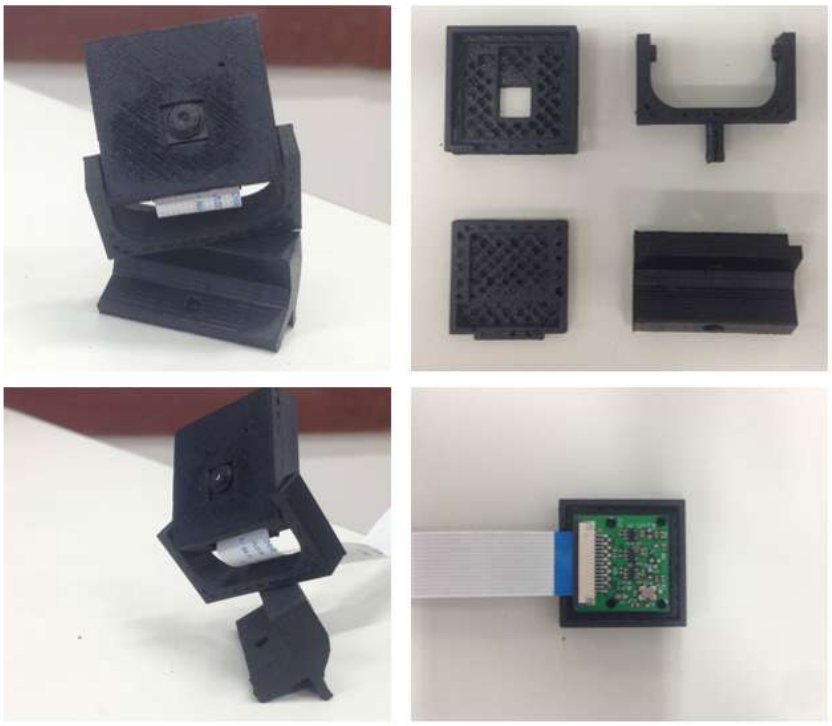


Figura 4.16: Resultado do protótipo de acoplamento da câmera

4.4 Protótipo viável

Ao fim do projeto e desenvolvimento, pôde-se então analisar o funcionamento do sistema. Em primeira instância realizou-se a instalação do sistema embarcado, figura 4.17. Esta com um posicionamento e uma área para obtenção de uma visão ampla do ambiente de interesse.

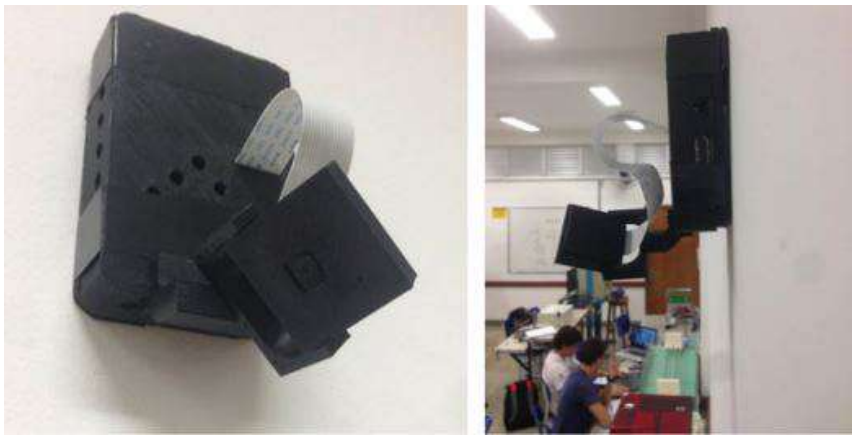


Figura 4.17: Resultado da instalação do sistema embarcado

O sistema encontra-se num posicionamento que é possível aplicar as três camadas do algoritmo e está estavelmente alocado, sem possíveis influências de ruídos. Em sequência foi feito análise e a conexão do sistema embarcado e do *smartphone* a rede interna. Este procedimento é fundamental para que o monitoramento do sistema seja possível através do aplicativo. A figura 4.18 demonstra o nome dos dispositivos conectados a rede os seus respectivos Protocolos de Internet e seu Endereço Físico (MAC).

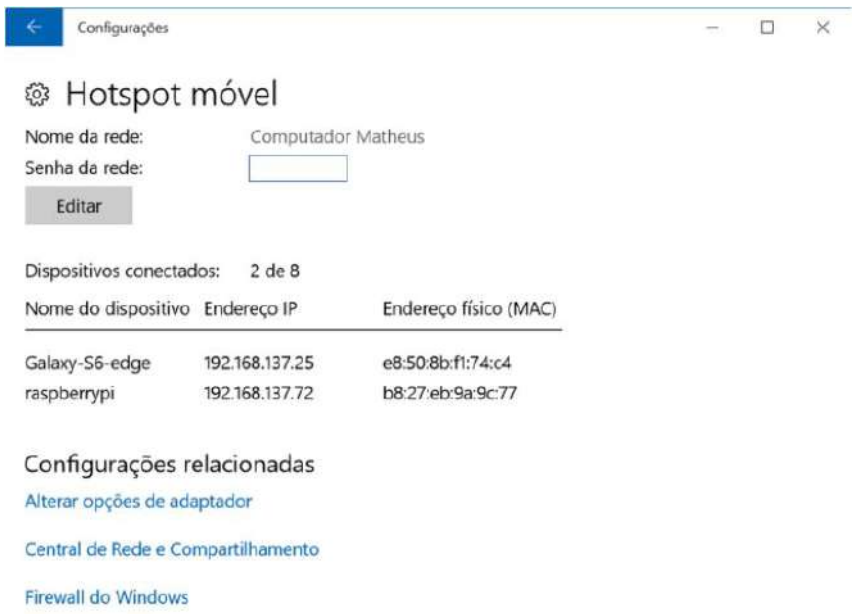


Figura 4.18: Compartilhamento de rede, *smartphone* e sistema embarcado

Portanto, após esses passos, pôde-se utilizar o aplicativo para verificar sua interação com o sistema embarcado. A figura 4.19 demonstra o funcionamento do monitoramento da quantidade de pessoas no ambiente.

Figura 4.19: Resultado do teste realizado para análise da comunicação entre o sistema embarcado e o aplicativo.

Na figura 4.20 pode analisar a aplicação do segundo algoritmo, com o retorno da variável com o sistema na condição entre as camadas dois e três.

Figura 4.20: Resultado do teste com a variável da condição entre as camadas dois e três.

Após os testes, pôde-se notar que a comunicação do dispositivo embarcado e o aplica-

tivo utilizando todos os parâmetros para funcionamento do protótipo está sendo estabelecida corretamente.

No entanto, ao realizar esta atividade de monitoramento, o sistema possui um leve *delay* para retorno das informações ocasionado pelo intensa computação do sistema embarcado. A implementação do algoritmo com as três camadas exige um alto processamento de dados do periférico, podendo comprometer a viabilidade da aplicação do sistema em ambientes que possuem uma dinâmica muito rápida. No mais, não impede que este seja aplicado em outros sistemas

Capítulo 5

Considerações Finais

O presente capítulo contém as considerações a respeito de todas as ações tomadas para o projeto e do desenvolvimento do trabalho. Serão apresentadas as propostas de continuidade, além do cronograma das atividades. Vale ressaltar que todos as pessoas presentes nas imagens para a an´alise dos resultados estão consentidas com a utilização das mesmas.

5.1 Conclusões

O trabalho apresenta as problemáticas em sistemas de monitoramento e as vantagens da utilização da visão computacional tanto na indústria quanto no nosso cotidiano para solucioná-las. Estes sistemas de computação visual em vídeos contêm informações relevantes, que podem, de certa forma, serem utilizadas em diversos ambientes. Outro conceito abordado é a utilização de sistemas embarcados para realização de atividade de monitoramento remoto. Neste contexto, notou-se a necessidade do projeto e desenvolvimento de um sistema embarcado capaz de identificar variáveis específicas em ambientes, utilizando-se de visão computacional, abordando assim, os mesmos conceitos de *smartcam*.

Visto isso, para realização do desenvolvimento do trabalho realizou-se a construção de uma metodologia composta por duas etapas: Estudo do projeto e; Projeto conceitual. Estas, foram particionadas de forma a organizar todo o conteúdo a ser desenvolvido. Diante disso, deu-se iniciativa ao levantamento do conteúdo literário para desenvolvimento do sistema, presente no capítulo dois, titulado Revisão de Literatura. Neste tópico, conceitos fundamentais como processamento de imagem, imagem digital, espaços de cores e visão computacional foram abordados para levantamento de conceitos e assim dar sequência ao projeto.

O desenvolvimento do trabalho deu-se restritamente aos passos da metodologia e seguindo os conceitos literários. Estes, foram fundamentais para o desenvolvimento das atividades e assim gerar resultados que enfatizam a viabilidade do projeto. Portanto, é

válido ressaltar, tendo em vista a finalidade do projeto, que os resultados foram satisfatórios perante a eficiência dos métodos utilizados. O aplicativo desenvolvido apresentou o comportamento esperado, com um design moderno e interativo, estabelecendo a conexão com o servidor corretamente.

O servidor desenvolve suas atividades corretamente, fazendo com o que o acesso dos hospedeiros aos dados sejam monitorados com as informações necessárias para certificar quem estão solicitando-os.

Com a análise do custo computacional, pôde-se observar que as técnicas de camadas para o processamento da imagem no dispositivo embarcado necessitam de um elevado processamento. Custo esse suprido, com restrições, pelo periférico computacional. Restrições estas que não permitem que o sistema seja aplicado em ambientes que possuem um fluxo rápido de pessoas, pois, a terceira camada do algoritmo requer um elevado nível de processamento, fazendo com que ocorra um atraso na gerência das informações. Uma questão fundamental para a otimização do processamento é a retirada da terceira camada, visto que esta requer maior poder de computação. No entanto, o ambiente poderá conter apenas o fluxo de pessoas, sendo assim a aplicação das duas camadas iniciais, retornaram de forma mais rápida a quantidade no ambiente através dos contornos das formas.

Outro fato interessante a ser ressaltado foi a abordagem dos conteúdos de computação para implementação dos algoritmos de identificação de perfil, desenvolvimento do servidor e aplicativo. E os conteúdos de eletrônica para realizar o processamento da digital das imagens, tratamento de sinais e aplicação de filtro em ruídos, foram de fundamental importância para expandir o conhecimento, enriquecer a formação, fomentar o perfil engenheiro e desenvolver o perfil pesquisador. Agregando assim a formação acadêmica, visto que muitos conceitos não estavam presentes na grade curricular.

À vista do que foi abordado, os resultados do protótipo consentem com o esperado, alcançando-se função crítica com sucesso através do triunfo dos objetivos geral e específicos. Portanto, com os resultados pode-se afirmar que o protótipo age de maneira satisfatória correspondendo minimamente as expectativas de projeto.

5.2 Propostas de continuidade

Considerando as etapas cumpridas, alcançando os objetivos propostos para a finalização do projeto. Para este trabalho, há propostas de atividades futuras.

A primeira recomendação é a implementação de uma camada de segurança para garantir a confiabilidade do acesso aos dados, não permitindo que o estes, tanto o conteúdo das câmeras quanto do valor das variáveis, sejam extraviados.

Para suprir a necessidade de monitoramento do ambiente, utilizando-se da imagem da

câmera, aconselha-se acrescentar no ambiente de monitoramento o acesso em real time a visão da câmera do sistema embarcado. A implementação desta extensão permitirá que o usuário tenha total acesso a visão do ambiente remotamente, além de poder, configurar o posicionamento do sistema embutido acompanhando pelo aplicativo.

Apêndice A

Códigos

A.1 Instalação da biblioteca OpenCV no sistema operacional Windows

Para ter conhecimento da biblioteca OpenCv 2.4.9 foi realizado a sua instalação no sistema operacional *Windows* 10 e configurada no *software Visual Studio* IDE 2013.

Etapa 1:

Primeiramente, realizou-se o *download* da biblioteca OpenCv 2.4.9 presente no *site* $http://www.opencv.org/$. A OpenCv 2.4.9 foi escolhida devido a estabilidade desta versão enquanto versões mais atuais apresentam algumas instabilidades na utilização dos métodos que serão aplicados no trabalho.

Etapa 2:

Posteriormente, realizou-se a instalação da biblioteca utilizando-se o executável disponível após o *download*. Instalando a biblioteca é questionado o local de instalação da mesma, no caso deste projeto, foi realizada a instalação em uma pasta chamada OpenCv-2.4.9 localizada em "C:".

Etapa 3:

Em seguida realizou-se o *download* do *software* Cmake compatível com o computador utilizado, no site $https://cmake.org/download/$. Este é um sistema multiplataforma responsável por estender a biblioteca desejada. Ao final da instalação do *software*, o arquivo foi executado e assim abriu-se uma interface em que foi inserido duas pastas. A primeira na opção '*Where is the source code:*' inseriu-se a pasta *source* da biblioteca OpenCv 2.4.9 instalada, ou seja, a pasta com o endereçamento $C:/OpenCv-2.4.9/opencv/sources$. Em seguida, para preenchimento da segunda opção '*Where to build the binaries:*', foi

criada uma pasta para obtenção da extensão da biblioteca. Esta, criada com o nome my_build no endereço $C : /OpenCv - 2.4.9/opencv/mybuild$. Após a inserção das pastas, foi clicado em *configure* e uma janela foi aberta para inserir as especificações do gerador do sistema do computador, conforme Figura A.1.

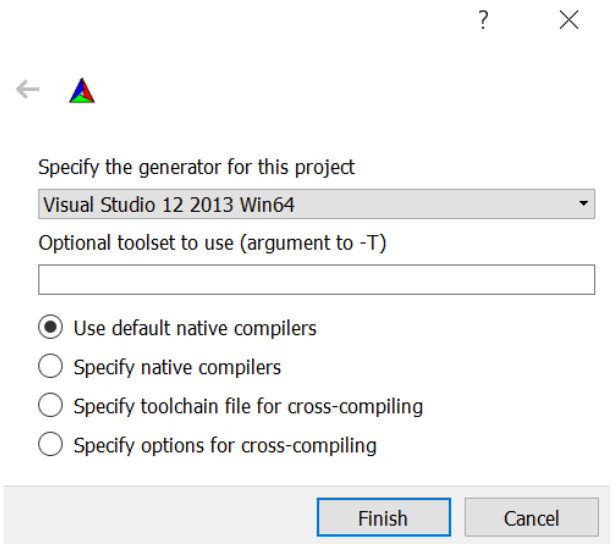


Figura A.1: Janela para preenchimento das especificações do sistema.

Etapa 4:

Aguardou-se então a configuração do sistema para gerar os arquivos necessários para obtenção da extensão da biblioteca compatível, após, o *software* CMake retornou as opções de extensões para serem incluídas à biblioteca, conforme figura A.2.

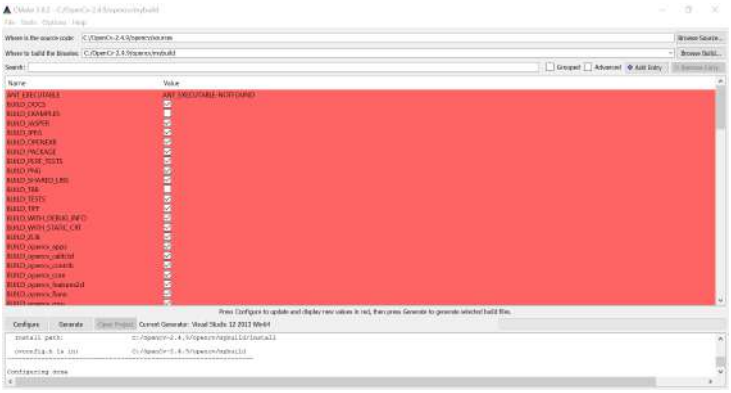


Figura A.2: Parâmetros para escolha das extensões a serem realizadas à biblioteca.

Etapa 5:

Posteriormente, selecionou-se a opção *generate* para gerar os arquivos necessários para a extensão da biblioteca $OpenCv2.4.9$. Os arquivos foram criados na pasta *mybuild* e utilizando o *software* Visual Studio IDE 2013 compilou os arquivos, Figura A.3.

Para isso, abriu-se o projeto 'OpenCv.sln' e na aba *Solution explorer* presente no Visual Studio realizou a configuração da IDE para construir o arquivo ALL BUILD em *Debug* e depois em *Release*. O *Debug* criou os arquivos apropriados $xxxxx249d.lib$, $xxxxx249d.exp$ e $xxxxx249d.pdb$. Já o *Release* criou todos os arquivos $xxxxx249.lib$, $xxxxx249.exp$ e $xxxxx249.dll$.

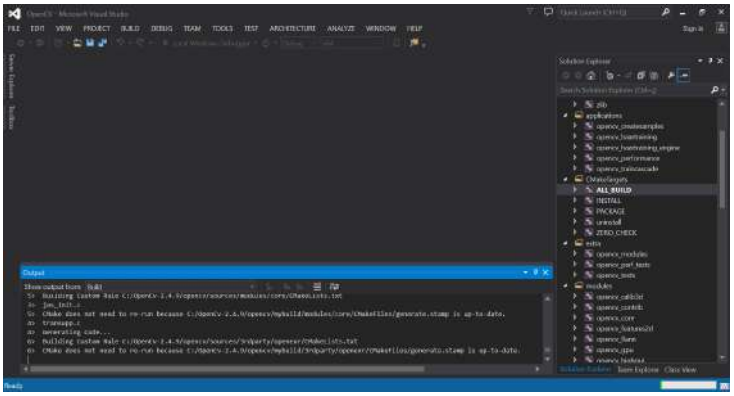


Figura A.3: Configuração para gerar os complementos da biblioteca OpenCv.

Etapa 6:

Depois de ambos os conjuntos de compilação estarem completos, ainda na aba *Solution Explorer* compilou o arquivo *INSTALL*, para que as bibliotecas *Debug* e *Release* fosse unidas nas pastas "lib" e "bin". Ao fim do processo de compilação da biblioteca atualizada obteve-se os novos diretórios

Etapa 7:

Após a obtenção dos novos diretórios utilizou-se a nova biblioteca para compilar e executar os programas OpenCv no Visual Studio. Para isso foi realizado a configuração do compilador C/C++ dizendo-o onde estão os arquivos de cabeçalho e configurando o sistema *Linker* dizendo-o onde estão localizados os arquivos de bibliotecas compilados.

Etapa 8:

Finalmente, após as todas as configurações adicionou-se a localidade da biblioteca *.dll* nas variáveis de ambiente, variáveis do sistema PATH.

A.2 Método Skin detection

```
//incluindo bibliotecas
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>
```

```
using namespace cv;
//declaração de variáveis
int findBiggestContour(vector<vector<Point> >);
Mat src, hsv, bw, imgCanny, canny_output;
int main(){
src = imread("image.jpg");  // busca da imagem que deve ser carregada
if (src.empty()) // caso não for encontrada retornar -1
return -1;
blur(src, src, Size(20, 20)); // aplicação do método blur
cvtColor(src, hsv, CV_BGR2HSV);// conversão do espaço de cor RGB para HSV
inRange(hsv, Scalar(0, 60, 10), Scalar(10, 120, 240), bw);
//preenchimento da maior área encontrada para cobrir toda a área do objeto
vector<vector<Point> > contours;
vector<Vec4i> hierarchy;
findContours(bw, contours, hierarchy,
CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE, Point(0, 0));
int s = findBiggestContour(contours);
Mat drawing = Mat::zeros(src.size(), CV_8UC1);
drawContours(drawing, contours, s, Scalar(255,255,255),
 -1, 8, hierarchy, 0, Point());
cv::Canny(drawing, imgCanny, // Aplicando a função Canny
0, 255);     // limite inferior e limite superior

// configuração das janelas para mostrar os resultados
namedWindow("blur", CV_WINDOW_NORMAL);
namedWindow("Canny", CV_WINDOW_NORMAL);
namedWindow("hsv", CV_WINDOW_NORMAL);
namedWindow("dr", CV_WINDOW_NORMAL);

//exibindo as janelas
imshow("hsv", hsv);
imshow("blur", src);
imshow("dr", drawing);
imshow("Canny", imgCanny);
waitKey(0);
return 0;}
```

```
// método para identificação do contorno da imagem após o filtro
int findBiggestContour(vector<vector<Point> > contours){
int indexOfBiggestContour = -1;
int sizeOfBiggestContour = 0;
for (int i = 0; i < contours.size(); i++){
if (contours[i].size() > sizeOfBiggestContour){
sizeOfBiggestContour = contours[i].size();
indexOfBiggestContour = i;}
}
return indexOfBiggestContour;}
```

A.3 Código de contabilização de círculos

```
#include "opencv2/imgproc/imgproc.hpp"
#include<opencv2/highgui/highgui.hpp>
#include <iostream>

using namespace std;
using namespace cv;

int main()
{
Mat src, src_gray;
src = imread("circ.png");  // buscando a imagem
if (src.empty())      // se a imagem não for encontrada
{ std::cout << "ERRO: Imagem não foi encontrada\n";   // Exibir mensagem
return(0);}    // sair do programa

cvtColor(src, src_gray, CV_BGR2GRAY); // convertendo imagem
GaussianBlur(src_gray, src_gray, Size(9, 9), 2, 2); // reduzindo os ruídos
vector<Vec3f> circulo; // criando vetor
HoughCircles(src_gray, circulo, CV_HOUGH_GRADIENT,
 1, src_gray.rows / 8, 200, 100, 0, 0); // aplicanto Hough Circle

for (size_t i = 0; i < circulo.size(); i++) // Contornando ciruclos
{
Point center(cvRound(circulo[i][0]), cvRound(circulo[i][1]));
int radius = cvRound(circulo[i][2]);
circle(src, center, 3, Scalar(0, 255, 0), -1, -0, 0); // ponto de centro
circle(src, center, radius, Scalar(0, 0, 255), 3, -8, 0); // contorno
}
cout << circulo.size(); // retorno da quantidade de circulos
// mostrando os resultados
namedWindow("Contagem de circulos", CV_WINDOW_NORMAL);
imshow("Contagem de circulos", src);
waitKey(0);
return 0;
}
```

Referências

AGC. acessado em 29/04/2017, `http://svtsim.com/moonjs/agc.html`.

ARAUJO, G. M. *Algoritmo para reconhecimento de características faciais baseado em filtros de correlação*. 2010. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio de Janeiro.

AZEVEDO, E.; CONCI, A. *Computação gráfica*: teoria e prática. [S.l.]: Elsevier, 2003.

BARR, M. *Programming embedded systems in C and C++*. [S.l.]: "O'Reilly Media, Inc.", 1999.

BERGER, A. S. *Embedded systems design*: an introduction to processes, tools, and techniques. [S.l.]: Focal Press, 2002.

BRADSKI, G.; KAEHLER, A. *Learning OpenCV*: computer vision with the opencv library. [S.l.]: "O'Reilly Media, Inc.", 2008.

COR. acessado em 12/05/2017, `http://www.uff.br/cdme/matrix`.

CUNHA, A. F. O que são sistemas embarcados. *Saber Eletrônica*, [S.l.], v.43, n.414, p.1–6, 2007.

DATTA, A. K.; MUNSHI, S. *Information Photonics*: fundamentals, technologies, and applications. [S.l.]: CRC Press, 2016.

DAWSON-HOWE, K. *A practical introduction to computer vision with opencv*. [S.l.]: John Wiley & Sons, 2014.

EMBARCADOS. acessado em 20/04/2017, `https://www.embarcados.com.br/`.

EMBEDDED. acessado em 29/04/2017, `https://goo.gl/sQWUeh`.

EMBRAPA. acessado em 20/04/2017, `https://goo.gl/gUbYVG`.

GILCHRIST, A. Industry 4.0 - The Industrial Internet of Things. *Thailand*, [S.l.], 2016.

GOLDSMITH, A. Wireless communications. , [S.l.], 2005.

HAYKIN, S. S. *Redes neurais*. [S.l.]: Bookman, 2001.

HIRSCHLER, R. *Controle metrológico da cor aplicado à estamparia digital de materiais têxteis*. 2009. Tese (Doutorado em Ciência da Computação) — PUC-Rio.

HUANG, T. Computer Vision: evolution and promise. *CERN EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH-REPORTS-CERN*, [S.l.], p.21–26, 1996.

IBMEC. acessado em 19/04/2017, `https://goo.gl/8fuacG`.

KUEHNI, R. G. Color space and its divisions. *Color Research & Application*, [S.l.], v.26, n.3, p.209–222, 2001.

LEE, J.-S.; SU, Y.-W.; SHEN, C.-C. A comparative study of wireless protocols: bluetooth, uwb, zigbee, and wi-fi. In: INDUSTRIAL ELECTRONICS SOCIETY, 2007. IECON 2007. 33RD ANNUAL CONFERENCE OF THE IEEE. *Anais...* [S.l.: s.n.], 2007. p.46–51.

LIST, R. R. V. Colour Spaces. , [S.l.], 2015.

MARENGONI, M.; STRINGHINI, S. Tutorial: introdução à visão computacional usando opencv. *Revista de Informática Teórica e Aplicada*, [S.l.], v.16, n.1, p.125–160, 2009.

MARQUES FILHO, O.; NETO, H. V. *Processamento digital de imagens*. [S.l.]: Brasport, 1999.

MATHER, P. M.; KOCH, M. *Computer processing of remotely-sensed images*: an introduction. [S.l.]: John Wiley & Sons, 2011.

NYTIMES. acessado em 19/04/2017, `https://goo.gl/tQnKKx`.

OPEN. acessado em 12/05/2017, `http://www.opencv.org/`.

PEREIRA, J. C. Metodologia de projeto aplicada à concepção de sistemas mecatrônicos a partir da elaboração de um modelo prescritivo de desenvolvimento. , [S.l.], 2016.

PHASE. acessado em 29/04/2017, `https://goo.gl/1imGXb`.

PRITHA, M. A.; JESLET, D. S. A Study on Sampling and Quantization Techniques of Image Processing. , [S.l.].

SCURI, A. E. Fundamentos da imagem digital. *Pontifícia Universidade Católica do Rio de Janeiro*, [S.l.], 1999.

SILVA, C. C.; ANDRADE MARTINS, R. de. A teoria das cores de Newton: um exemplo do uso da história da ciência em sala de aula. *Ciência & Educação*, [S.l.], v.9, n.1, p.53–65.

STEFEN, C. A. Introdução ao sensoriamento remoto. In: *Anais...* [S.l.: s.n.], 2008.

THOMAZINI, D.; ALBUQUERQUE, P. U. B. Sensores industriais: fundamentos e aplicações. *São Paulo*, [S.l.], v.3, p.32, 2005.

TKALCIC, M.; TASIC, J. F. *Colour spaces*: perceptual, historical and applicational background. [S.l.]: IEEE, 2003. v.1.

VIEIRA, D. A. *et al*. Visão computacional para monitoramento ambiental das áreas cobertas por linhas de transmissão utilizando reconhecimento de padrões. , [S.l.], 2015.