UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

Rodrigo Ferraz Orrú

Sistema de vigilância por imagens com armazenamento reduzido

São Carlos
2019

RODRIGO FERRAZ ORRÚ

SISTEMA DE VIGILÂNCIA POR IMAGENS COM ARMAZENAMENTO REDUZIDO

Trabalho de Conclusão de Curso apresentado à Escola de Engenharia de São Carlos, da Universidade de São Paulo

Curso de Engenharia Elétrica, com ênfase em Eletrônica

ORIENTADOR: Prof. Dr. Evandro Luis Linhari Rodrigues

São Carlos

2019

AUTORIZO A REPRODUÇÃO TOTAL OU PARCIAL DESTE TRABALHO, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Orrú, Rodrigo Ferras
O75s Sistema de vigilância por imagens com armazenamento reduzido / Rodrigo Ferras Orrú; orientador Evandro Luis Linhari Rodrigues. São Carlos, 2019.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2019.

1. Sistema de Segurança. 2. Internet das Coisas. 3. Sistema Embarcado . I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE APROVAÇÃO

Nome: Rodrigo Ferraz Orrú

Título: "Sistema de vigilância por imagens com armazenamento reduzido"

Trabalho de Conclusão de Curso defendido e aprovado
em 12/06/19,

com NOTA 8,7 (OITO, SETE), pela Comissão Julgadora:

Prof. Associado Evandro Luis Linhari Rodrigues - Orientador - SEL/EESC/USP

Prof. Associado Marcelo Andrade da Costa Vieira - SEL/EESC/USP

Prof. Associado Adilson Gonzaga - SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

Dedicatória

Aos meus pais, Eduardo e Carla, que sempre me dizem o que é melhor. Ao Prof. Dr. Evandro Luis Linhari Rodrigues, pelo voto de confiança e pelos conselhos dados. Aos meus irmãos, Ana e Marcelo, pelos momentos de companheirismo. Aos meus companheiros de classe, por estarem comigo em todos os momentos da minha graduação.

Falhar é uma opção aqui. Se você não está errando, você não está inovando o suficiente
(ELON MUSK)

Resumo

Num contexto em que a questão da segurança pública vem sido amplamente discutida e o tópico Internet das Coisas é cada vez mais falado, encontra-se esse projeto de um sistema de segurança formado por uma câmera e um sistema embarcado. O sistema embarcado (Raspberry Pi) realiza o tratamento completo das gravações feitas pela câmera. Primeiramente são identificados os momentos de movimentos através da técnica de subtração de fundo, depois são formadas sequências de aproximadamente o mesmo tamanho otimizadas por um algoritmo que utiliza-se do conceito de busca em amplitude e então eles são sobrepostos através de uma técnica de composição de imagens para formar um curto vídeo mostrando diversos acontecimentos detectados na gravação de forma rápida e simultânea. No mercado existe um sistema da empresa FLIR que realiza um procedimento semelhante, porém o processamento é realizado na nuvem e o custo é mais alto. Assim, esse projeto tem como diferencial um sistema de segurança com resposta ao usuário bem rápida, com baixo custo e que facilita a busca de eventos com movimento na cena.

Palavras-Chave: Sistema de Segurança, Internet das Coisas, Sistema Embarcado

Abstract

In a context where the issue of public safety has been widely discussed and the topic Internet of Things is increasingly talked about, it is this project of a security system consisting of a camera and an embedded system. The embedded system (Raspberry Pi) performs the full treatment of the recordings made by the camera. First the moments of movements are identified through the background subtraction technique and then sequences of approximately the same size are formed optimized by an algorithm using the concept of breadth-first search and then they are superimposed through an image composition technique to form a short video showing various events detected in recording quickly and simultaneously. In the market there is a FLIR company system that performs a similar procedure, however the processing is performed in the cloud and the cost is higher. Thus, this project has the differential of a security system with very quick response to the user, low cost and facilitates the search of events with movement in the scene.

Keywords: Security System, Internet of Things, Embedded System.

Lista de Figuras

2.1: Exemplo de rede semântica 26
2.2: Árvore de busca correspondente a rede da Figura 2.1 27
2.3: Procedimento de busca em profundidade 27
2.4: Procedimento de busca em amplitude 28
3.1: Raspberry Pi 3 Model B em detalhe 31
3.2: Fluxograma geral do projeto 34
3.3: *Frame* de vídeo teste 36
3.4: Saída parcial produzida pelo algoritmo 36
3.5: Regiões de movimento detectadas 37
3.6: Arquivo de texto para vídeo contendo movimento 38
3.7: Árvore de busca para primeira sequência de vídeos 40
3.8: Árvore de busca para segunda sequência de vídeos 41
3.9: Árvore de busca para terceira sequência de vídeos 41
3.10: Árvore de busca para quarta sequência de vídeos 42
3.11: Árvore de busca para quinta sequência de vídeos 42
3.12: Tela de instalação do sistema operacional 44
3.13: Arquivo de configurações de SSH 45
4.1: Exemplo 1 de *frame* do vídeo final gerado pelo algoritmo 50
4.2: Exemplo 2 de *frame* do vídeo final gerado pelo algoritmo 50
4.3: Figura com tabela de tempo de processamento dos principais algoritmos 51
4.4: Gráfico de validação de modelo de linearidade 54
4.5: Exemplo de problema de múltiplas *tags* para um mesmo corpo em movimento 55
4.6: Exemplo de *frame* de vídeo gerado pela tecnologia RapidRecap 56
4.7: Exemplo de *frame* de vídeo gerado pela tecnologia RapidRecap com falhas 56
B.1: Modal de Reserva de IP 104
B.2: Modal de PortFowarding 105

Lista de Tabelas

Tabela 3.1: Tabela comparativa entre modelos de Raspberry 32

Tabela 4.1: Tabela comparativa de perfomance..... 52

Tabela 4.2: Tabela com dados de teste da influência do movimento no tempo de processamento 53

Lista de Siglas

- **AGC :** Apollo Guidance Computer
- **ARM :** Acorn RISC Machine
- **BBC :** British Broadcasting Corporation
- **BSD :** Berkeley Software Distribution
- **CISC :** Complex Instruction Set Computer
- **CPU :** Central Processing Unit
- **CUDA :** Compute Unified Device Architecture
- **FPS :** *Frames* per second
- **GCC :** GNU Compiler C
- **GPU :** Graphics Processing Unit
- **GUI :** Graphical User Interface
- **HD :** Hard Disk
- **HDMI :** High-Definition Multimedia Interface
- **IA :** Inteligência Artificial
- **IBGE :** Instituto Brasileiro de Geografia e Estatística
- **IOT :** Internet of Things
- **IP :** Internet Protocol
- **JPEG :** Joint Photographics Experts Group
- **MAC :** Media Access Control
- **MB :** Megabyte
- **MATLAB :** MATrix LABoratory
- **MHZ :** Megahertz
- **NIST :** National Institute of Standards and Technology
- **NOR :** Non-Operating Reefer
- **OpenGL :** Open Graphics Library
- **OpenCV :** Open Source Computer Vision Library
- **PC : pe**rsonal computer
- **PNG :** Portable Network Graphics
- **RAM :** Random Access Memory
- **RGB :** abreviatura de um sistema de cores aditivas: o Vermelho (Red), o Verde (Green) e o Azul (Blue)
- **RISC :** Reduced Instruction Set Computer
- **SD :** Secure Digital
- **SO :** Sistema Operacional
- **SSH :** Secure Shell
- **TCP :** Transmission Control Protocol
- **USB :** Universal Serial Bus
- **VNC :** Virtual Network Computing
- **VPU :** Video Processing Unit

Sumário

1. Introdução ... 19

1.1. Motivação ... 20

1.2. Objetivos ... 20

1.3. Justificativa/Relevância ... 21

1.4. Organização do Trabalho ... 21

2. Fundamentação Teórica ... 22

2.1. Visão Computacional e Processamento de Imagens ... 22

2.2. Detecção de movimentos em vídeos ... 23

2.3. Composição de Imagens ... 24

2.4. Inteligência Artificial ... 25

2.5. Sistemas Embarcados ... 28

2.5.1. Arquitetura ARM ... 29

2.5.2. Linux ... 29

3. Materiais e Métodos ... 31

3.1 Materiais ... 31

3.2. Softwares ... 32

3.2.1. MATLAB ... 32

3.2.2. OpenCV ... 33

3.2.3. Raspbian ... 34

3.3. Métodos ... 34

3.3.1. Desenvolvimento em MATLAB ... 34

3.3.2. Desenvolvimento em C++ ... 43

4. Resultados e Discussões ... 50

5. Conclusões ... 57

5.1. Implementações futuras ... 57

6. Referências Bibliográficas ... 59

Apêndice A ... 62

Apêndice B ... 104

1. Introdução

A população urbana no Brasil em 1980 equivalia a 67,7%, já em 2010 aumentou para 84,36%, mostrando a tendência ao crescimento das cidades nas últimas décadas o que indica mais pessoas atrás de oportunidades de emprego e melhora de vida [1]. Porém, em março de 2017, foi registrado um aumento na taxa de desemprego (13,7%) por conta da crise econômica na qual o país se encontrava, consequentemente, deixando famílias sem meios de sustento, aumentando a desigualdade social e restrições econômicas [2]. A alta taxa de desemprego é um dos fatores para o aumento da criminalidade que tem se agravado dia-a-dia afetando a vida de dos cidadãos gerando medo e insegurança [3]. De acordo com o IBGE, no ano de 2009, 11,9 milhões de pessoas foram vítimas de roubo ou furto nas cidades brasileiras, sendo que 40,7% dos furtos ocorrem em residências retirando a sensação de segurança dos moradores [4]. Para gerar uma sensação de segurança, os cidadãos buscam opções que lhes deem mais controle sobre seus patrimônios comerciais e residenciais, como muros cada vez mais altos, correntes nas portas, contratação de vigias, cercas eletrificadas e tecnologias de vigilância. Como indicam pesquisas foi registrado em 2009, 34,8 milhões de domicílios que utilizavam câmeras de vigilância.

À medida que a necessidade de restringir o acesso a condomínios e prédios comerciais se torna mais comum, o uso de equipamentos não tecnológicos e recursos humanos para funções de proteção, monitoramento e controle de ações deixa de ser a primeira opção ao pensar em segurança. O desenvolvimento de tecnologias eletrônicas tornou os preços de sistemas de monitoramento mais acessíveis [5]. São muitas as ofertas de sistemas de monitoramento que registram o fluxo de pessoas, garantindo a segurança de estabelecimentos hoje em dia, por esse motivo existe grande variedade de equipamentos para esse fim (câmeras, sistemas de controle e armazenamento), as combinações entre eles podem nem sempre ser tão eficazes.

A melhora da qualidade das câmeras de segurança (em questão de qualidade de definição, detecção de ruídos, controle de saturação, etc.) a *priori* parece ser um aspecto somente positivo, porém traz consigo uma limitação: o espaço para o armazenamento das gravações é cada vez maior e o tempo de gravação armazenado geralmente corresponde a menos de um mês, apagando acontecimentos anteriores. Além disso, no tempo gravado, há momentos nos quais por um longo período nenhuma movimentação é detectada, mas esses momentos são armazenados da mesma forma ocupando espaço de memória.

Esse trabalho realiza a sobreposição dos movimentos encontrados em um vídeo com duração de um dia na forma de um pequeno filme, reduzindo dessa forma a quantidade de informação a ser armazenada e tornando mais fácil a procura de acontecimentos que desejados pelas filmagens. Dessa forma é possível armazenar em memória mais que um mês de gravação e o usuário poderá encontrar eventos com movimento na cena de forma muito mais rápida do que em sistemas convencionais.

1.1. Motivação

A motivação deste trabalho se deu após a realização da disciplina SEL0373 - Projeto em Sistemas Digitais na qual foi desenvolvido um projeto utilizando sistemas embarcados e a disciplina SEL0339 - Introdução à Visão Computacional, que abordou os conceitos básicos de visão computacional. Além disso, outro fator que influenciou esse projeto foi a precariedade de segurança pública no país.

A falta de segurança é evidente nas cidades brasileiras e a percepção pela população é comprovada por pesquisas recentes e a ocorrência de assaltos até em regiões movimentadas. O desejo de se sentir seguro é crescente e as possibilidades se mostram muito variadas mostrando necessária a comparação de formas diferentes de proteção para se chegar a uma melhor opção.

O uso desse sistema torna possível armazenar grande tempo de gravação usando menos espaço de armazenamento porque grava todo o tempo, identifica os momentos em que há movimento no ambiente, sobrepõe as imagens, adiciona *tags* com os horários a qual correspondem e transforma em pequenos filmes que requerem menos espaço de armazenamento, possibilitando encontrar com mais rapidez movimentos incomuns e suspeitos.

A ideia desse projeto é dar mais segurança para o proprietário de um imóvel (esse podendo ser um comércio ou uma residência pela amplitude de aplicação do sistema) ao sair do local, permitindo permanecer mais tempo longe sem ter que se preocupar com invasões e furtos sem poder tomar providências por não estar informado da situação a tempo.

Além disso, a maioria dos sistemas de segurança convencionais da atualidade, apesar de realizarem a gravação apenas quando existe movimento, não realizam a sobreposição dos movimentos dificultando dessa forma encontrar um evento que não se sabe a hora exata que ocorreu e ainda ocupando muito espaço de memória.

Existe um sistema do mercado com uma tecnologia denominada Rapid Recap que realiza a sobreposição dos movimentos, porém possui um custo aproximado de $200 e possui algumas falhas na detecção que esse trabalho procura corrigir.

1.2. Objetivos

O objetivo desse projeto é a criação de um sistema de segurança de baixo custo que otimize simultaneamente a quantidade de informação a ser armazenada e o tempo de busca de eventos ocorridos durante um período de tempo, através de algoritmo que identifique os movimentos, otimize a melhor combinação de movimentos para que as sequências formada por eles tenham aproximadamente o mesmo tamanho e faça a sobreposição dos movimentos encontrados neste período em um vídeo com *tag* e suportados por CPU e Câmera com boa relação de custo benefício.

Para verificar se o objetivo foi cumprido, esse sistema desenvolvido será comparado a outra solução existente no mercado.

1.3. Justificativa/Relevância

A justificativa principal desse projeto deve-se ao alto custo de armazenamento de filmagens de um período muito longo e principalmente o tempo necessário para busca de um determinado evento com movimento nas cenas. Esse projeto visa reduzir drasticamente esse problema.

1.4. Organização do Trabalho

Pra melhor organização e entendimento do texto, a monografia foi dividida em 5 capítulos listados baixo:

- **Capítulo 1** – Introdução: contém a motivação, objetivos e justificativa do projeto.

- **Capítulo 2** – Embasamento Teórico: Descreve os principais conceitos e as ferramentas utilizadas para execução do projeto

- **Capítulo 3** – Materiais e Métodos: Discorre sobre os materiais e métodos utilizados durante o desenvolvimento do projeto

- **Capítulo 4** – Resultados e Discussões: Apresenta os resultados parciais obtidos pelo projeto e discorre sobre os mesmos.

- **Capítulo 5** – Conclusões Parciais: Conclui o que foi realizado e indica os caminhos que irão tomar o projeto.

2. Fundamentação Teórica

Neste capítulo são descritos os principais fundamentos desse projeto, que tem como base alguns tópicos muito importantes: visão computacional, inteligência artificial e sistemas embarcados.

2.1. Visão Computacional e Processamento de Imagens

Antes de entrar no tópico principal desta seção, é preciso definir o que é uma imagem digital e como ela é obtida. Uma imagem digital monocromática é uma função bidimensional f(x,y), na qual x e y são as coordenadas espaciais de cada *pixel* (elemento básico de uma imagem), cada *pixel* é representado digitalmente por um número que é proporcional a sua intensidade luminosa (nível de cinza). No caso de imagens coloridas no sistema RGB, cada imagem é representada por três matrizes bidimensionais, cada uma ligada a um dos canais de cores [6].

Para uma imagem digital ser obtida e processada por um computador, é necessário, após a obtenção da imagem através de um sensor (câmera) a realização do processo de amostragem, que discretiza as coordenadas da imagem e o processo de quantização, que discretiza os valores de brilho (nível de cinza). Normalmente o número de valores de brilho que são utilizados para uma imagem digital é uma potência de dois e o valor mais utilizado atualmente é o 256 (8 bits ou 24 no caso de imagens coloridas). Já um vídeo digital é um conjunto de imagens que são apresentadas sequencialmente que pode conter também som. Cada imagem em um vídeo é denominada *frame* e a frequência em que as imagens são apresentadas é denominado *frame rate*.

O Processamento de Imagens tem por finalidade aplicar métodos de tratamento e análise de imagens com objetivo de extrair características importantes, que auxiliem na compreensão da imagem e na tomada de decisões inteligentes O Processamento de Imagens consiste na manipulação computacional de arquivos de imagens digitais com o objetivo de avaliar grandezas mecânicas representadas por tais imagens. As etapas deste processo podem ser sintetizadas em capturar a imagem de um objeto através de uma câmera e envaiar esta imagem, depois de digitalizada, a um microcomputador e aplicar nela uma série de operações matemáticas a fim de se obter outra imagem que facilite a buscas de informações relevantes, levando a um reconhecimento de padrões que em visão computacional, trata-se da interpretação e identificação de objetos e características específicas em uma imagem. Este reconhecimento de padrões em combinação com a Inteligência Artificial pode levar a decisões rápidas e tempestivas. IA é normalmente usada junto à visão computacional para obtenção, reconhecimento e classificação de objetos e características nas imagens [7]. Em geral é aplicada em três etapas: percepção, cognição e ação. A etapa de percepção traduz os sinais provenientes do ambiente em símbolos, a de cognição manipula estes símbolos e a de ação traduz o resultado destes símbolos em ações que devem ser aplicadas de volta ao ambiente.

Processamento de imagens é uma área em crescimento e vem sendo utilizada por diversas áreas do conhecimento, como por exemplo: por geógrafos, para estudar padrões de relevo ou poluição, pela medicina, pela arqueologia, para aplicações em segurança pública, entre muitas outras.

Essa área de conhecimento é derivada diretamente do processamento de sinais, pois uma imagem nada mais é que um conjunto de sinais que trazem consigo diversas informações [8]. Para conseguirmos extrair mais facilmente essas informações é necessário executar diversos processos.

O processamento de imagens normalmente é dividido em diversas etapas. O primeiro passo efetivo do processamento é denominado pré-processamento, nessa etapa são realizadas ações de filtragem de ruídos e correções de distorções geométricas causadas pelos sensores. A filtragem espacial se fundamenta em uma operação de convolução de uma máscara (*template*) e de uma imagem digital. Alterando-se a máscara obtemos diversos resultados diferentes com inúmeras aplicações. Dos filtros mais comuns utilizados no processamento digital de imagens estão os filtros de média, que normalmente são aplicados para suavizar a imagem, para que não haja variações abruptas nos níveis de cinza da imagem, e os filtros de mediana, que são muito aplicados para eliminação do ruído do tipo sal e pimenta.

Após a etapa de pré-processamento, normalmente passa-se para etapa de análise, que consiste em começar a extrair algumas características da imagem, como texturas, bordas, vizinhanças, etc. Para isso normalmente são aplicadas uma cadeia de processamentos e essa etapa pode ficar tão complicada conforme o necessário para a aplicação [7]. Depois parte-se para segmentação, isto é, a separação dos objetos a serem identificados, ou do movimento da cena do plano de fundo da imagem (*background*).

2.2. Detecção de movimentos em vídeos

Para detecção de movimento existem diversas técnicas. A técnica de subtração de fundo consiste em comparar cada novo quadro com o modelo de fundo da cena previamente calculado [9]. O cálculo do modelo de fundo pode ser feito de diversas maneiras, podendo ser desde apenas uma imagem do fundo, se isso for suficiente para resolver o problema ou até mesmo um conjunto de amostras obtidas a partir de uma janela de observação, neste caso, como não existe apenas uma imagem a ser comparada com o quadro atual serão necessárias outras operações para a medir a distância deste em relação ao modelo de fundo e assim verificar se existe movimento na cena [10]. O principal problema desse método consiste em calcular bem o modelo de fundo e existem diversos fatores que podem prejudicar esse cálculo como por exemplo:

(a) os objetos que constituem o fundo da cena podem mover-se e ficarem parados numa nova posição ou novos objetos podem ser adicionados ao fundo, fazendo com que sempre seja detectado movimento em relação ao fundo previamente calculado, até que um novo modelo de fundo seja calculado;

(b) em ambientes externos as variações na iluminação do dia modificam o fundo;

(c) o fundo nem sempre é completamente estático, pequenas variações como árvores balançando ao vento podem gerar detecção de movimento, entre outros.

Outra técnica para detecção de movimento é a da diferença temporal. Esse método consiste em calcular a diferença entre os *pixels* de dois ou mais *frames* de uma sequência de imagens para identificar se existe movimento na cena [11]. Dessa forma enquanto a técnica de subtração de fundo calcula um modelo de fundo consistente, esta técnica utiliza como referência o *frame* anterior. Existem falhas neste método, quando por exemplo um objeto para na cena por alguns *frames* e depois volta a mover. Neste caso, não seria detectado um objeto novo na cena durante esses *frames* e o movimento seria dividido em dois movimentos separados. Outra complexidade desse método consiste em estipular o valor do limiar do que é considerado movimento para o algoritmo.

Outro método um pouco mais elaborado é o do fluxo óptico. O fluxo óptico é a distribuição das velocidades aparentes dos padrões de brilho numa imagem. Este fluxo contém várias informações que podem ser utilizadas para diversas aplicações, incluindo rastreamento de objetos em movimento [12].

Existem três grupos principais para os métodos para a computação do Fluxo Óptico: técnicas diferenciais, técnicas de correlação e técnicas baseadas em frequência/energia. Apesar das diferenças entre os diferentes métodos, o Fluxo Óptico pode ser dividido conceitualmente em cinco estágios de processamento: pré-filtragem, extração de medidas básicas, cálculo de velocidades, detecção dos objetos e, por último, a validação [13].

Os principais problemas desse método são o alto custo computacional e a dificuldade para representar os objetos em movimento [14].

Por fim, a última etapa do processamento de imagens é a classificação, que é considerada a etapa de mais alto nível e tem como objetivo reconhecer ou inferir identidade aos objetos. É nessa etapa que o uso da visão computacional é bem importante, pois até chegarmos aqui já conseguimos extrair uma lista de características da imagem que podem nos ajudar a classificar o objeto em estudo.

Em geral, devido à complexidade dos problemas nessa área, o processamento de imagens não possui solução única e muitas vezes são limitados pelo hardware que está sendo utilizado.

Existem diversas bibliotecas e ferramentas para o processamento de imagens e a utilização de visão computacional. Nos materiais serão descritas duas ferramentas utilizadas no trabalho.

2.3. Composição de Imagens

No ano de 1977, Alvy Ray Smith e Ed Catmull, inventaram um método para realizar a composição entre uma imagem de primeiro plano parcialmente opaca sobre uma imagem de fundo completamente opaca. A opacidade de cada *pixel* no primeiro plano é dada por uma terceira matriz denominada alfa, cujos valores variam de 0.0 (totalmente transparente) a 1.0 (totalmente opaco). Posteriormente essa matriz foi introduzida como um quarto canal no sistema RGB formando o sistema RGBA [15]. A Composição entre duas imagens pode ser obtida através da equação 2.1.

$$I = \alpha F + (1 - \alpha)B \tag{2.1}$$

Onde I é a imagem final, α é a matriz alfa, F é a imagem em primeiro plano e B a imagem de fundo. Através dessa equação é possível observar:

- Caso o valor de um elemento de alfa seja 0 o valor deste *pixel* na imagem resultante é igual ao *pixel* correspondente da imagem de fundo;
- Caso o valor de um elemento de alfa seja 1 o valor deste *pixel* na imagem resultante é igual ao *pixel* correspondente a imagem em primeiro plano;
- Caso o valor de um elemento de alfa seja um valor entre 0 e 1 o valor deste pixel na imagem resultante é uma combinação entre os *pixels* da imagem em primeiro plano com a imagem de fundo;

2.4. Inteligência Artificial

A inteligência artificial é um ramo de pesquisa da ciência da computação que busca, através de símbolos computacionais, construir mecanismos ou dispositivos que simulem a capacidade do ser humano de pensar e resolver problemas [16]. O estudo e desenvolvimento desse ramo de pesquisa tiveram início na Segunda Guerra Mundial. Pode-se dizer que o primeiro grande trabalho reconhecido como IA foi realizado por Warrem Macculloch e Walter Pitts (1943), que desenvolveram o primeiro neurônio artificial. Logo surgiram outros nomes, como Alan Turing, que propôs o Teste de Turing, o qual testa a capacidade de uma máquina exibir comportamento inteligente de forma que não pode ser distinguido de um ser humano [17]. Em 1957, criado por Frank Rosenblatt, foi inventado o *perceptron*, que tinha como propósito resolver problemas de classificações de padrões. Anos depois, Bernard Widrow, desenvolveu o ADALINE, que carrega consigo uma poderosa lei de aprendizado e devido a isso é utilizado até hoje para resolver alguns problemas de classificações de padrões [18].

A partir de 2015 a popularidade de IA cresceu drasticamente. Muito disso tem a ver com a disponibilidade dos GPUs que fazem com que processamento paralelo seja mais rápido, mais barato e mais poderoso. O movimento gerado pela *Big Data* também tem estimulado o desenvolvimento e uso do IA. *Machine Learning* é o estágio mais avançado da IA que é a prática de usar algoritmos para coletar dados, aprender com eles, e então fazer uma determinação ou predição sobre alguma coisa no mundo. Comparando-se com a forma tradicional de se programar que é a implementação de rotinas de software, com um set específico de instruções para completar uma tarefa em particular, a máquina é “treinada” usando uma quantidade grande de dados e algoritmos que dão e ela a habilidade de aprender como executar a tarefa.

E evoluindo no tema de IA o aprendizado profundo é a área de mais rápido crescimento em inteligência artificial, ajudando os computadores a compreender dados, como imagens, som e texto. Usando vários níveis de redes neurais, os computadores agora têm a capacidade de ver, aprender e reagir a situações complexas tão bem quanto ou melhor que os humanos.

A aprendizagem profunda e as capacidades de autoaprendizagem resultam em maior precisão e processamento mais rápido, pode-se então extrair características de alto nível, não-lineares necessários para uma classificação precisa. Os dados brutos são alimentados através de redes neurais profundas, que aprendem a identificar o objeto no qual são treinados. Como referência, as redes neurais profundas são a primeira família de algoritmos no aprendizado de máquinas que não requerem *feature engineering* manual, mas aprendem por conta própria, processando e extraindo as características de alto nível a partir de dados brutos.

Em geral, a aprendizagem de máquina estabeleceu o fundamento para a aprendizagem profunda evoluir. Aprendizagem profunda herdou as principais características do modelo tradicional de aprendizagem da máquina, e evoluiu um passo adiante ensinando-se novas habilidades e ajustando as existentes [19].

Além de redes neurais artificiais, a área de inteligência artificial possui diversos outros ramos de pesquisa, como o de redes semânticas e buscas. A busca é um mecanismo genérico usado para resolução de problemas na falta de outro método mais direto. Existem dois tipos principais de buscas: buscas cegas, que realizam buscas sistemáticas sem orientação de um melhor caminho a priorizar e as buscas heurísticas, que realizam buscas sistemáticas, em que as opções a serem exploradas são ordenadas, inv estigando-se assim caminhos mais promissores primeiro.

Uma rede semântica é transformada em uma árvore de busca considerando-se todos os caminhos possíveis que não resultem em "*loops*". A árvore de busca é formada por nós, que denotam um caminho e ramos, que conectam caminhos. A figura 2.1 representa um exemplo de uma semântica e a 2.2 representa a árvore de busca correspondente à essa rede.

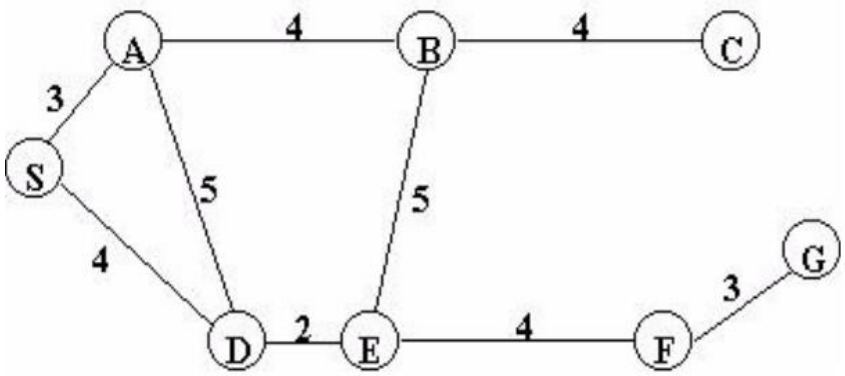

Figura 2.1: Exemplo de rede semântica

Fonte: Notas de aula de Marcos Henrique Terra disponível em [20].

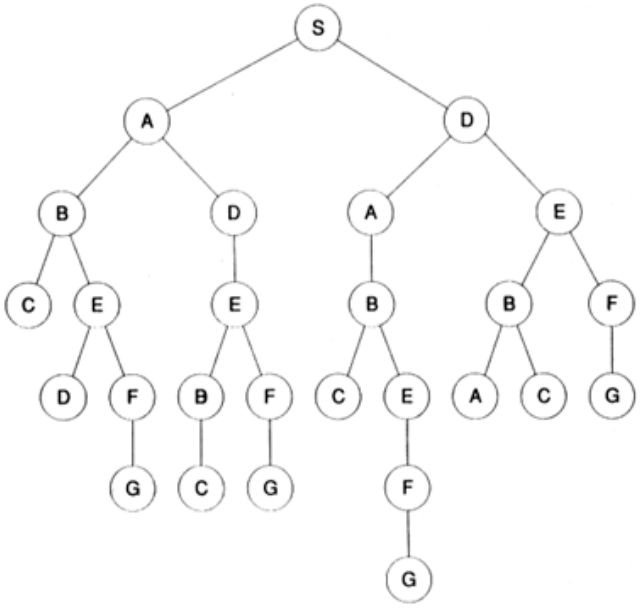

Figura 2.2: Árvore de busca correspondente a rede da Figura 2.1

Fonte: Notas de aula de Marcos Henrique Terra disponível em [20].

Dos métodos de buscas cegas podemos citar a busca em profundidade e a busca em amplitude cujo procedimento de busca é ilustrado nas figuras abaixo. A busca em profundidade funciona da seguinte forma: a busca inicia no nó raiz e explora tanto quanto possível cada um dos seus ramos, antes de retroceder (*backtracking*). Já a busca em amplitude começa pelo nó raiz e explora todos os nós vizinhos. Para cada um desses nós mais próximos, explora os seus nós vizinhos inexplorados e assim por diante, até que ele encontre o alvo da busca [21]. A figura 2.3 ilustra o processo de busca em profundidade enquanto a figura 2.4 o processo de uma busca em amplitude ou largura.

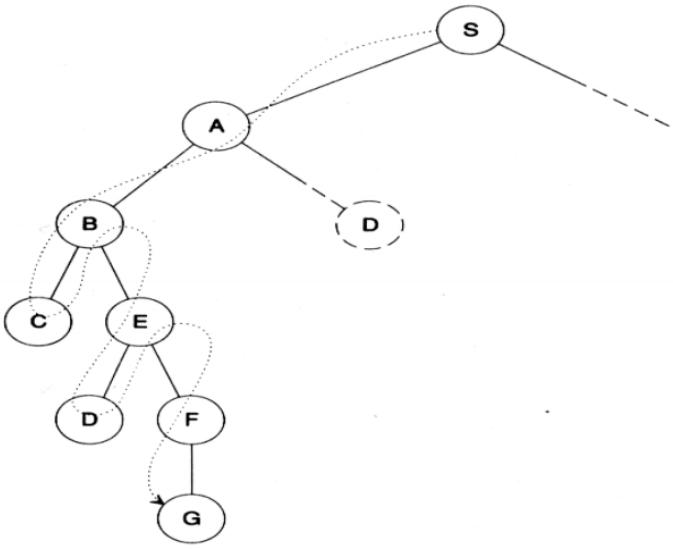

Figura 2.3: Procedimento de busca em profundidade

Fonte: Notas de aula de Marcos Henrique Terra disponível em [43].

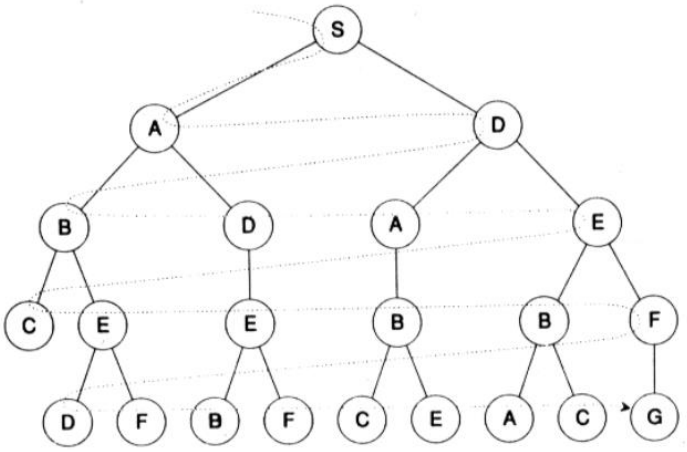

Figura 2.4: Procedimento de busca em amplitude

Fonte: Notas de aula de Marcos Henrique Terra disponível em [20].

2.5. Sistemas Embarcados

Um sistema embarcado é um sistema microprocessado que tem como propósito, diferentemente de um computador de uso pessoal, a realização de tarefas específicas. Essa característica torna o sistema embarcado muito mais eficiente e barato para executar as tarefas a ele destinadas.

O primeiro sistema embarcado que se tem conhecimento é o AGC. Era um computador que operava a 1,024 MHz e era responsável pelo total controle das espaçonaves Apollo, que levaram diversas vezes o homem a Lua nos anos 60 e 70. O AGC, no entanto, não possuía processador, era todo feito com portas NOR [22].

Atualmente, com a crescente de projetos na área de internet das coisas (IOT) e o avanço tecnológico, que aumenta o poder de processamento dos sistemas embarcados e tornam-nos mais acessíveis e baratos, seu uso vem sendo amplamente difundido.

Cada vez mais os sistemas embarcados estão conectados às redes corporativas e à Internet, vencendo uma barreira importante, pois, tradicionalmente, a sua interconexão vinha sendo isolada de outras redes. Devido a limitação presente em hardware de sistemas embarcados, a computação em nuvem pode ser aplicada nestes sistemas para ampliar o poder de processamento dos mesmos. Atualmente, com a popularidade do conceito de Indústria 4.0 essa combinação encontra-se cada vez mais presente. Os sistemas embarcados utilizadas para produção em fábricas, por exemplo, geram uma grande quantidade de dados que podem ser enviados para uma infraestrutura em nuvem, analisados por técnicas de *Big Data*, utilizadas em aplicações de gerenciamento, monitoração e *Data Mining* visando, entre outras coisas, a predição de falhas e manutenção. Dessa forma, os sistemas embarcados ficam responsáveis pelo processamento mais simples e mais imediato dos dados enquanto o restante dos dados

é enviado para a nuvem para um processamento mais lento, que necessita de um poder de processamento maior, para extrair outras informações importantes [23].

2.5.1. Arquitetura ARM

A arquitetura ARM surgiu na década de 80 e foi desenvolvida pela empresa Acorn Digital Computers que foi contratada pela BBC para desenvolver o projeto de uma nova arquitetura para o microcomputador chamado BBC Computer Literacy Project [24].

Esse projeto foi influenciado pelo conceito RISC, que ao contrário da CISC, geralmente executam as instruções em apenas um ciclo de máquina tornando-a mais rápida para uma mesma frequência de clock. Esse aumento de performance, no entanto, exige uma quantidade maior de memória para armazenar o mesmo programa [25].

As principais características dessa arquitetura são [26]:

- Uso de um grande número de registradores (Arquitetura Load/Store mútipla);
- Conjunto reduzido e simplificado de instruções e controle sobre a Unidade de Controle
- Execução de uma instrução por ciclo;
- Projeto otimizado do Pipeline de instruções para lidar com a alta proporções de desvios condicionais e chamadas a procedimentos;
- Endereçamento simplificado;
- Baixo consumo de energia;

2.5.2. Linux

O Linux é o projeto *Open Source* mais difundido no mundo na história da computação. Trata-se de um projeto de um Kernel lançado inicialmente por Linus Torvalds em 1991. O Kernel é responsável pelo gerenciamento do hardware, dos periféricos, pela execução dos programas do usuário e pela segurança e manutenção do sistema como um todo [27].

O sistema operacional (GNU/Linux) como um todo é composto por várias partes, as principais são [28]:

Bootloader: Software que é responsável pela inicialização do sistema como um todo.

Kernel: É o núcleo do sistema e gerencia a CPU, a memória e os dispositivos periféricos

Root filesystem: local em que ficam armazenadas as bibliotecas, arquivos, configurações de sistema e programas.

O nome GNU/Linux é o nome dado a união entre o Projeto GNU e o Kernel Linux [29]. Foi desenvolvido com o projeto uma série de ferramentas denominadas GNU Toolchain. Essas ferramentas incluem [30]:

GNU make: Ferramenta de compilação e build

GNU Compiler Collection (GCC): é a coleção de compiladores. gcc é o compilador em C e g++ " é o compilador para C++. Outros compiladores que estão presentes neste conjunto são o gccgo (Go), gcj (Java), gfortran (Fortran) e GNAT (Ada).

GNU Binutils: é um conjunto de ferramentas que gerencia arquivos binários como bibliotecas, arquivos de objeto (* .o), código-fonte de assembly, etc. Essas ferramentas são usadas pelo GNU Make, GCC e pelo depurador GNU.

GNU Bison: é um gerador de *parser*. Isso significa que ele lê a especificação de uma "gramática" da linguagem de programação (na forma de *tokens*) e cria um analisador. *Tokens* são *strings* que representam algum grupo especial ou parte da programação. Um gerador de léxico lê o arquivo lex e o converte em código-fonte C, que pode ser compilado para gerar um léxico. O lexer lê o código-fonte de um programa e produz código *tokenizado*. O gerador de analisador (neste caso, Bison), lê uma especificação da sintaxe da linguagem de programação e faz um analisador que verifica a estrutura de sintaxe do idioma especificado. O analisador lê o código *tokenizado* (de um léxico) para ver se a sintaxe está correta.

GNU m4: é um pré-processador de macros. Em outras palavras, antes que o código-fonte do projeto seja analisado, lexado ou compilado, o GNU m4 interpretará o código das macros. Uma macro é um código dentro do código-fonte que executa algumas tarefas no próprio código fonte antes de compilar.

GNU Debugger (GDB): é a ferramenta de depuração usada no GNU Toolchain. Muitas arquiteturas de CPU e linguagens de programação são suportadas.

GNU build system (autotools): é um conjunto de utilitários que permitem aos programadores tornar seus pacotes de código fonte portáveis para outros sistemas baseados em Unix. Este conjunto de ferramentas inclui Autoconf , Autoheader , Automake , Libtool e GNUlib .

Existem diversas distribuições que utilizam o Kernel Linux para o sistema operacional, essas distribuições são versões de sistemas operacionais e abrangem tipos completamente diferentes de usuários, pois algumas delas são mais básicas e de fácil utilização e outras já exigem um conhecimento maior de computação para sua utilização [31].

Neste capítulo foram apresentados os conceitos fundamentais utilizados de forma prática no desenvolvimento de todo projeto. O desenvolvimento desse será descrito de forma detalhada no próximo capítulo.

3. Materiais e Métodos

Neste capítulo, serão descritos os materiais e métodos usados no desenvolvimento do projeto, que tem como base um sistema embarcado Raspberry Pi.

3.1 Materiais

- Computador com software Matlab
  - ✓ Processador: Intel® Core™ I7-5500U CPU @ 2.40GHz
  - ✓ Memória RAM: 8GB
  - ✓ Sistema Operacional: Windows 10 64bits
- Webcam integrada ao computador para aquisição dos vídeos:
  - ✓ Resolução: 960x540 *pixels*
  - ✓ Formato de vídeo: mp4
  - ✓ *Frame rate*: 30fps
- Raspberry Pi 3 Model B

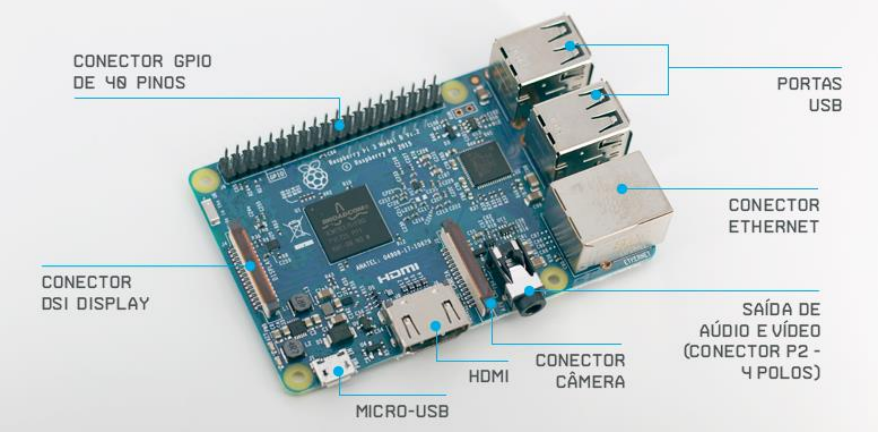

Figura 3.1: Raspberry Pi 3 Model B em detalhe

A Raspberry Pi, ilustrada pela Figura 3.1, é um sistema embarcado desenvolvido no Reino Unido pela Fundação Raspberry Pi. O principal objetivo é promover o ensino em Ciência da Computação básica em escolas, inclusão e empoderamento social [32]. É uma placa bem poderosa e de valor de custo relativamente baixo, cerca de $35, ideal para diversos projetos de IOT.

O modelo escolhido neste projeto foi a Raspberry Pi 3 Model B, na Tabela 3.1 segue uma comparação entre esse modelo e o seu antecessor. Os principais motivos para a escolha da placa foram devido ao seu alto poder de processamento e a presença de uma antena WI-FI integrada a placa que facilita a sua conexão a Internet.

Placa	Raspberry Pi 2 Model B	Raspberry Pi 3 Model B
Processador	Broadcom BCM2836	Broadcom BCM2837
CPU	Quadcore ARM Cortex-A7, 32Bit	Quadcore ARM Cortex-A53, 64Bit
Clock	900 MHz	1.2GHz
RAM	1 GB	1 GB
GPU	250 MHz VideoCore IV®	400 MHz VideoCore IV®
Conectividade de rede	1 x 10 / 100 Ethernet (RJ45 Port)	1 x 10 / 100 Ethernet (RJ45 Port)
Conectividade sem fio	-	802.11n Wireless LAN (Wi-Fi) e Bluetooth 4.1, BLE
Portas USB	4 x USB 2.0	4 x USB 2.0
GPIOs	40 pinos	40 pinos
Interface com câmera	15-pin MIPI	15-pin MIPI
Interface com display	DSI 15 Pin / HDMI Out / Composite RCA	DSI 15 Pin / HDMI Out / Composite RCA
Fonte (Corrente)	1.8 A	2.5 A

Tabela 3.1: Tabela comparativa entre modelos de Raspberry

- Monitor
- Teclado
- Mouse
- Cabo HDMI
- Cartão Micro SD 64GB
- Adaptador USB de Cartão Micro SD
- Roteador ARRIS
    - ✓ Modelo: TG1692A

3.2. Softwares

3.2.1. MATLAB

O MATLAB é um software interativo que se destina a cálculos numéricos e gráficos científicos, desenvolvido no fim dos anos 70 por Cleve Moler. O MATLAB integra análise numérica, cálculo com matrizes, processamento de sinais e construção de gráficos em ambiente fácil de usar onde problemas e soluções são expressos somente como eles são escritos matematicamente. Seu ponto forte está na manipulação e cálculos matriciais, o que o torna uma ferramenta muito útil para processamento de imagens. Ele possui diversas extensões, ou *toolboxes*, com funções prontas para diversas áreas. Uma delas é a de processamento de imagens e visão computacional. Seu uso é indicado para implementar e

testar soluções com facilidade e precisão, sem perder tempo com detalhes específicos de linguagem de programação, e por isso é muito utilizado para aplicações que necessitam de prototipagem rápida [33].

3.2.2. OpenCV

Desenvolvida pela Intel, em 2000, é uma biblioteca multiplataforma e possui mais de 2500 algoritmos otimizados. Totalmente livre ao uso acadêmico e comercial, para o desenvolvimento de aplicativos na área de Visão computacional, bastando seguir o modelo de licença BSD Intel.

Foi idealizada para tornar visão computacional acessível para programadores e possuem diversas aplicações, como reconhecer faces, identificar objetos, detectar movimentos, seguir movimento dos olhos entre muitas outras funções já implementadas. Essa biblioteca está disponível para diversas linguagens de programação como C, C++, JAVA e PYTHON, porém seu melhor desempenho é na linguagem C e C++, pois as operações matemáticas podem ser executadas com paralelismo em computadores com mais de um núcleo [34]. Esta biblioteca está dividida em diversos módulos, que separam os algoritmos de estruturas básicas, algoritmos de aquisição de vídeo, de processamento de imagens, e outras funções [35], e agora apresenta a aceleração de GPU para operação em tempo real.

A Unidade central de processamento (CPU) é o processador de um computador e influencia em basicamente todos os recursos do computador desde o processamento de dados, a memória e os cálculos dos seus eletrônicos. A CPU é o cérebro do seu computador.

A unidade de processamento gráfico GPU também conhecida como VPU (unidade de processamento visual) é o processador da sua placa de vídeo. Apesar das novas tecnologias permitirem que as GPUs façam mais do que processar somente vídeo, sua principal função ainda continua sendo esta.

A CPU pode fazer qualquer tipo de cálculo de processamento, incluindo os gráficos, porém, o processo através da CPU seria lento demais e por isto é utilizado um processador específico para esta função. A GPU além de realizar a tarefa do processamento gráfico, pode ser utilizada para processar soluções analíticas de *Big Data*, embora muitas pessoas que trabalham com tecnologia acreditarem que seja muito exótico a escolha de GPU's como opção de hardware para processar aplicações de *Analytics*, porém fato é que cada vez mais as GPU's tem sido usadas para análises avançadas utilizando algoritmos de *Machine Learning*, especialmente *Deep Learning*.

A NVIDIA percebeu que as GPU's, tipicamente utilizadas para renderização de imagens poderiam ser usadas para ajudar a resolver problemas matemáticos que requerem computação intensiva. Para isto a NVIDIA desenvolveu as GPU's para uso geral e a linguagem de programação CUDA, permitindo que os poderosos processadores gráficos sejam usados para processar dados. Desde sistemas embarcados até utilização por usuários domésticos ou mesmo supercomputadores. Suas aplicações incluem: restauração de imagens, segmentação, filtragem, interpolação e reconstrução. GPU's são chips

de computador que realizam cálculos matemáticos de forma veloz e paralelizada com boa relação de custo benefício [36].

3.2.3. Raspbian

Raspbian é uma distribuição de Linux livre baseado no Debian otimizado para o hardware Raspberry Pi. O Raspbian possui mais de 35.000 pacotes inclusos, software pré-compilado de fácil instalação. Foi concluída em junho de 2012. No entanto, ainda está em desenvolvimento ativo com ênfase em melhorar a estabilidade e o desempenho de tantos pacotes Debian quanto possível [37].

3.3. Métodos

3.3.1. Desenvolvimento em MATLAB

O primeiro passo para o desenvolvimento do projeto foi criar os algoritmos para resolver o problema na plataforma Matlab. Essa plataforma foi escolhida pelo motivo de já possuir bibliotecas para tratamento de imagens e por facilitar a prototipagem rápida da solução, já que se trata de uma linguagem interpretada e não compilada, e pelo fato de não ser necessário a preocupação com os tipos e dimensionamento das variáveis, o que facilita muito a escrita do programa. O fluxograma geral da solução proposta pelo projeto é mostrado na Figura 3.2.

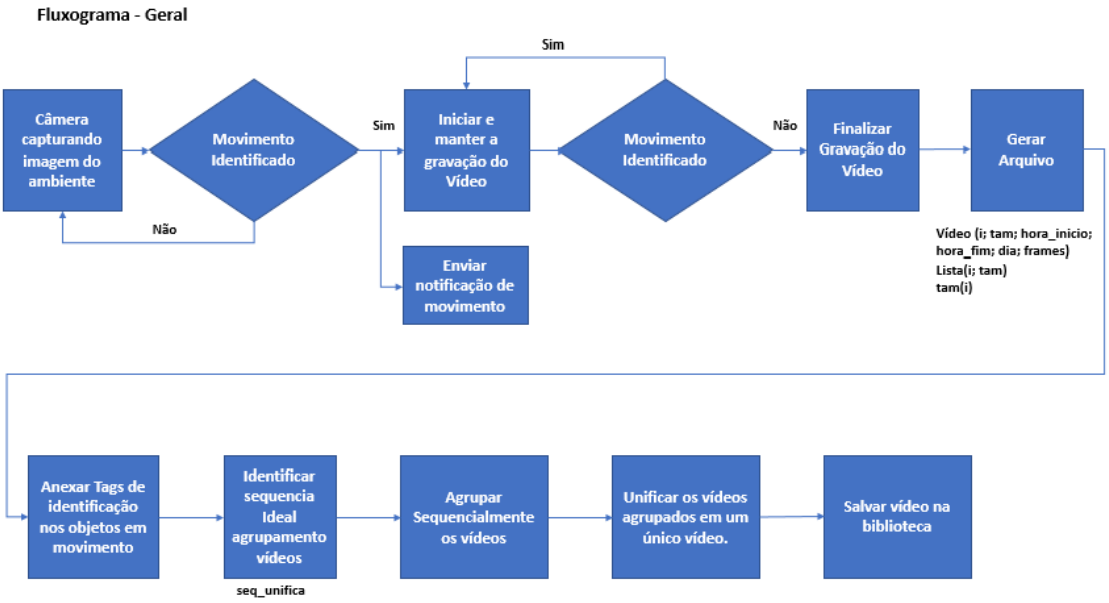

Figura 3.2: Fluxograma geral do projeto

Primeiramente, gravou-se um vídeo teste com uma webcam de alguns momentos de movimento e outros em que se tinha apenas o fundo estático e foi desenvolvido um algoritmo que detecta esses momentos de movimento e salva-os separadamente em uma pasta que contém todos movimentos detectados.

Existem diversas maneiras de detectar movimento, como por exemplo, subtração de fundo, fluxo óptico, diferença temporal. O algoritmo utilizado para o desenvolvimento do projeto foi o de diferença temporal, pois o algoritmo de subtração de fundo não se mostrou eficiente devido a grandes variações de luminosidade de um ambiente externo e o método que utiliza fluxo óptico exigiria um custo computacional muito alto que poderia trazer complicações quando fosse aplicado em um sistema embarcado limitado.

O algoritmo de diferença temporal consiste em calcular a diferença dos *pixels* de dois *frames* subsequentes. Para reduzir o ruído inerente na imagem, foi aplicado um filtro de média antes de calcular a diferença dos dois *frames*. Depois de calculada a diferença, ela foi binarizada e foi aplicado um filtro de mediana para eliminar possíveis ruídos sal e pimenta, a imagem formada por esse processo era, portanto, uma imagem em preto e branco, em que os *pixels* em branco representam *pixels* em que ocorreram mudanças significativas. Caso o número de *pixels* detectados seja maior que um número estipulado para que não fossem detectados ruídos remanescentes, o algoritmo passa a gravar os *frames* até que o programa não detecte mais nenhum movimento por um segundo seguido. Quando isso ocorre, o algoritmo salva o vídeo de movimento em uma pasta separada que contém todos os movimentos detectados e continua buscando por outros movimentos até que o vídeo que está sendo analisado termine e todos tenham sido detectados e gravados.

Depois do algoritmo de detecção de movimento, foi desenvolvido um algoritmo para calcular as regiões de movimento em cada *frame* e criar um arquivo de texto com a posição central de todas as regiões de movimento de cada *frame* para todos os vídeos de movimento arquivados na pasta separada. Esses arquivos serão utilizados posteriormente para adicionar uma *tag* com o horário do movimento em cada uma dessas regiões.

O algoritmo funciona da seguinte maneira: novamente ele utiliza a técnica de diferença temporal e calcula a diferença entre o *frame* atual e um *frame* a três quadros de distância, esse padrão foi escolhido para que os *pixels* em movimento ficassem mais destacados, depois aplica um filtro de média, binariza o resultado, aplica o filtro de mediana e por fim aplica um filtro que detecta bordas. Esse último filtro foi aplicado para reduzir o número de *pixels* analisados para calcular as regiões em movimento. Um exemplo do que acontece pode ser visto nas imagens a seguir. A Figura 3.3 mostra um *frame* do teste, enquanto a Figura 3.4 apresenta o resultado parcial do algoritmo em cima desse *frame*.

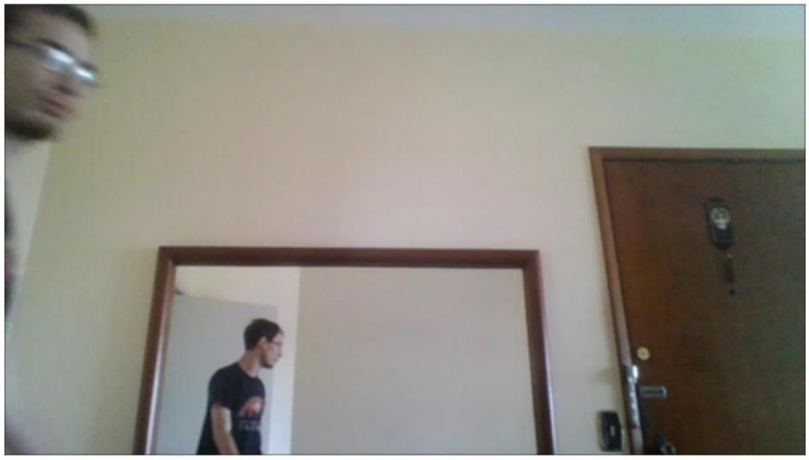

Figura 3.3: *Frame* de vídeo teste

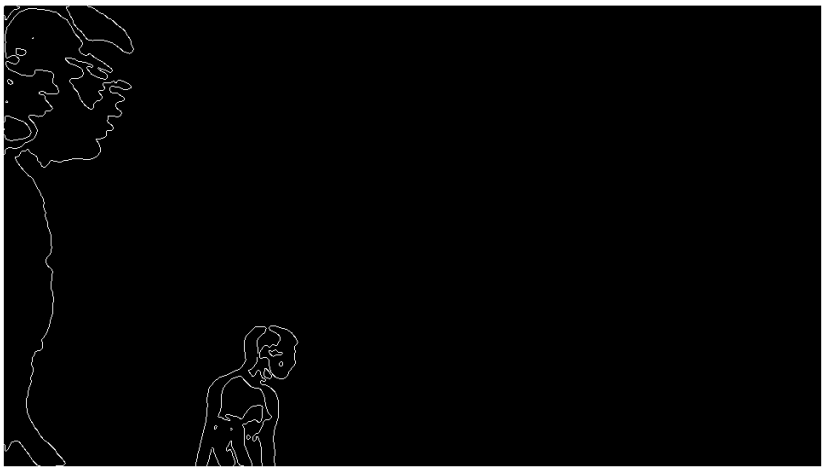

Figura 3.4: Saída parcial produzida pelo algoritmo

Com esse resultado, conectamos os *pixels* brancos detectados formando a região que se encontra cada movimento. O algoritmo conecta todos os *pixels* que estão próximos, em uma distância de até 60 *pixels* em qualquer direção do *pixel* branco mais próximo. No caso desse *frame* as regiões detectadas são mostradas na Figura 3.5.

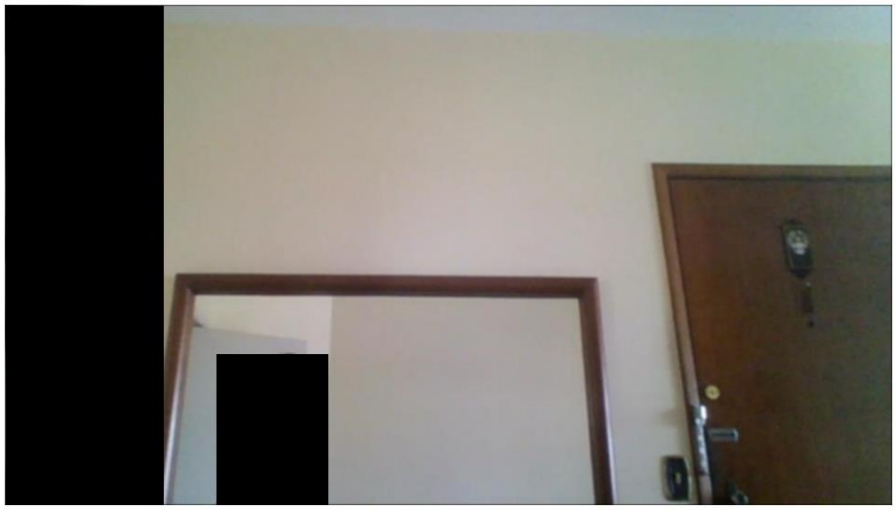

Figura 3.5: Regiões de movimento detectadas

Neste caso, foram detectadas duas regiões de movimento separadas e, portanto, será armazenado no arquivo de texto, a posição do centro de cada uma das regiões e a hora do movimento para esse *frame*. Esse processo é repetido até que todos os movimentos sejam verificados e que sejam criados arquivos de texto para cada um dos vídeos da pasta que continha todos os movimentos detectados. Um exemplo de como fica o arquivo de texto para um dos movimentos pode ser visto na Figura 3.6 apresentada abaixo.

Pos x	Pos y	Hora	Min	Pos x	Pos y	Hora	Min
502	607	13	46	462	836	13	46
496	624	13	46	485	858	13	46
491	600	13	46	479	850	13	46
462	596	13	46	456	829	13	46
456	590	13	46	346	862	13	46
452	586	13	46	331	846	13	46
431	648	13	46	317	844	13	46
364	736	13	46				
356	731	13	46				
356	725	13	46				
354	722	13	46				
356	717	13	46				
357	713	13	46				
359	708	13	46				
363	704	13	46				
362	700	13	46				
362	696	13	46				
360	686	13	46				
359	671	13	46				
358	656	13	46				
358	640	13	46				
358	620	13	46				
358	585	13	46				
358	562	13	46				
358	521	13	46				
357	494	13	46				
357	457	13	46				
356	428	13	46				
353	391	13	46				
350	361	13	46				
346	340	13	46				
342	324	13	46				
339	290	13	46				
336	276	13	46				
330	262	13	46				
327	246	13	46				
324	230	13	46				
322	211	13	46				
320	196	13	46				
319	188	13	46				
316	196	13	46				
313	180	13	46				
312	175	13	46				
312	175	13	46				
313	175	13	46				
312	174	13	46				
312	173	13	46				
320	24	13	46	448	260	13	46
354	14	13	46	449	256	13	46
325	15	13	46	449	251	13	46
448	245	13	46				
449	231	13	46				
449	229	13	46				
450	226	13	46				
450	221	13	46				
451	222	13	46				
452	216	13	46				
452	220	13	46				
451	217	13	46				
448	203	13	46				
448	201	13	46				

Figura 3.6: Arquivo de texto para vídeo contendo movimento

Cada linha representa o *frame* do vídeo, a primeira e a segunda coluna indicam a posição central da região de movimento nas coordenadas x e y respectivamente, a terceira e a quarta coluna representam a hora e o minuto em que o movimento ocorreu. No caso de mais de uma região de movimento por *frame*, os valores são armazenados na mesma linha, porém nas colunas seguintes conforme pode-se observar pela figura acima.

O próximo algoritmo desenvolvido serve para formar as sequências de movimento que serão sobrepostas. Inicialmente, tentou-se sobrepor todos os movimentos em cima do movimento mais longo detectado usando uma técnica de transparência. Porém, notou-se que conforme a quantidade de movimentos detectados fosse aumentando, uma perda significativa de qualidade foi observada, ficando

impossível observar o que estava acontecendo na cena. Através de testes, foi identificado que o número aproximado de sobreposições que podia conter em um *frame* para não haver perda significativa de informação era entre 4 e 7 sobreposições, dependendo do caso. Para isso, no caso de serem detectados um número "n" elevado de movimentos é necessário um algoritmo para formar sequências de tal forma que o número máximo de sobreposições em um único *frame* não ultrapasse esse valor experimental.

O planejamento para o desenvolvimento desse algoritmo foi o seguinte:

Dado os vídeos dos movimentos detectados no ambiente monitorado, cada um contendo as seguintes informações: Vídeo (i, tam, *frames*)

Onde:

- i → é o número sequencial identificador do vídeo
- tam → é o tamanho (quantidade de *frames* do vídeo (i)
- *frames* → são os *frames* do vídeo (i).

Podemos calcular o número total de *frames* que continham movimento através da somatória dos *frames* dos movimentos individuais. Esse cálculo é mostrado na equação 3.1.

$$\mathbf{Total_Frame} = \sum_{i=1}^{n} tam(i) \qquad (3.1)$$

Também podemos obter a média do número de *frames* de cada vídeo de movimento com a fórmula 3.2.

$$\mathbf{Media_Tam_Frame} = \frac{\mathbf{Total_Frame}}{n} \qquad (3.2)$$

Além disso, podemos estimar o número de *frames* que cada sequência deverá ter, de forma que o total de "n" vídeos formem no máximo nmáx (número máximo de sobreposições que deve ocorrer em um *frame*) sequências e de forma que exista uma margem de tamanho para as sequências, mas os tamanhos das sequências sejam parecidos. A equação 3.3 calcula o tamanho que deverá ter idealmente cada sequência.

$$Tam_Corte = int\left(\frac{\frac{Total_{Frame}}{nmax - 1} + 3 \times \frac{Total_{Frame}}{nmax}}{4}\right) \qquad (3.3)$$

O erro aceitável de tamanho para cada sequência foi fornecido pela fórmula 3.4.

$$erro = \frac{(Tam_Corte \times nmáx - Total_Frame)}{nmax + 1} \qquad (3.4)$$

Por fim, utilizou-se o conceito de árvore de busca, onde cada nó é um dos “n” vídeos, sendo que o vetor de vídeos está indexado do maior para o menor. A ideia do algoritmo foi combinar primeiro os vídeos maiores e, por estarem ordenados do maior pro menor, foi aplicado o conceito de busca em amplitude. Dessa forma, começa se a abertura da árvore no nó raiz com o vídeo de maior tamanho. No primeiro nível estão os (n-1) vídeos conectados ao nó raiz, e assim por diante. A busca será encerrada quando a soma dos vídeos que estão no caminho atingir um valor entre Tam_Corte e Tam_Corte – erro. Caso o valor da soma dos vídeos que estão no caminho ultrapasse o valor de Tam_Corte a árvore a partir desse nó não e mais aberta. Quando a busca é encerrada, retira-se os vídeos do caminho encontrado do vetor dos vídeos, salva-os em um arquivo de vídeo separado unificando-os e repete-se o procedimento com os vídeos restantes no vetor, até que não exista mais nenhum vídeo. Para o vídeo teste que foi gravado inicialmente foram detectados 13 movimentos, cada vídeo foi representado por uma letra de a até m seguindo a ordem alfabética. O resultado desse algoritmo para esse exemplo, utilizando nmáx = 5 pode ser mostrado pelas Figuras 3.7, 3.8, 3.9, 3.10 e 3.11.

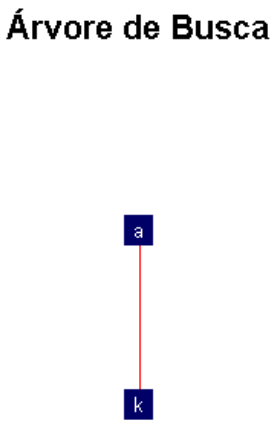

Figura 3.7: Árvore de busca para primeira sequência de vídeos

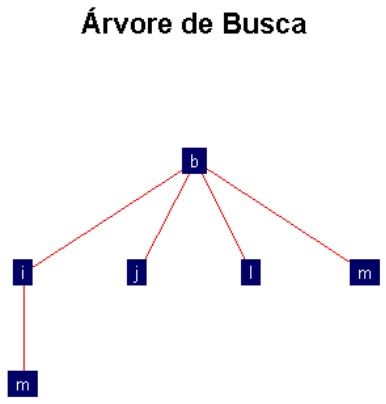

Figura 3.8: Árvore de busca para segunda sequência de vídeos

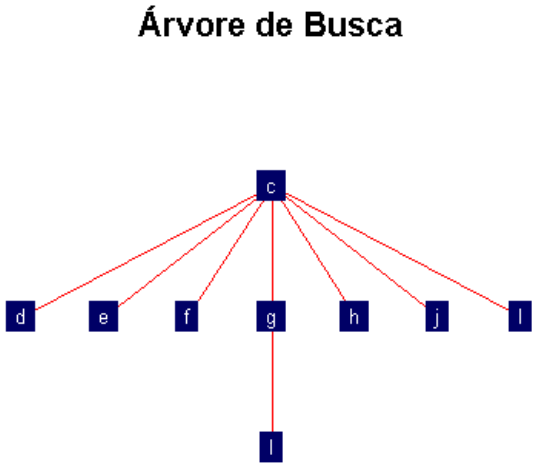

Figura 3.9: Árvore de busca para terceira sequência de vídeos

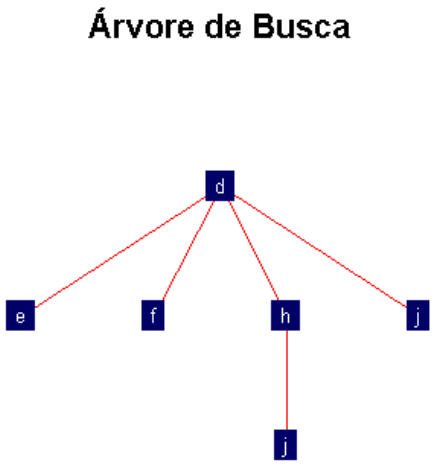

Figura 3.10: Árvore de busca para quarta sequência de vídeos.

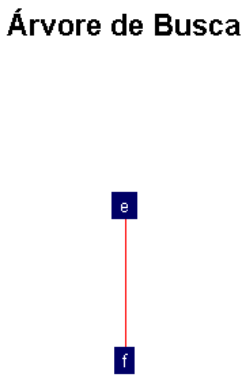

Figura 3.11: Árvore de busca para quinta sequência de vídeos.

Formou-se, portanto, 5 sequências: a-k, b-i-m, c-g-l, d-h-j, e-f.

Por fim, foi desenvolvido um algoritmo que possui como entrada essas sequências e utilizando a técnica de transparência, sobrepõe os movimentos em um único vídeo final que resume todo o movimento ocorrido no vídeo teste. Esse algoritmo teve como base um algoritmo já existente no Matlab chamado “imfuse” com apenas algumas alterações para que fosse possível escolher o nível de

transparência das sobreposições. Depois que a sobreposição é feita, adiciona-se as *tags* com os horários em cada uma das regiões de movimento.

3.3.2. Desenvolvimento em C++

Depois de implementados e testados os algoritmos desenvolvidos na plataforma Matlab, partiu-se para o desenvolvimento dos algoritmos na linguagem C++, utilizando a biblioteca OpenCV. Primeiro foi necessário fazer as configurações iniciais na Raspberry Pi.

O primeiro passo foi formatar o Cartão Micro SD, para isso foi utilizada a ferramenta DiskPart do Windows e o adaptador USB do cartão. Os comandos executados no prompt do Windows foram os descritos a seguir:

> diskpart (Abre o utilitário)

> list disk (Listará as unidades no seu computador)

> select disk 1 (Onde 1 é o Cartão Micro SD que será formatado)

> clean (Limpa o cartão)

> select disk 1

> create partition primary (Cria uma única partição)

> format quick fs=FAT32 (Formata a partição em formato FAT32)

> exit (Sair do Diskpart)

> exit (Sair do CMD)

Depois de formatado o cartão, foi feito o download da versão 3.0.1 do Noobs em ZIP disponível em [38].

O **Noobs** é uma das formas mais simples e práticas de instalar um sistema operacional para o Raspberry, pois é um pacote que contém várias versões do Sistema Operacional, com uma interface que facilita o processo de instalação.

Os arquivos do Noobs foram descompactados na raiz do Cartão Micro SD que foi formatado.

Feito isso, a Raspberry foi ligada ao monitor através do cabo HDMI, além disso o mouse e teclado foram ligados a porta USB da mesma, por fim o cartão SD, com os arquivos do SO, foi inserido e a fonte de alimentação foi conectada. Neste momento os arquivos de inicialização são carregados e o processo de instalação se inicia. A Figura 3.12 apresenta a tela de instalação inicial que é carregada no monitor.

Figura 3.12: Tela de instalação do sistema operacional

Primeiro foi configurada a rede Wifi e depois o sistema operacional escolhido Raspbian foi instalado. O processo demora alguns minutos e após a instalação o Raspbian automaticamente reinicia o sistema.

Quando o sistema é inicializado é preciso configurar linguagem, usuário e senha. Depois de realizados esses procedimentos, partiu-se para instalação dos softwares necessários para fazer acesso remoto a Raspberry e dessa forma não precisar mais do monitor, teclado e mouse. Para esse acesso, foi necessário fazer a instalação do software VNC server na Raspberry que a ser controlada remotamente, esse software está disponível em [39], além disso no computador deve ser instalado o VNC Viewer disponível em [40] e o WinSCP disponível em [41]. O VNC Viewer é utilizado para fazer o acesso remoto e o WinSCP é o software utilizado para fazer transferências de arquivos via SSH. Para possibilitar as duas formas de acesso à Raspberry foi necessário entrar em Menu > Preferências > Raspberry Pi Configuration > Interfaces e habilitar as opções SSH e VNC. Além disso, por questão de segurança a porta de acesso SSH foi alterada da porta padrão 22 para a 22000 seguindo o seguinte procedimento:

$ cd /etc/network/ssh

$ sudo nano sshd_config

O primeiro comando acessa o diretório em que ficam os arquivos de configuração de SSH, enquanto o segundo abre o arquivo de configuração como administrador para que ele possa ser alterado. Neste arquivo foi alterado o valor da porta conforme mostra Figura 3.13.

```
  GNU nano 2.7.4                    Arquivo: sshd_config

#       $OpenBSD: sshd_config,v 1.100 2016/08/15 12:32:04 naddy Exp $

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Port 22000
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
                              [ 123 linhas lidas ]
^G Ajuda     ^O Gravar    ^W Onde está?^K Recort txt^J Justificar^C Pos atual
^X Sair      ^R Ler o arq ^\ Substituir^U Colar txt ^T VerfOrtog ^_ Ir p/ linha
```

Figura 3.13: Arquivo de configurações de SSH.

Para facilitar o acesso remoto, fixou-se no roteador o IP local da Raspberry e foi liberado no Firewall do roteador as portas para permitir acesso remoto ao dispositivo. O detalhamento desse procedimento pode ser encontrado no Apêndice B.

Com esses procedimentos executados, foi possível fazer acesso remoto utilizando o IP público de qualquer lugar, o que tornou mais simples o acesso ao sistema embarcado para desenvolver os algoritmos em C++.

Para o desenvolvimento dos algoritmos em C++ foi utilizado a biblioteca OpenCV que teve que ser instalada. Para fazer a instalação da biblioteca, é importante atualizar os pacotes que já estão instalados no sistema. Para isso foram executados os comandos a seguir:

$ sudo apt-get update

$ sudo apt-get upgrade

O primeiro comando atualiza a lista de pacotes disponíveis e suas versões, enquanto o segundo atualiza todos os pacotes encontrados no comando anterior.

Feito isso foi necessário instalar algumas dependências da biblioteca e, portanto, foram executados vários comandos descritos a seguir:

$ sudo apt-get install build-essential cmake pkg-config

O comando acima instala ferramentas para executar o *build* de um pacote (build-essential e cmake) e o uma ferramenta que auxilia na compilação de bibliotecas.

$ sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev

Neste segundo comando são instalados pacotes para que o OpenCV consiga lidar com tipos distintos de extensões de imagens, como PNG, JPEG, etc.

$ sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev

$ sudo apt-get install libxvidcore-dev libx264-dev

Nos dois comandos acima são instalados pacotes para que o OpenCV possa trabalhar com arquivos de vídeo, acessar *frames* individuais, processar a transmissão entre outras funcionalidades.

$ sudo apt-get install libgtk2.0-dev libgtk-3-dev

O comando anterior foi executado para instalar os pacotes que o módulo *highgui* do OpenCV depende para associar a sua Interface Gráfica ao Usuário (GUI), e dessa forma ser capaz de mostrar as imagens na tela.

$ sudo apt-get install libatlas-base-dev gfortran

Esse comando instala pacotes que optimizam várias rotinas internas do OpenCV.

$ sudo apt-get install python3-dev

$ sudo apt-get install python3-pip

$ sudo pip3 install numpy scipy

Por fim, esses três comandos instalam ferramentas para fazer o OpenCV conseguir ser executado em python, uma ferramenta que gerencia e instala pacotes escritos em python e por último, instala bibliotecas para manipulação de arrays em python de forma optimizada, no caso do projeto, não foram utilizados e foram usados apenas para compilação completa da biblioteca.

Com todas as dependências instaladas foi feito o *download* dos repositórios do OpenCV a ser instalados executando os comandos abaixo:

$ wget -O opencv.zip https://github.com/opencv/opencv/archive/3.2.0.zip

$ wget -O opencv_contrib.zip https://github.com/opencv/opencv_contrib/archive/3.2.0.zip

$ unzip opencv.zip

$ unzip opencv_contrib.zip

Depois de descompactados os arquivos, basta criar um diretório para fazer o *build* da biblioteca usando os comandos a seguir:

$ cd ~/opencv-3.2.0/

$ mkdir build

$ cd build

E por fim configurar o build usando o cmake.

```
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
  -D CMAKE_INSTALL_PREFIX=/usr/local \
  -D INSTALL_PYTHON_EXAMPLES=ON \
  -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.2.0/modules \
  -D BUILD_EXAMPLES=ON ..
```

Por fim basta compilar o OpenCV usando o comando make:

$ make

Neste comando poderia se adicionar um atributo -j4 para fazer com que os quatro núcleos para fazer o processamento da compilação em paralelo e acelerar o processo, porém quando utilizado a

compilação apresentou erros. Usando apenas um dos núcleos o processo demora algumas horas para ser finalizado.

Depois de finalizado basta instalar o OpenCV usando os dois comandos a seguir:

$ sudo make install

$ sudo ldconfig

E por fim, reiniciar o sistema operacional:

$ reboot

Após a inicialização do sistema, começaram a ser desenvolvidos os algoritmos usando a biblioteca. O fluxograma seguido foi o mesmo que o proposto para ser desenvolvido no Matlab. Porém, devido as diferenças entre as linguagens, foi necessário fazer algumas alterações no código. As principais implementações novas que foram necessários serão detalhadas a seguir.

Primeiramente, foi necessário criar duas funções para contar quantos movimentos existiam na pasta. Em Matlab já existe uma função que conta o número de arquivos dentro de um diretório. O código das funções pode ser visto abaixo.

```
bool FileExistenceCheck(const string& name) {
    struct stat buffer;
    if((stat(name.c_str(), &buffer) == 0))
        cout << name.c_str() << endl;
    return (stat(name.c_str(), &buffer) == 0);
}

int fileCounter(string dir, string prefix, string extension)
{
    int returnedCount = 0;
    int possibleMax = 5000000; //some number you can expect.
    string letra = "a";
    for (int istarter = 0; istarter < possibleMax; istarter++) {
        string fileName = "";
        fileName.append(dir);
        fileName.append(prefix);
        fileName.append(letra);
        fileName.append(extension);
        bool status = FileExistenceCheck(fileName);
        returnedCount = istarter;
        letra[0] = letra[0]+1;
        if (!status)
```

```
                break;
        }
        return returnedCount;
}
```

Basicamente, a função busca dentro do diretório, arquivos específicos que seguem uma regra previamente definida, no caso do projeto a função foi utilizada para buscar os arquivos com o nome mova.avi, movb.avi, movc.avi e assim por diante e dessa forma contar quantos movimentos foram detectados.

Outra mudança que teve que ser feita foi na leitura do arquivo de texto das *tags* que gera matrizes esparsas. O Matlab não tem problema em lidar com essas matrizes, que possuem índices sem nenhum conteúdo dentro, já em C++ foi preciso criar formas para lidar com essas matrizes. A forma encontrada para lidar com essas matrizes foi a mostrada abaixo.

```
while(!file.eof()){
                count2 = -1;
                count1 = count1 + 1;
                getline(file,line);
                if(line.length()){
                        stringstream iss(line);
                        while(iss >> valor){
                                count2 = count2 + 1;
                                sparse.push_back(i);
                                sparse.push_back(count1);
                                sparse.push_back(count2);
                                sparse.push_back(valor);
                                tags.push_back(sparse.t());
                                sparse.release();
                        }
                }
        }
```

O valor i indicava o número do movimento ao qual o dado pertencia, o valor count1 representa a linha, ou *frame*, ao qual o dado pertencia e o valor count2 indicava a coluna dentro dessa linha ao qual o dado pertencia e por fim, a última coluna dessa matriz equivale ao valor do dado. Depois da leitura do txt dessa forma, bastou criar maneiras de lidar com os dados dessa nova forma.

O restante do código teve poucas alterações, apenas foi necessário usar funções do OpenCV que executavam a mesma tarefa que as funções do Matlab.

Para compilar os códigos basta executar o comando a seguir:

$ g++ Script.cpp -o Script `pkg-config --cflags --libs opencv`

Onde g++ é o compilador C++, Script.cpp é o arquivo que roda todos os algoritmos desenvolvidos para o projeto em C++ e Script é o nome do arquivo de saída.

Por fim o OpenCV foi instalado no computador também para ser possível fazer testes de performance e fazer comparações entre o processamento no embarcado e o processamento no computador, além de comparar o código feito com a biblioteca OpenCV em C++ com os códigos feitos no Matlab.

Para a instalação no PC, utilizou-se o arquivo exe com a biblioteca disponível em [42]. Depois de executar o exe e extrair os arquivos da biblioteca basta abrir o Visual Studio e criar um novo projeto.

Neste projeto, foram incluídos os mesmos algoritmos desenvolvidos para o sistema embarcado, fazendo apenas as alterações necessárias do local em que o vídeo foi lido e os resultados seriam salvos.

Depois de incluídos esses arquivos no projeto, partiu-se para inclusão do OpenCV neste projeto do Visual Studio. Nas propriedades do projeto é necessário primeiramente verificar se a plataforma escolhida está em x64. Posteriormente, dentro de C/C++ selecionar a opção Geral e inserir o caminho onde estão os arquivos do OpenCV dentro de “Diretórios de Inclusão Adicionais”. No caso do projeto o caminho dos arquivos foi “C:\opencv\build\include”. Feito isso, ainda é necessário incluir nas configurações gerais do vinculador os arquivos lib do OpenCV em “Diretórios de Bibliotecas Adicionais”. No caso do projeto os arquivos estavam no diretório “C:\opencv\build\x64\vc14\lib”. Por fim, nas configurações de entrada do vinculador é necessário incluir a biblioteca de debug “opencv_world320d.lib”. Feito isso, o projeto está pronto para ser compilado. Para executar o executável gerado, é necessário incluir nas variáveis de ambiente do sistema o diretório “C:\opencv\build\x64\vc14\binC:\opencv\build\x64\vc14\bin” que contém as dll.necessárias para executar o programa.

Neste capítulo foram descritos os materiais e métodos utilizados para execução do projeto, que posteriormente foi testado para verificar se o objetivo do trabalho foi concluído. No próximo capítulo serão apresentados os resultados e discussões dos testes realizados.

4. Resultados e Discussões

O primeiro teste realizado com o algoritmo foi verificar os resultados visuais e o tempo de processamento para o vídeo exemplo que foi usado como base para construção do algoritmo no Matlab. A seguir, são apresentadas as Figuras 4.1 e 4.2 que mostram dois *frames* distintos do vídeo gerado pelo algoritmo. Nestes exemplos, é possível enxergar com clareza todos os movimentos sobrepostos e verificar o horário que aconteceu cada um desses movimentos.

.

Figura 4.1: Exemplo 1 de *frame* do vídeo final gerado pelo algoritmo.

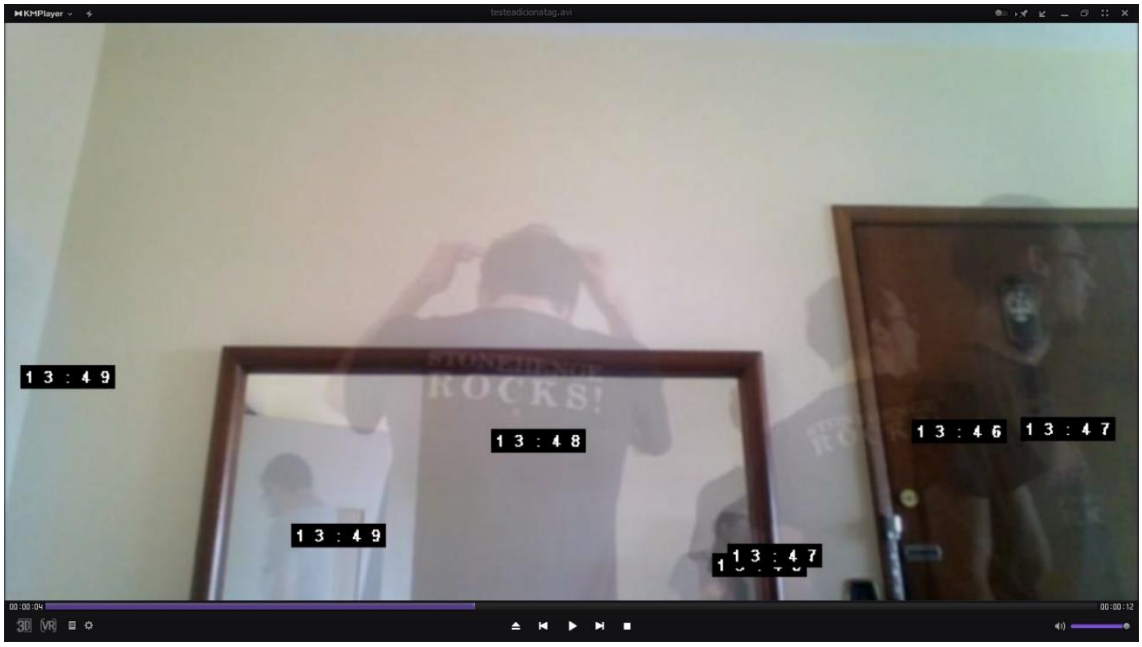

Figura 4.2: Exemplo 2 de *frame* do vídeo final gerado pelo algoritmo

O algoritmo foi processado no PC utilizado durante todo o desenvolvimento do projeto, que não possui uma rotina de alta prioridade para esse processamento. O tempo para que todos os procedimentos fossem executados foi de aproximadamente 20 minutos para um vídeo teste de praticamente 4min de duração com taxa de 30fps. Esse tempo era maior, mas as otimizações feitas nos algoritmos como a redução temporária do número de *pixels* das imagens para fazer alguns cálculos, substituição de funções demoradas por funções mais eficientes conseguiram reduzir bastante o tempo de processamento, porém esse continuava alto. Isso deve-se a falta de otimização do Matlab para esse tipo de processamento e a complexidade inerente desse problema. Na Figura 4.3 é mostrado o tempo para execução de todos os procedimentos e os algoritmos que levam mais tempo para serem executados.

Nome da Função	Chamadas	Tempo	Tempo próprio
Script	1	1195.261 s	0.285 s
DetectaMov	1	494.477 s	12.085 s
Sobrepoe	1	409.426 s	0.982 s
imfuser	1351	290.239 s	0.885 s
VideoReader.read	15198	287.410 s	279.338 s
AdicionaTags	1	225.134 s	95.252 s
imfuser>calculateOverlayImages	1351	218.300 s	3.241 s
resampleImageToNewSpatialRef	2702	176.739 s	53.192 s
medfilt2	10055	110.880 s	0.861 s
ordfilt2	10055	103.279 s	2.715 s
imfilter	21824	101.466 s	5.749 s
...er.VideoWriter>VideoWriter.writeVideo	4181	93.895 s	8.490 s
images\private\ordf (MEX-file)	10055	88.647 s	88.647 s
conversor	1984	81.995 s	1.806 s

Figura 4.3: Figura com tabela de tempo de processamento dos principais algoritmos.
Fonte: MATLAB

Nota-se que existe muito tempo gasto para ler *frames* e salvar os vídeos (aproximadamente 30% do tempo total de execução), dificultando assim o processo de redução do tempo total através de otimizações no código.

O vídeo teste de 4 minutos possui 150 mb de tamanho em disco e o vídeo final produzido pela sobreposição de todos os movimentos possui o tamanho de apenas 10mb. Dessa forma, foi possível

reduzir em 15 vezes o tamanho do vídeo. Esse número pode ser ainda maior dependendo do número de sobreposições que forem feitas num mesmo *frame*. No caso do exemplo utilizado foram de 5 sobreposições simultâneas, porém o aumento de sobreposições passa a prejudicar a visibilidade do produto final.

Neste teste, verificou-se também uma variação muito grande no local da *tag* com o horário, essa variação era causada por posicionar a *tag* sempre no centro do movimento do *frame* analisado. Essa constante variação gera desconforto ao ver o vídeo e por isso foi feita uma modificação que caso a posição central de um *frame* não variasse de 40 *pixels* em qualquer direção em relação ao *frame* anterior, o local da *tag* final era mantido neste *frame*, além disso após esse teste retirou-se a função imfuse que consumia muito tempo de processamento e a substitui por uma simples soma ponderada entre os *frames* reduzindo o tempo da função Sobrepoe aproximadamente pela metade, essa modificação reduziu o tempo de processamento sem comprometer em nada o resultado final.

Feito esse teste, partiu-se para um teste de performance entre as linguagens e a plataforma utilizada no processamento, para esse teste foram gravados três vídeos com uma taxa de aproximadamente 16fps, com durações e constância na presença de movimento distintas. Os resultados do tempo de processamento desses exemplos podem ser vistos na Tabela 4.1.

Tempo de vídeo (s)	Frames totais	Frames em Movimento	Tempo C++ PC (s)	Tempo Matlab PC (s)	Tempo C++ Raspberry (s)
157)s5	26248	1073	660,56	1609,56	3758,54
884	14732	885	417,70	999,74	2307,35
301	5020	374	161,58	366,71	840,25

Tabela 4.1: Tabela comparativa de perfomance.

Nota-se com esse teste que o tempo de processamento no sistema embarcado é aproximadamente 2,3 vezes maior do que no computador utilizando o Matlab e 5,5 vezes maior do que no computador utilizando os mesmos algoritmos em C++ com a biblioteca OpenCV. Percebe-se também que para esses casos o tempo de processamento em C++ no computador de todos os algoritmos foi sempre menor que o tempo de vídeo para uma gravação com 16fps, já para o sistema embarcado esse tempo foi sempre maior. Portanto, para o sistema funcionar de forma ideal no sistema embarcado esse código precisa ser otimizado para o tempo de processamento ser menor que o tempo de vídeo e dessa forma poder funcionar em paralelo com a aquisição do vídeo.

Após esse teste, foi realizada a gravação de 30 vídeos de aproximadamente 5 a 6 minutos de duração e taxa de 16fps com frequência de movimentos distintas para verificar a influência dessa frequência no tempo de processamento de vídeo. Esses testes foram todos processados no PC utilizando os algoritmos gerados em C++ por ser mais eficaz. Os resultados coletados podem ser observados na Tabela 4.2 a seguir.

Frames totais	Frames em Movimento	Tempo (s)
6085	89	132,33
5017	207	128,72
5054	156	127,47
6513	261	160,89
4983	256	136,84
5406	126	120,88
6536	194	156,43
5534	276	145,91
5109	118	117,09
4775	325	144,23
4994	761	210,01
4921	593	183,60
5397	675	222,71
4877	585	189,15
5020	374	162,91
4849	378	158,57
5382	425	164,27
4843	421	161,91
5671	466	178,33
4960	280	138,92
5232	747	230,01
5098	882	233,21
5014	935	260,03
5043	978	259,92
5469	967	260,51
5348	962	252,88
5361	1109	287,32
5378	1042	277,54
5209	1075	278,99
5265	1107	296,89

Tabela 4.2: Tabela com dados de teste da influência do movimento no tempo de processamento

Com os dados coletados, buscou-se encontrar através de um processo de regressão linear a relação entre as variáveis. Ao aplicar a regressão linear obteve-se um modelo com coeficiente de determinação (R^2) de 0,9839, pode-se concluir, portanto, que para faixa de teste o modelo linear representa muito bem os dados e que o tempo de processamento varia linearmente com a quantidade de *pixels* total e com a quantidade de *pixels* em movimento. Dessa forma, vídeos com pouca presença de

movimento são processados de forma muito mais rápida. Na Figura 4.4 segue o gráfico dos pontos observados em relação aos pontos calculados pelo modelo proposto.

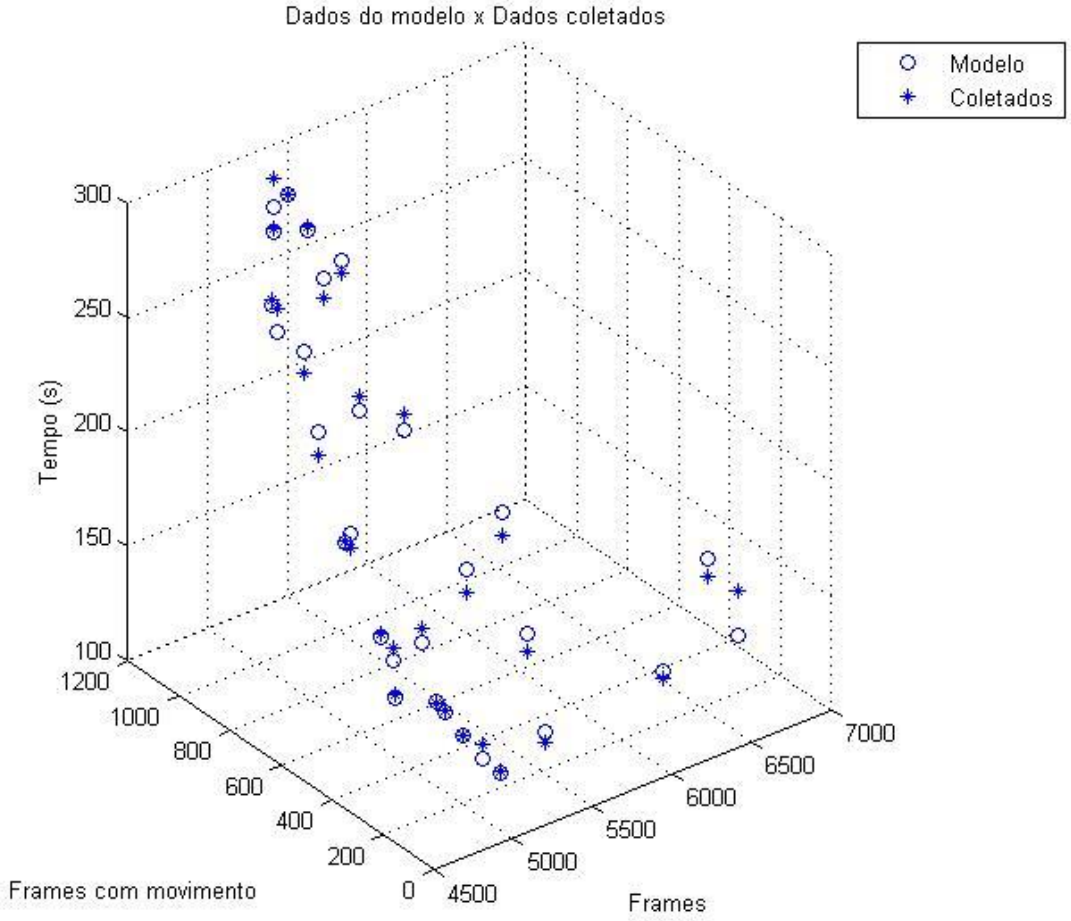

Figura 4.4: Gráfico de validação de modelo de linearidade

Foram detectados alguns problemas nas *tags* quando pequenas partes do corpo se moviam e o resto permanecia parado. Esse acontecimento fazia com que fossem detectadas regiões de movimentos separadas para um mesmo corpo em movimento. O fenômeno resultante pode ser observado na Figura 4.5 abaixo.

Figura 4.5: Exemplo de problema de múltiplas *tags* para um mesmo corpo em movimento

Neste caso, foram detectadas três regiões de movimento distintas. Esse problema necessitaria de soluções mais inteligentes para realizar o tracking do movimento, com reconhecimento total do objeto em movimento.

Outro problema detectado devido ao algoritmo utilizado para detecção de movimento foi que caso o objeto em movimento ficasse parado por alguns segundos e só depois voltasse a se mover o sistema iria detectar dois movimentos distintos.

Esse sistema foi por fim comparado com outro produto semelhante, o FLIR FX. Esse produto tem o custo de $199 e possui uma tecnologia denominada Rapid Recap que compõe vários movimentos realizados durante um dia num único vídeo [43]. Em termos de custos, o projeto realizado possui um valor mais baixo, pois o sistema proposto é composto pela Raspberry que possui o preço de $35 e uma câmera comum com preço próximo a $15. Outra diferença entre as propostas está no local de processamento, enquanto o sistema proposto processa todo o algoritmo no sistema embarcado, o produto FLIR FX é responsável pela captação das filmagens e transmissão para a nuvem [44], enquanto o restante do processamento é realizado nela. Dessa forma, em termos de desempenho o projeto realizado demora mais pra gerar o resultado final. Por fim, o resultado final dos dois produtos, apesar de terem uma proposta semelhante, é bem distinto. O projeto executado utiliza de sobreposição por transparência e dessa forma todos os movimentos detectados ficam menos opacos no resultado final, enquanto o fundo permanece praticamente inalterado, dessa forma existe uma limitação na quantidade de sobreposições que podem ser feitas para não prejudicar a visibilidade do produto final. Já o produto analisado, utiliza de outras técnicas para sobrepor as cenas de movimento. A Figura 4.6 mostra um *frame* do vídeo gerado pelo FLIR FX. Nessa imagem é possível observar que não foi utilizado a transparência para realizar a

sobreposição dos movimentos. Na Figura 4.7, no entanto, é possível observar algumas falhas na técnica utilizada. Nos quadrados vermelhos é possível observar que faltam partes do corpo do indivíduo que está se movimentando.

Figura 4.6: Exemplo de *frame* de vídeo gerado pela tecnologia RapidRecap

Figura 4.7: Exemplo de *frame* de vídeo gerado pela tecnologia RapidRecap com falhas

Porém, apesar de algumas falhas, o sistema projetado atende ao que foi proposto e é capaz de reduzir significativamente o espaço armazenado de filmagens, essa redução é de no mínimo 5 vezes, assumindo que a maioria das filmagens tenham a presença de movimento e pode chegar a valores bem mais elevados para filmagens com baixa presença de movimento, além disso o sistema gera vídeos finais mais curtos que facilitam a busca de um determinado evento. Existem melhorias a serem feitas que serão discutidas no próximo capítulo.

5. Conclusões

Esse projeto foi baseado no conceito de Internet das Coisas e tem como objetivo desenvolver um sistema de segurança que facilite a vida das pessoas, além de as deixarem mais tranquilas em suas residências ou com seus estabelecimentos. O grande diferencial do projeto está no seu baixo custo tanto de produção, pois necessita de poucos materiais, quanto de manutenção. Além disso, o sistema de armazenamento dos vídeos necessitará de um espaço muito menor comparado a outros sistemas de segurança, o que torna o projeto extremamente interessante.

O projeto deu oportunidade para estudo na área de visão computacional, um estudo mais aprofundado de uma linguagem de programação orientada a objeto (C++), otimização de algoritmos, inteligência artificial e diversas outras áreas relacionadas a sistemas embarcados.

Os resultados encontrados até aqui foram muito animadores para a continuidade da evolução da ideia. O algoritmo utilizou apenas 33% da CPU do sistema embarcado, o que nos permite explorar o aumento do uso da CPU para aumentar a performance do sistema.

Comparando com outro sistema já existente no mercado, é possível notar vantagens e desvantagens entre os produtos. A principal vantagem está no preço de custo, enquanto as desvantagens estão na falta de uma interface com o usuário, e no tempo de processamento. O resultado final de ambos possui falhas e limitações distintas.

Para a transformação do projeto em um produto ainda são necessárias algumas melhorias no sistema tanto em tempo de performance, quanto em interface com o usuário que não foi o foco desse projeto.

5.1. Implementações futuras

As principais implementações futuras para o projeto são otimizações gerais no algoritmo para reduzir tempo de processamento, estudar a possibilidade do uso da GPU para processamento através da biblioteca OpenCL e/ou estudar a possibilidade do uso de *threads* para realizar tarefas em paralelo, aumentando o uso da CPU e acelerando o processamento dos algoritmos. Além disso, a utilização de algoritmos mais inteligentes para reduzir algumas falhas de detecção e no tracking de movimento.

Também é necessário passar a realizar a captação das filmagens através do sistema embarcado, isso foi descartado até o momento do projeto pois o tempo para processamento de 1 segundo do arquivo de vídeo é maior que 1 segundo e dessa forma é inviável o processamento em tempo real.

Outra implementação possível é o estudo do uso de processamento em nuvem. Dessa forma o sistema embarcado ficaria responsável pela captação das filmagens, os processamentos mais simples e a transmissão dos dados para a nuvem, enquanto essa processaria a parte mais pesadas dos algoritmos.

Ou ainda, estudar a possibilidade da utilização de hardware para realizar a detecção de movimento, reduzindo grande parte do software para esse fim, essa implementação reduziria aproximadamente 1/3 o tempo de execução do algoritmo, em relação ao algoritmo atual.

Por fim, é necessária a criação de uma interface com o usuário, podendo ser via aplicação no computador, via Web ou aplicativo. Caso seja uma aplicação que utilize a internet também é necessário o estudo de *cybersecurity*. Uma norma amplamente utilizada para esse fim que deve ser modelo para a criação dessa interface é a NIST [45].

6. Referências Bibliográficas

[1] IBGE, Censo Demográfico 1980 e 2010 <http://7a12.ibge.gov.br/vamosconhecer-o-brasil/nosso-povo/caracteríticas-da-populacao.html> Acesso em: 17 de junho de 2017.

[2] Pesquisa Nacional por Amostra de Domicílios Contínua (Pnad Contínua) <http://www.ibge.gov.br/home/estatistica/indicadores/trabalhoerendimento/pnad_contin ua/> Acesso em: 17 de junho de 2017. /

[3] Santos, M.J.; KASSOUF, A.L. Estudos Econômicos das Causas da Criminalidade no Brasil: Evidências e Controvérsias. Revista Econômica. Maio/Agosto, 2008.

[4] Índices de assaltos e furtos no Brasil < http://www.estadao.com.br/noticias/geral,ibge-11-9-milhoes-foram-vitimas-deroubos-em-1-ano,653922> Acesso em: 17 de junho de 2017.

[5] OLIVEIRA, A.F. Empreses de vigilância no sistema de prestação de serviços de segurança patrimonial privada: uma avaliação da estrutura de governança. Tese de Doutorado. Escola Superior de Agricultura "Luiz de Queiroz" Universidade de São Paulo. Piracicaba, 2004.

[6] QUEIROZ, J.E.R.; GOMES, H.M. Introdução ao Processamento Digital de Imagens. Revista RITA: instruções para preparação de documentos em Word. Vl.VIII. nº1, 2001.

[7] A. Erhardt-Ferron.Theory and Applications of Digital Image Processing. Universityof Applied Sciences Offenburg, 1 edition, 2000.

[8] ALBUQUERQUE, M. P.; ALBUQUERQUE, M.P. Processamento de Imagens: Métodos e Análises. Disponível em: < http://www.cbpf.br/~mpa/curso2.htm> Acesso em: 10 de junho de 2017.

[9] HIGASHIMO, W. A. Estudo Comparativo de Algoritmos de Subtração de Fundo em Sequência de Imagens. Dissertação (Mestrado) Universidade Estadual de Campinas, Campinas, São Paulo, Mar 2006.

[10] FIGUEROA, P. J.; LEITE, N. J.; BARROS, R. M. L. Background recovering in outdoor image sequences: An example of soccer players segmentation. Image and Vision Computing, 2006.

[11] GONZALEZ, R. E.; WOODS, R. E. Digital Image Processing. 3rd. ed. New York: Prentice Hall, 2008.

[12] BARBOSA, R. L.; GALLIS, R. B. A.; SILVA, J. F. C. da; JÚNIOR, M. M. A computação do fluxo óptico em imagens obtidas por um sistema móvel de mapeamento terrestre. Revista Brasileira de Cartografia, v. 57, n. 2, p. 72 78, 2005.

[13] BARRON, J. L.; FLEET, D. J.; BEAUCHEMIN, S. S. Performance of optical ow techiniques. International Journal of Computer Vision, v. 12, n. 1, p. 43 77, 1994.

[14] COSTA, L. H. L. Desenvolvimento de um sistema semi-automático para determinação de durezas Brinell e Vickers utilizando técnicas de processamento e análise de imagem. Dissertação (Mestrado) Centro Federal de Educação Tecnológica do Ceará, 2008.

[15] Wallace, Bruce A. "Merging and transformation of raster images for cartoon animation". SIGGRAPH Computer Graphics. New York City, New York: ACM Press. 15 (3): 253–262, 1981.
[16] Inteligência Artificial < http://brasilescola.uol.com.br/informatica/inteligenciaartificial.htm> Acesso em: 10 de maio de 2017.
[17] GOMES, D.S. Inteligência Artificial: Conceitos e Aplicações. Revista Olhar Científico - Faculdades Associada de Ariquemes. Vl.01, nº 2, Ago/Dez, 2010.
[18] Redes Neurais. < http://www.din.uem.br/ia/neurais/#neural> Acesso em: 10 de maio de 2017.
[19] Qual a diferenca entre Machine Learning e Deep Learning|ComputerWorld. < https://computerworld.com.br/brandpost/qual-diferenca-entre-machine-learning-e-deep-learning/> Acesso em: 1 de maio de 2019.
[20] SEL0362 – Inteligência Artificial – LASI < http://www.sel.eesc.usp.br/lasi/lasi/?page_id=672&lang=pt> Acesso em: 22 de junho de 2019.
[21] Recapitulando | PETNews - Busca em Profundidade e em Largura. < http://www.dsc.ufcg.edu.br/~pet/jornal/junho2012/materias/recapitulando.html> Acesso em : 1 de maio de 2019.
[22] Sistemas Embarcados – Definição e importância – Embarcados. < https://www.embarcados.com.br/sistema-embarcado/> Acesso em: 22 de abril de 2019.
[23] Computação em Nuvem e Sistemas Embarcados (Tecnology Leadership Council - Brazil). < https://www.ibm.com/developerworks/community/blogs/tlcbr/entry/computacao_em_nuvem_e_sistemas_embarcados?lang=en> Acesso em: 24 de abril de 2019.
[24] Breve Histórico da ARM e evolução da arquitetura - Embarcados < https://www.embarcados.com.br/breve-historico-da-arm/> Acesso: em 26 de abril de 2019.
[25] Introduction to ARM thumb | Embedded < https://www.embedded.com/electronics-blogs/beginner-s-corner/4024632/Introduction-to-ARM-thumb> Acesso em: 26 de abril de 2019.
[26] Arquiteturas RISC < http://www.zottesso.com.br/professor/disciplinas/Arquitetura/03%20-%20RISC,%20CISC.pdf> Acesso em: 26 de abril de 2019.
[27] Linux - The Linux Foundation < https://www.linuxfoundation.org/projects/linux/> Acesso em: 1 de maio de 2019.
[28] What is Linux | Linux.com | The source for Linux information < https://www.linux.com/what-is-linux> Acesso em: 1 de maio de 2019.
[29] Linux e GNU - Projeto GNU - Free Software Foundation < https://www.gnu.org/gnu/linux-and-gnu.pt-br.html> Acesso em: 1 de maio de 2019.
[30] GNU Toolchain Explained | Linux.com < https://www.linux.org/threads/gnu-toolchain-explained.10570/> Acesso em: 1 de maio de 2019.
[31] Distribuições Linux: características, diferenças e objetivos < https://becode.com.br/distribuicoes-linux-caracteristicas-e-objetivos/> Acesso em: 1 de maio de 2019.

[32] Raspberry Pi Foundation - About Us < https://www.raspberrypi.org/about/> Acesso em: 2 de maio de 2019.
[33] O que é o MATLAB? < http://w3.impa.br/~zubelli/tutorial/node1.html> Acesso em: 10 de maio de 2017.
[34] OpenCV. <http://opencv.org/about.html> Acesso em: 9 de maio de 2017.
[35] OpenCV: OpenCV modules. < https://docs.opencv.org/3.2.0/> Acesso em: 20 de abril de 2019.
[36] GPU E Deep Learning - Ciência e Dados < http://www.cienciaedados.com/gpu-e-deep-learning/> Acesso em: 28 de abril de 2019.
[37] FrontPage - Raspbian < https://www.raspbian.org> Acesso em: 1 de maio de 2019.
[38] Download NOOBS for Raspberry Pi <https://www.raspberrypi.org/downloads/noobs/> Acesso em: 1 de março de 2019.
[39] Baixar o VNC Server para a Raspberry Pi | VNC Connect <https://www.realvnc.com/pt/connect/download/vnc/raspberrypi/> Acesso em: 5 de março de 2019.
[40] Baixar o VNC Viewer | VNC Connect <https://www.realvnc.com/pt/connect/download/viewer/> Acesso em: 5 de março de 2019.
[41] Downloading WinSCP-5.15.1-Setup.exe :: WinSCP <https://winscp.net/download/WinSCP-5.15.1-Setup.exe> Acess o em: 10 de março de 2019.
[42] OpenCV – Browse /opencv-win/3.2.0 at SourceForge.net <https://sourceforge.net/projects/opencvlibrary/files/opencv-win/3.2.0> Acesso em: 3 de março de 2019.
[43] New FLIR Security Camera Shows You an Entire Day's Activity at Once <https://www.popularmechanics.com/technology/gadgets/a14951/flir-fx-rapid-recap/> Acesso em: 24 de maio de 2019.
[44] FLIR® FX With RapidRecap <https://docs-emea.rs-online.com/webdocs/1498/0900766b81498ee0.pdf> Acesso em: 24 de maio de 2019.
[45] Standarts and Measurements | NIST <https://www.nist.gov/services-resources/standards-and-measurements> Acesso em: 25 de junho de 2019.

Apêndice A

Códigos em Matlab

```
clear all
close all
clc

DetectaMov();
AdicionaTags();
CriaSequencias();
Sobrepoe();

function DetectaMov()
   v = VideoReader('C:\Users\Eduardo Orru\Documents\Pastas
Rodrigo\TCC\Videos\WIN_20170429_13_45_23_Pro.mp4');
   horainicial = [13 45 23];
   nFrames = v.NumberOfFrames;
   titulo = 'C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\movimentos\mova.avi';
   flag = 0;
   atual = read(v,1);
   j = -30;
   for i = 2:nFrames
      if mod(i,16) < 1
         if horainicial(3) < 59
         horainicial(3) = horainicial(3)+1;
         else
            if horainicial(2) < 59
               horainicial(2) = horainicial(2) + 1;
               horainicial(3) = 0;
            else
               horainicial(1) = horainicial(1) + 1;
               horaincial(2) = 0;
            end
         end
      end
      seguinte = read(v,i);
      atual = imfilter(atual,fspecial('average', [3 3]));
      seguinte = imfilter(seguinte,fspecial('average', [3 3]));
```

```
    diff = imabsdiff(seguinte,atual);
    diff = im2bw(diff,0.2);
    diff = medfilt2(diff);
    x = sum(sum(diff));
    if x > 20
        if flag == 0
            horavideo = horainicial;
            mov = VideoWriter(titulo);
            mov.FrameRate = 16;
            open(mov)
        end
        writeVideo(mov,atual);
        j = i;
        flag = 1;
    else
        if i - j < 30
            writeVideo(mov,atual);
        else
            if flag == 1
                titulotxt = [titulo(1:end-3) 'txt'];
                fileopen = fopen(titulotxt,'w');
                fprintf(fileopen, '%3d %3d %3d',horavideo);
                fclose(fileopen);
                close(mov)
                r = VideoReader(titulo);
                nFramesmov = r.NumberOfFrames;
                if nFramesmov > 45
                    titulo(66) = titulo(66)+1;
                end
            end
            flag = 0;
        end
    end
    atual = seguinte;
end
if flag == 1
    titulotxt = [titulo(1:end-3) 'txt'];
    fileopen = fopen(titulotxt,'w');
    fprintf(fileopen, '%3d %3d %3d',horavideo);
    fclose(fileopen);
```

```
        close(mov)
    end
    clear('r')
    clear('v')
end

function AdicionaTags()
    tag = 'C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\tags\taga';
    titulo = 'C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\movimentos\mova.avi';
    D = dir('C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\movimentos\');
    const = length(D(not([D.isdir])))/2;
    for p = 1:const
        v = VideoReader(titulo);
        titulotxt = [titulo(1:end-3) 'txt'];
        arq = fopen(titulotxt);
        horainicial = fscanf(arq,'%f')';
        nFrames = v.NumberOfFrames;
        tagtxt = [tag '.txt'];
        fileopen = fopen(tagtxt,'w');
        limx = [];
        limy = [];
        lasticm = [0];
        lastjcm = [0];
        for k = 1:nFrames-30
            if mod(k,16) < 1
              if horainicial(3) < 59
               horainicial(3) = horainicial(3)+1;
               else
                  if horainicial(2) < 59
                     horainicial(2) = horainicial(2) + 1;
                     horainicial(3) = 0;
                  else
                     horainicial(1) = horainicial(1) + 1;
                     horaincial(2) = 0;
                  end
               end
            end
            icm = 0;
            jcm = 0;
            regiaox = [];
```

```
regiaoy = [];
contador = 1;
coluna = 0;
atual = read(v,k);
seguinte = read(v,k+3);
diff = imabsdiff(seguinte,atual);
diff = imfilter(diff,fspecial('average', [3 3]));
diff = im2bw(diff,0.1);
diff = medfilt2(diff);
diff = edge(diff,'sobel');
diffred = imresize(diff,0.5,'nearest');
diffampliado = zeros(350,560);
diffampliado(41:310,41:520) = diffred;
[x y] = find(diffampliado == 1);
while(x)
   [a b] = find(diffampliado(x(1)-40:x(1)+40,y(1)-40:y(1)+40) == 1);
   if size(a,1) > 1
      regiaox = [regiaox x(1)];
      regiaoy = [regiaoy y(1)];
   else
      if y(1) >= coluna
         if((max(regiaox) - min(regiaox))*(max(regiaoy) - min(regiaoy)) > 200)
            limx(contador,1,k) = 2*(min(regiaox)-40);
            limx(contador,2,k) = 2*(max(regiaox)-40);
            limy(contador,1,k) = 2*(min(regiaoy)-40);
            limy(contador,2,k) = 2*(max(regiaoy)-40);
            icm = round((limx(contador,1,k) + limx(contador,2,k))/2);
            jcm = round((limy(contador,1,k) + limy(contador,2,k))/2);
            if (icm && jcm)
               for i=1:size(lasticm,2)
                  if(abs(icm - lasticm(1,i)) < 40 && abs(jcm - lastjcm(1,i)) < 40)
                     icm = lasticm(1,i);
                     jcm = lastjcm(1,i);
                  end
               end
               if jcm < 960-80
                  if icm < 540-80
                     inicio = [icm, jcm, horainicial(1), horainicial(2)];
                     fprintf(fileopen,' %5d %5d %5d %5d', inicio);
                  else
```

```
                    inicio = [icm-19, jcm, horainicial(1), horainicial(2)];
                    fprintf(fileopen,' %5d %5d %5d %5d', inicio);
                end
            else
                if icm < 540-80
                    inicio = [icm, jcm-79, horainicial(1), horainicial(2)];
                    fprintf(fileopen,' %5d %5d %5d %5d', inicio);
                else
                    inicio = [icm-19, jcm-79, horainicial(1), horainicial(2)];
                    fprintf(fileopen,' %5d %5d %5d %5d', inicio);
                end
            end
        end
        lasticm(contador) = icm;
        lastjcm(contador) = jcm;
        contador = contador +1;
    end
    regiaox = [];
    regiaoy = [];
  end
end
diffampliado(x(1),y(1)) = 0;
if(y)
  if coluna < max(b)+y(1)-41

    coluna = max(b)+y(1)-41;
  end
end
x(1) = [];
y(1) = [];
end
fprintf(fileopen,'\n');
end
fclose(fileopen);
tag(60) = tag(60) + 1;
titulo(66) = titulo(66) + 1;
end
end

function CriaSequencias()
```

```
D = dir('C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\movimentos\');
const = length(D(not([D.isdir])))/2;
nmax = 5;
titulo = 'C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\movimentos\mova.avi';
tagtxt = 'C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\tags\taga.txt';
for i = 1:const
    v(i) = VideoReader(titulo);
    nFrames(i) = v(i).NumberOfFrames-30;
    A = importdata(tagtxt,'\n');
    for j = 1:size(A,1)
        if str2num(A{j})
            tag(j,1:size(str2num(A{j})),2),i) = str2num(A{j});
        end
    end
    titulo(66) = titulo(66) + 1;
    tagtxt(60) = tagtxt(60)+1;
end

for i = 1:size(nFrames,2)-1
    maximo = i;
    for j= i:size(nFrames,2)
        if nFrames(j) > nFrames(maximo)
            maximo = j;
        end
    end
    if i ~= maximo
        aux = nFrames(i);
        aux2 = v(i);
        aux3 = tag(:,:,i);
        nFrames(i) = nFrames(maximo);
        v(i) = v(maximo);
        tag(:,:,i) = tag(:,:,maximo);
        nFrames(maximo) = aux;
        v(maximo) = aux2;
        tag(:,:,maximo) = aux3;
    end
end
[sequencia valormax] = algoritmo(nFrames,const,nmax);
titulo = 'C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\sequencias\seq1.avi';
for j = 1:size(sequencia,1)
```

```
mov = VideoWriter(titulo);
mov.FrameRate = 16;
open(mov);
seqtxt = [titulo(1:end-3) 'txt'];
fileopen = fopen(seqtxt,'w');
seqtag = [];
min = [];
hora = [];
icm = [];
jcm = [];
for k = 1:size(sequencia,2)
    if sum(sequencia(j,k))
        for i = 1:nFrames(sum(sequencia(j,k))-96)
            writeVideo(mov,read(v(sum(sequencia(j,k))-96),i));
        end
        seqtag = [seqtag; tag(1:nFrames(sum(sequencia(j,k))-96),:,sum(sequencia(j,k))-96)];
    end
end
for i = 1:size(seqtag,2)
    if mod(i,4) == 0
        min = [min i];
    else
        if mod(i,4) == 1
            icm = [icm i];
        else
            if mod(i,4) == 2
                jcm = [jcm i];
            else
                hora = [hora i];
            end
        end
    end
end
vet_icm = seqtag(:,icm);
vet_jcm = seqtag(:,jcm);
vet_hora = seqtag(:,hora);
vet_min = seqtag(:,min);
for i = 1:size(vet_icm,1)
    if find(vet_icm(i,:))
        vet_icm_no = vet_icm(i,find(vet_icm(i,:)));
```

```
            vet_jcm_no = vet_jcm(i,find(vet_icm(i,:)));
            vet_hora_no = vet_hora(i,find(vet_icm(i,:)));
            vet_min_no = vet_min(i,find(vet_icm(i,:)));
            for k = 1:size(vet_icm_no,2)
                fprintf(fileopen, '%5d %5d %5d
%5d',vet_icm_no(k),vet_jcm_no(k),vet_hora_no(k),vet_min_no(k));
            end
        end
        fprintf(fileopen,'\n');
    end
    fclose(fileopen);
    close(mov);
    seqtxt(66) = seqtxt(66) +1;
    titulo(66) = titulo(66)+1;
  end
end

function [sequencia valormaximo] = algoritmo(vetor,const,nmax)
  coordenada = {};
  for i = 1:const
    coordenada = [coordenada vetor(i)];
  end
  s = {'a'};
  for i = 1:const-1
    s = [s char(s{end}(1)+1)];
  end
  stringtotal = [];
  for i = 1:const
    stringtotal = [stringtotal s{i}];
  end
  con = {};
  for i = 1:const
    aux = stringtotal;
    aux(1:i)=[];
    con{i} = aux;
  end
  cidade = struct('nome', s, 'coordenada', coordenada,'conexao', con);
  x = 0;
  vetor = [];
  for i = 1:size(cidade,2)
```

```
    vetor = [vetor cidade(i).coordenada];
end
soma = sum(vetor);
n = round((sum(vetor)/(nmax-1) + 3*sum(vetor)/nmax)/4);
erroaceitavel = (n*nmax - soma)/(nmax+1);
disp(n);
disp(erroaceitavel);
while(size(cidade,2))
    ex = [];
    ey = [];
    x = x+1;
    i = 1;
    k = 2;
    j = size(cidade,2);
    [a valormaximo(x) arvore] = criararvore(cidade,i,n,erroaceitavel);
    sequencia(x,1) = cidade(1).nome;
    cidade(1) = [];
    for i = 1:size(cidade,2)
        cidade(i).conexao = strrep(cidade(i).conexao,arvore(1).nome,'');
    end
    while(arvore(a).pai ~=0)
        sequencia(x,k) = arvore(a).nome;
        cidade(buscacidade(cidade,arvore(a).nome)) = [];
        for i = 1:size(cidade,2)
            cidade(i).conexao = strrep(cidade(i).conexao,arvore(a).nome,'');
        end
        a = arvore(a).pai;
        k = k+1;
    end
    pai = [];
    nome = [];
    for i = 1:size(arvore)
        nome = [nome;' ' arvore(i).nome ' '];
        pai = [pai,arvore(i).pai];
    end
    nivel=size(arvore,1);
    arvorebusca = struct('nome', nome, 'pai', pai, 'nivel', nivel);
    for j = 1:size(arvore,1)
        for i = 1:size(arvore,1)
            if(arvorebusca.pai(i) == j)
```

```
                arvorebusca.nivel(i) = arvorebusca.nivel(j)-1;
            end
        end
    end
    arvorebusca.nivel = (arvorebusca.nivel - min(arvorebusca.nivel) + 1);
    arvorebusca.nivel = arvorebusca.nivel/(max(arvorebusca.nivel)+1);
    figure('Color',[1 1 1]);
    plotararvore(pai,",",arvorebusca.nivel)
    c = get(gcf, 'Children');
    c = get(c,'Children');
    if size(c,1) == 2
        ex = get(c(2),'XData');
        ey = get(c(2), 'YData');
    else
        auxx = get(c,'XData');
        auxy = get(c,'YData');
        ex(1) = 0.5;
        ey(1) = arvorebusca.nivel(1);
        for i = 1:size(auxx,2)
            if mod(i-1,3) == 0
                ex = [ex auxx(i)];
                ey = [ey auxy(i)];
            end
        end
    end
    text(ex,ey,nome ,...
        'BackgroundColor',[0 0 0.4],...
        'HorizontalAlignment','center',...
        'Color', [1 1 1]);
    title('\bf\fontsize{16}Árvore de Busca')
    axis('off');
end

function [a valormax arvore] = criararvore(cidade,i,n,erroaceitavel)
    nome = cidade(i).nome;
    noatual = 1;
    pai = 0;
    arvore = {};
    cond = 1;
    k=0;
```

```
valores = [];
arvore= struct('nome', nome, 'no', noatual, 'pai', pai, 'soma', cidade(i).coordenada);
while(1)
   k = k + 1;
   if(size(arvore,1)>= k)
      for j = 1:size(cidade(buscacidade(cidade,arvore(k).nome)).conexao,2)
         x = k;
         while(arvore(x).pai ~=0)
            if(arvore(arvore(x).pai).nome == cidade(buscacidade(cidade,arvore(k).nome)).conexao(j));
               cond = 0;
               break
            else cond = 1;
            end
            x = arvore(x).pai;
         end
         if cond == 1
            noatual = noatual + 1;
            pai = k;
            nome = cidade(buscacidade(cidade,arvore(k).nome)).conexao(j);
            valor = cidade(buscacidade(cidade,nome)).coordenada + arvore(pai).soma;
            if(valor < n)
               noatual = noatual + 1;
               arvore = [arvore;struct('nome', nome, 'no', noatual, 'pai', pai, 'soma', valor)];
            end
         end
         if(n - arvore(end).soma < erroaceitavel)
            valormax = arvore(end).soma;
            a = size(arvore,1);
            disp(arvore)
            return
         end
      end
   else break
   end
end
for i = 1:size(arvore,1)
      valores = [valores arvore(i).soma];
end
valormax = max(valores);
a = find(valores == valormax);
```

```
    a = min(a);

    function i = buscacidade(struct,cidade)
    for j=1:size(struct,2)
        if(cidade == struct(j).nome)
            break;
        else j = -1;
        end
    end
    i=j;

function plotararvore(p,c,d,nivel)

[x,y,h]=treelayout(p);

if nargin == 4
    y = nivel;
end

f = find(p~=0);
pp = p(f);


X = [x(f); x(pp); NaN(size(f))];
Y = [y(f); y(pp); NaN(size(f))];

X = X(:);
Y = Y(:);

n = length(p);

if nargin == 1,
    if n < 500,
        plot (x, y, 'ro', X, Y, 'r-');
    else
        plot (X, Y, 'r-');
    end;
else
    [~, clen] = size(c);
    [~, dlen] = size(d);
```

```
    if n < 500
        if clen>0 && dlen>0
            plot (x, y, c, X, Y, d);
        elseif clen>0,
            plot (x, y, c, X, Y , 'r-');
        elseif dlen>0,
            plot (x, y, 'ro', X, Y, d);
        else
            plot(x, y, 'ro', X, Y, 'r-');
        end;
    elseif dlen>0
        plot (X, Y, d);
    else
        plot (X, Y, 'r-');
    end;
end;
xlabel(['height = ' int2str(h)]);
axis([0 1 0 1]);

function Sobrepoe()
    D = dir('C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\sequencias\');
    const = length(D(not([D.isdir])))/2;
    titulo = 'C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\sequencias\seq1.avi';
    mov = VideoWriter('C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\VideosFinais\videofinal.avi');
    mov.FrameRate = 16;
    open(mov);
    for i = 1:const
        v(i) = VideoReader(titulo);
        nFrames(i) = v(i).NumberOfFrames;
        titulo(66) = titulo(66)+1;
    end
    for i = 1:size(nFrames,2)-1
        maximo = i;
        for j= i:size(nFrames,2)
            if nFrames(j) > nFrames(maximo)
                maximo = j;
            end
        end
        if i ~= maximo
            aux = nFrames(i);
```

```
            aux2 = v(i);
            nFrames(i) = nFrames(maximo);
            v(i) = v(maximo);
            nFrames(maximo) = aux;
            v(maximo) = aux2;
        end
    end
    for i = 1:nFrames(1)
        frame1{i,1} = read(v(1),i);
    end
    for i = 1:nFrames(2)
        for j = 2:const
            if nFrames(j) >= i
                weight1 = (j-1)/j;
                weight2 = 1/j;
                for k = 1:3
                    frameatual = read(v(j),i);
                    frame1{i,1}(:,:,k) = uint8( weight1 .* single(frame1{i,1}(:,:,k)) + weight2 .*
single(frameatual(:,:,k)));
                end
            end
        end
    end
    seqtxt = 'C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\sequencias\seq1.txt';
    icm = [];
    jcm = [];
    hora = [];
    min = [];
    for i = 1:const
        A = importdata(seqtxt,'\n');
        for j = 1:size(A,1)
            if str2num(A{j})
                seqtag(j,1:size(str2num(A{j}),2),i) = str2num(A{j});
            end
        end
        seqtxt(66) = seqtxt(66) + 1;
    end
    seqtagfinal = [];
    for i = 1:const
        seqtagfinal = [seqtagfinal seqtag(:,:,i)];
```

```
end
for i = 1:size(seqtagfinal,2)
    if mod(i,4) == 0
        min = [min i];
    else
        if mod(i,4) == 1
            icm = [icm i];
        else
            if mod(i,4) == 2
                jcm = [jcm i];
            else
                hora = [hora i];
            end
        end
    end
end
vet_icm = seqtagfinal(:,icm);
vet_jcm = seqtagfinal(:,jcm);
vet_hora = seqtagfinal(:,hora);
vet_min = seqtagfinal(:,min);
tagtxt ='C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\VideosFinais\tagfinal.txt';
fileopen = fopen(tagtxt,'w');
for i = 1:nFrames(1)
    for j = 1:size(vet_icm,2)
        if(vet_icm(i,j))
            hora = conversor(num2str(vet_hora(i,j)),num2str(vet_min(i,j)));
            frame1{i}(vet_icm(i,j):vet_icm(i,j)+19,vet_jcm(i,j):vet_jcm(i,j)+79,1) = hora;
            frame1{i}(vet_icm(i,j):vet_icm(i,j)+19,vet_jcm(i,j):vet_jcm(i,j)+79,2) = hora;
            frame1{i}(vet_icm(i,j):vet_icm(i,j)+19,vet_jcm(i,j):vet_jcm(i,j)+79,3) = hora;
        end
    end
    if find(vet_icm(i,:))
        vet_icm_no = vet_icm(i,find(vet_icm(i,:)));
        vet_jcm_no = vet_jcm(i,find(vet_icm(i,:)));
        vet_hora_no = vet_hora(i,find(vet_icm(i,:)));
        vet_min_no = vet_min(i,find(vet_icm(i,:)));
        for k = 1:size(vet_icm_no,2)
            fprintf(fileopen, '%5d %5d %5d %5d',vet_icm_no(k),vet_jcm_no(k),vet_hora_no(k),vet_min_no(k));
        end
    else
```

```
            fprintf(fileopen,'\n');
        end
        fprintf(fileopen,'\n');
        writeVideo(mov,frame1{i});
    end
    fclose(fileopen);
    close(mov);
end

function hora = conversor(strhora,strmin)
    hora = [];
    if size(strhora,2) == 1
        strhora = ['0' strhora];
    end
    if size(strmin,2) == 1
        strmin = ['0' strmin];
    end
    doisponto = imread('C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\algarismos\doisponto.bmp');
    algarismo0 = imread('C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\algarismos\algarismo0.bmp');
    algarismo1 = imread('C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\algarismos\algarismo1.bmp');
    algarismo2 = imread('C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\algarismos\algarismo2.bmp');
    algarismo5 = imread('C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\algarismos\algarismo5.bmp');
    algarismo6 = imread('C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\algarismos\algarismo6.bmp');
    algarismo7 = imread('C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\algarismos\algarismo7.bmp');
    algarismo3 = imread('C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\algarismos\algarismo3.bmp');
    algarismo4 = imread('C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\algarismos\algarismo4.bmp');
    algarismo8 = imread('C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\algarismos\algarismo8.bmp');
    algarismo9 = imread('C:\Users\Eduardo Orru\Documents\Pastas Rodrigo\TCC\algarismos\algarismo9.bmp');
    [x y] = find(doisponto == 0);
    xmin = min(x);
    xmax = max(x);
    ymin = min(y);
    ymax = max(y);
    doisponto = doisponto(xmin:xmax,ymin:ymax);
    algarismo0 = algarismo0(xmin:xmax,ymin:ymax);
    algarismo1 = algarismo1(xmin:xmax,ymin:ymax);
    algarismo2 = algarismo2(xmin:xmax,ymin:ymax);
    algarismo3 = algarismo3(xmin:xmax,ymin:ymax);
    algarismo4 = algarismo4(xmin:xmax,ymin:ymax);
    algarismo5 = algarismo5(xmin:xmax,ymin:ymax);
```

```
algarismo6 = algarismo6(xmin:xmax,ymin:ymax);
algarismo7 = algarismo7(xmin:xmax,ymin:ymax);
algarismo8 = algarismo8(xmin:xmax,ymin:ymax);
algarismo9 = algarismo9(xmin:xmax,ymin:ymax);
doisponto = im2bw(doisponto,0.02);
algarismo0 = im2bw(algarismo0,0.02);
algarismo1 = im2bw(algarismo1,0.02);
algarismo2 = im2bw(algarismo2,0.02);
algarismo3 = im2bw(algarismo3,0.02);
algarismo4 = im2bw(algarismo4,0.02);
algarismo5 = im2bw(algarismo5,0.02);
algarismo6 = im2bw(algarismo6,0.02);
algarismo7 = im2bw(algarismo7,0.02);
algarismo8 = im2bw(algarismo8,0.02);
algarismo9 = im2bw(algarismo9,0.02);
for i = 1:2
    switch strhora(i)
        case '0'
            hora = [hora algarismo0];
        case '1'
            hora = [hora algarismo1];
        case '2'
            hora = [hora algarismo2];
        case '3'
            hora = [hora algarismo3];
        case '4'
            hora = [hora algarismo4];
        case '5'
            hora = [hora algarismo5];
        case '6'
            hora = [hora algarismo6];
        case '7'
            hora = [hora algarismo7];
        case '8'
            hora = [hora algarismo8];
        case '9'
            hora = [hora algarismo9];
    end
end
hora = [hora doisponto];
```

```
    for i = 1:2
        switch strmin(i)
            case '0'
                hora = [hora algarismo0];
            case '1'
                hora = [hora algarismo1];
            case '2'
                hora = [hora algarismo2];
            case '3'
                hora = [hora algarismo3];
            case '4'
                hora = [hora algarismo4];
            case '5'
                hora = [hora algarismo5];
            case '6'
                hora = [hora algarismo6];
            case '7'
                hora = [hora algarismo7];
            case '8'
                hora = [hora algarismo8];
            case '9'
                hora = [hora algarismo9];
        end
    end
    hora = 255*hora;
end
```

Códigos em C++

```
#include <iostream>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <algorithm>
#include <chrono>
#include <ctime>
#include <opencv2/opencv.hpp>
#include "criararvore.cpp"
#include "algoritmo.cpp"
#include "DetectaMov.cpp"
```

```
#include "AdicionaTags.cpp"
#include "CriaSequencias.cpp"
#include "conversor.cpp"
#include "Sobrepoe.cpp"

using namespace std;
using namespace cv;

int main() {
	auto start = chrono::system_clock::now();
	DetectaMov();
	AdicionaTags();
	CriaSequencias();
	Sobrepoe();
	auto end = chrono::system_clock::now();
	chrono::duration<double> elapsed_seconds = end - start;
	time_t end_time = chrono::system_clock::to_time_t(end);

	cout << "finished computation at " << ctime(&end_time)
		<< "elapsed time: " << elapsed_seconds.count() << "s\n";
	cin.get();
	return 0;
}


using namespace std;
using namespace cv;

int DetectaMov()
{
	VideoCapture vid,r;
	VideoWriter mov;
	ofstream file;
	string titulo = "C:/Users/Eduardo Orru/Documents/Pastas Rodrigo/TCC/movimentos/mova.avi";
	string titulotxt;
	Mat frameatual, frameseguinte, diff;
	Scalar movimento;
	int fourcc;
	int HoraInicial[3] = {22, 40, 25};
	int HoraVideo[3];
```

```
int flag = 0, j = -30, i, length, length2;
vid = VideoCapture("C:/Users/Eduardo Orru/Documents/Pastas
Rodrigo/TCC/Videos/WIN_20190519_00_41_45_Pro.mp4",CAP_FFMPEG);
length = int(vid.get(CAP_PROP_FRAME_COUNT));
cout << "Numero total de Frames: " << length << endl;
vid >> frameatual;
blur(frameatual, frameatual, Size(3, 3));
for (i = 1;i < length; i++)
{
        if (i % 16 < 1)
        {
                if (HoraInicial[2] < 59)
                {
                        HoraInicial[2] = HoraInicial[2] + 1;
                }
                else
                {
                        if (HoraInicial[1] < 59)
                        {
                                HoraInicial[1] = HoraInicial[1] + 1;
                                HoraInicial[2] = 0;
                        }
                        else
                        {
                                HoraInicial[0] = HoraInicial[0] + 1;
                                HoraInicial[1] = 0;
                        }
                }
        }
        vid >> frameseguinte;
        blur(frameseguinte, frameseguinte, Size(3,3));
        absdiff(frameatual, frameseguinte, diff);
        cvtColor(diff, diff, COLOR_RGB2GRAY);
        threshold(diff, diff, 51, 255, THRESH_BINARY);
        medianBlur(diff, diff, 3);
        diff = diff / 255;
        movimento = sum(sum(diff));
        if (movimento[0] > 20)
        {
                if (flag == 0)
```

```
{
        for (int k = 0; k < 3; k++)
        {
                HoraVideo[k] = HoraInicial[k];
        }
        fourcc = mov.fourcc('X', 'V', 'I', 'D');
        mov.open(titulo, fourcc, 16, Size(960, 540), true);
}
mov << frameatual;
j = i;
flag = 1;
}
else
{
if (i - j < 30)
{
        mov << frameatual;
}
else
{
        if (flag == 1)
        {
                titulotxt = titulo;
                titulotxt.replace(67, 3, "txt");
                file.open(titulotxt);
                file << HoraVideo[0] << " " << HoraVideo[1] << " " <<
HoraVideo[2];
                file.close();
                mov.release();
                cout << "Video " << titulo[65] << " Gravado" << endl;
                r = VideoCapture(titulo);
                length2 = int(r.get(CAP_PROP_FRAME_COUNT));
                if (length2 > 45)
                {
                        titulo[65] = titulo[65] + 1;
                }
        }
        flag = 0;
}
}
```

```
		frameatual = frameseguinte.clone();
	}
	if (flag == 1)
	{
		titulotxt = titulo;
		titulotxt.replace(67, 3, "txt");
		file.open(titulotxt);
		file << HoraVideo[0] << " " << HoraVideo[1] << " " << HoraVideo[2];
		file.close();
		mov.release();
		cout << "Video " << titulo[65] << " Gravado" << endl;
	}
	r.release();
	vid.release();
	return 0;
}

using namespace std;
using namespace cv;
bool FileExistenceCheck(const string& name) {
	struct stat buffer;
	if((stat(name.c_str(), &buffer) == 0))
		cout << name.c_str() << endl;
	return (stat(name.c_str(), &buffer) == 0);
}

int fileCounter(string dir, string prefix, string extension)
{
	int returnedCount = 0;
	int possibleMax = 5000000; //some number you can expect.
	string letra = "a";
	for (int istarter = 0; istarter < possibleMax; istarter++) {
		string fileName = "";
		fileName.append(dir);
		fileName.append(prefix);
		fileName.append(letra);
		fileName.append(extension);
		bool status = FileExistenceCheck(fileName);
		returnedCount = istarter;
		letra[0] = letra[0]+1;
```

```
		if (!status)
			break;
	}
	return returnedCount;
}

int AdicionaTags()
{
	string tag = "C:/Users/Eduardo Orru/Documents/Pastas Rodrigo/TCC/tags/taga";
	string titulo = "C:/Users/Eduardo Orru/Documents/Pastas Rodrigo/TCC/movimentos/mova.avi";
	string titulotxt,tagtxt;
	string line;
	int number;
	vector<int> horainicial;
	ifstream file;
	ofstream filetag;
	int length;
	vector<int> limx_max,limx_min,limy_max,limy_min,regiaox,regiaoy;
	int icm;
	int jcm;
	vector<int> lasticm, lastjcm;
	int min_x, max_x, min_y, max_y;
	int coluna;
	int contador;
	int end;
	int count = fileCounter("C:/Users/Eduardo Orru/Documents/Pastas Rodrigo/TCC/movimentos/", "mov",
".txt");
	VideoCapture vid;
	Mat frameatual, frameseguinte, diff, diffred,location,surround;
	Mat diffampliado = Mat::zeros(350, 560, CV_8U);
	Size size;
	cout << "Número de Movimentos: " << count << endl;
	for (int i = 0; i < count; i++) {
		vid = VideoCapture(titulo);
		length = int(vid.get(CAP_PROP_FRAME_COUNT));
		cout << "Número de frames: " << length << endl;
		cout << tag << endl;
		cout << titulo << endl;
		titulotxt = titulo;
		titulotxt.replace(67, 3, "txt");
```

```
file.open(titulotxt);
getline(file, line);
stringstream iss(line);
while (iss >> number) {
	horainicial.push_back(number);
}
file.close();
cout << "Hora: " << horainicial[0] << ":" << horainicial[1] << ":" << horainicial[2] << endl;
tagtxt = tag + ".txt";
filetag.open(tagtxt);
lasticm.clear();
lastjcm.clear();
for (int j = 0; j <= length - 30; j++) {
	if (j % 16 < 1) {
		if (horainicial[2] < 59) {
			horainicial[2] = horainicial[2] + 1;
		}
		else {
			if (horainicial[1] < 59) {
				horainicial[1] = horainicial[1] + 1;
				horainicial[2] = 0;
			}
			else {
				horainicial[0] = horainicial[0] + 1;
				horainicial[1] = 0;
			}
		}
	}
	icm = 0;
	jcm = 0;
	regiaox.clear();
	regiaoy.clear();
	contador = 0;
	coluna = 0;
	vid.set(1, j);
	vid.read(frameatual);
	vid.set(1, j+1);
	vid.read(frameseguinte);
	absdiff(frameatual, frameseguinte, diff);
	blur(diff, diff, Size(3, 3));
```

```
cvtColor(diff, diff, CV_BGR2GRAY);
threshold(diff, diff, 25, 255, CV_THRESH_BINARY);
medianBlur(diff, diff, 3);
Sobel(diff, diff,-1,1,1);
resize(diff, diffred,Size(0,0), 0.5, 0.5);
threshold(diffred, diffred, 25, 255, CV_THRESH_BINARY);
diffred.rowRange(0, 270).copyTo(diffampliado.rowRange(40,
310).colRange(40,520));
findNonZero(diffampliado.t(), location);
while (location.total() > 0) {
        findNonZero(diffampliado(Range(location.at<Point>(0).x-40,
location.at<Point>(0).x+40), Range(location.at<Point>(0).y-40, location.at<Point>(0).y+40)),surround);
        double minVal;
        double maxVal;
        minMaxLoc(surround.reshape(1).col(0), &minVal, &maxVal);
        if(surround.rows > 1){
                regiaox.push_back(location.at<Point>(0).x);
                regiaoy.push_back(location.at<Point>(0).y);
        } else {
                if (regiaox.size()) {
                        min_x = *min_element(regiaox.begin(), regiaox.end());
                        max_x = *max_element(regiaox.begin(), regiaox.end());
                        min_y = *min_element(regiaoy.begin(), regiaoy.end());
                        max_y = *max_element(regiaoy.begin(), regiaoy.end());
                }
                else {
                        min_x = 0;
                        min_y = 0;
                        max_x = 0;
                        max_y = 0;
                }
                if(location.at<Point>(0).y >= coluna){
                        if((max_x - min_x)*(max_y - min_y) > 200){
                                limx_min.push_back(2*(min_x - 40));
                                limx_max.push_back(2*(max_x - 40));
                                limy_min.push_back(2*(min_y - 40));
                                limy_max.push_back(2*(max_y - 40));
                                icm =  round((limx_min[contador] +
limx_max[contador])/2);
```

```
jcm =  round((limy_min[contador] +
limy_max[contador])/2);
if(icm && jcm){
for (int k = 0; k < lasticm.size(); k++) {
if (abs(icm - lasticm[k]) < 40 &&
abs(jcm - lastjcm[k]) < 40) {
icm = lasticm[k];
jcm = lastjcm[k];
}
}
if(jcm < 960-80){
if(icm < 540-80){
filetag << icm << " " << jcm << " " <<
horainicial[0] << " " << horainicial[1];
} else{
filetag << icm-19 << " " << jcm << " " <<
horainicial[0] << " " << horainicial[1];
}
} else{
if(icm < 540-80){
filetag << icm << " " << jcm-79 << " " <<
horainicial[0] << " " << horainicial[1];
} else{
filetag << icm-19 << " " << jcm-79 << " " <<
horainicial[0] << " " << horainicial[1];
}
}
}
if (lasticm.size() <= contador) {
lasticm.push_back(int());
lastjcm.push_back(int());
}
lasticm[contador] = icm;
lastjcm[contador] = jcm;
filetag << " ";
contador = contador + 1;
}
regiaox.clear();
regiaoy.clear();
}
```

```
                        }
                        diffampliado.row(location.at<Point>(0).x).col(location.at<Point>(0).y) = 0;
                        if(location.cols){
                                if(coluna < maxVal + location.at<Point>(0).y - 40){
                                        coluna = maxVal + location.at<Point>(0).y - 40;
                                }
                        }
                        location = location.rowRange(1,location.rows);
                }
                filetag << endl;
                limx_min.clear();
                limx_max.clear();
                limy_min.clear();
                limy_max.clear();
            }
            vid.release();
            filetag.close();
            end = titulo.length();
            horainicial.erase(horainicial.begin(), horainicial.end());
            titulo[end-5] = titulo[end-5] + 1;
            end = tag.length();
            tag[end-1] = tag[end-1] + 1;
        }
        return 0;
}

using namespace std;
using namespace cv;

int CriaSequencias()
{
        string tagtxt = "C:/Users/Eduardo Orru/Documents/Pastas Rodrigo/TCC/tags/taga.txt";
        string titulo = "C:/Users/Eduardo Orru/Documents/Pastas Rodrigo/TCC/movimentos/mova.avi";
        int nmax = 5;
        int count = fileCounter("C:/Users/Eduardo Orru/Documents/Pastas Rodrigo/TCC/movimentos/", "mov",
".txt");
        int end,maximo;
        vector<VideoCapture> vid;
        VideoCapture auxvid;
        vector<int> length;
```

```
int auxlength;
ifstream file;
string line;
vector<string> sequencia;
int valor ,count1, count2;
Mat sparse,tags;
Mat auxtags;
for(int i = 0; i < count; i++){
        count1 = -1;
        vid.push_back(VideoCapture(titulo));
        if (int(vid[i].get(CAP_PROP_FRAME_COUNT)) - 30 > 0) {
                length.push_back(int(vid[i].get(CAP_PROP_FRAME_COUNT)) - 30);
        }
        else {
                length.push_back(int(vid[i].get(CAP_PROP_FRAME_COUNT)));
        }
        file.open(tagtxt);
        while(!file.eof()){
                count2 = -1;
                count1 = count1 + 1;
                getline(file,line);
                if(line.length()){
                        stringstream iss(line);
                        while(iss >> valor){
                                count2 = count2 + 1;
                                sparse.push_back(i);
                                sparse.push_back(count1);
                                sparse.push_back(count2);
                                sparse.push_back(valor);
                                tags.push_back(sparse.t());
                                sparse.release();
                        }
                }
        }
        end = tagtxt.length();
        tagtxt[end-5] = tagtxt[end-5] + 1;
        end = titulo.length();
        titulo[end-5] = titulo[end-5] + 1;
        file.close();
}
```

```
for(int i = 0; i < count-1; i++){
	maximo = i;
	for(int j = i; j < count; j++){
		if(length[j] > length[maximo]){
			maximo = j;
		}
	}
	if (i != maximo){
		tags.col(0).setTo(-1,tags.col(0)==maximo);
		tags.col(0).setTo(maximo,tags.col(0)==i);
		tags.col(0).setTo(i,tags.col(0)==-1);
		auxlength = length[i];
		auxvid = vid[i];
		length[i] = length[maximo];
		vid[i] = vid[maximo];
		length[maximo] = auxlength;
		vid[maximo] = auxvid;
	}
}
sequencia = algoritmo(length,count,nmax);
string seqtitulo="C:/Users/Eduardo Orru/Documents/Pastas Rodrigo/TCC/sequencias/seqa.avi";
string seqtxt="C:/Users/Eduardo Orru/Documents/Pastas Rodrigo/TCC/sequencias/seqa.txt";
VideoWriter mov;
ofstream seqfile;
int fourcc;
vector<int> auxvec;
vector<int> min,hora,icm,jcm;
Mat seqtag;
Mat frameatual;
for(int i = 0; i < sequencia.size(); i++){
	fourcc = mov.fourcc('X', 'V', 'I', 'D');
	mov.open(seqtitulo, fourcc, 16, Size(960, 540), true);
	seqfile.open(seqtxt);
	seqtag.release();
	for(int j = 0; j < sequencia[i].size(); j++){
		for(int k = 0; k < length[int(sequencia[i][j])-97]; k++){
			vid[int(sequencia[i][j])-97] >> frameatual;
			mov << frameatual;
		}
		for(int k = 0; k < tags.rows; k++){
```

```
                        if(tags.at<int>(k,0) == int(sequencia[i][j])-97){
                                seqtag.push_back(tags.row(k));
                        }
                }
        }
        int col = 0;
        int k = 0;
        for(int j = 0; j < seqtag.rows; j++){
                if(k > seqtag.at<int>(j,1)){
                        k = 0;
                        seqfile << endl;
                }
                        while(k < seqtag.at<int>(j,1)){
                        seqfile << endl;
                        k = k + 1;
                }
                if(col == seqtag.at<int>(j,2)){
                        seqfile << seqtag.at<int>(j,3) << " ";
                } else{
                        col = 0;
                        seqfile << seqtag.at<int>(j,3) << " ";
                }
                col = col + 1;
        }
        mov.release();
    seqfile.close();
    end = seqtxt.length();
    seqtxt[end-5] = seqtxt[end-5] + 1;
    seqtitulo[end-5] = seqtitulo[end-5] + 1;
    }
    return 0;
}

using namespace std;
using namespace cv;

vector<string> algoritmo(vector<int> vetor,int count,int nmax){
    vector<Cidade> cidade;
    char s = 'a';
    int x = -1, n, a;
```

```
int soma= 0, erroaceitavel;
string stringtotal,stringaux;
vector<string> sequencia;
for(int i = 0; i < count; i++){
        cidade.push_back(Cidade());
        cidade[i].nome = s;
        s = s + 1;
        cidade[i].coordenada = vetor[i];
        cout << vetor[i] << endl;
}
for(int i = 0; i < count; i++){
        stringtotal = stringtotal +  cidade[i].nome;
}
for(int i = 0 ;i < count; i++){
        stringaux = stringtotal;
        stringaux.erase(i,1);
        cidade[i].conexao = stringaux;
}
for (int val : vetor) soma += val;
n = (soma/(nmax-1) + 3*soma/nmax)/4;
erroaceitavel = (n*nmax - soma)/(nmax+1);
while(cidade.size()){
        x = x + 1;
        vector<Arvore> arvore = criararvore(cidade,n,erroaceitavel);
        a = arvore.size() - 1;
        sequencia.push_back(string());
        sequencia[x].push_back(cidade[0].nome);
        cidade.erase(cidade.begin());
        for (int i = 0; i < cidade.size() ; i++){

cidade[i].conexao.erase(remove(cidade[i].conexao.begin(),cidade[i].conexao.end(),arvore[0].nome),cida
de[i].conexao.end());
        }
        while(arvore[a].pai != -1){
                sequencia[x].push_back(arvore[a].nome);
                cidade.erase(cidade.begin() + buscacidade(cidade,arvore[a].nome));
                for (int i = 0; i < cidade.size() ; i++){

cidade[i].conexao.erase(remove(cidade[i].conexao.begin(),cidade[i].conexao.end(),arvore[a].nome),cida
de[i].conexao.end());
```

```
        }
                        a = arvore[a].pai;
                }
        }
        for(int i = 0; i < sequencia.size(); i++){
                cout << sequencia[i] << endl;
        }
        return sequencia;
}

using namespace std;
using namespace cv;

struct Cidade
{
    char  nome;
    int coordenada;
    string conexao;
};

struct Arvore
{
        char nome;
        int no;
        int pai;
        int soma;
};

int buscacidade(vector<Cidade> cidade, char local){
        int j;
        for(j = 0; j < cidade.size(); j++){
                if(local == cidade[j].nome){
                        break;
                } else if(j == cidade.size()-1){
                        j = -1;
                        break;
                }
        }
        return j;
};
```

```
vector<Arvore> criararvore(vector<Cidade> cidade, int n, int erroaceitavel){
	char nome = cidade[0].nome;
	int noatual = 1;
	int pai = -1;
	vector<Arvore> arvore;
	int cond = 1;
	int k = -1;
	int x;
	int valor;
	int valormaximo;
	int a;
	int end;
	vector<int> valores;
	arvore.push_back(Arvore());
	arvore[0].nome = nome;
	arvore[0].no = noatual;
	arvore[0].pai = pai;
	arvore[0].soma = cidade[0].coordenada;
	while(1){
		k = k + 1;
		if(arvore.size() >= k + 1){
			for(int j = 0; j <  cidade[buscacidade(cidade,arvore[k].nome)].conexao.size(); j++){
				x = k;
				while(arvore[x].pai != -1){
					if(arvore[arvore[x].pai].nome ==
cidade[buscacidade(cidade,arvore[k].nome)].conexao[j]){
						cond = 0;
						break;
					}
					else{
						cond = 1;
					}
					x = arvore[x].pai;
				}
				if(cond == 1){
					noatual = noatual + 1;
					pai = k;
					nome = cidade[buscacidade(cidade,arvore[k].nome)].conexao[j];
```

```
                                        valor = cidade[buscacidade(cidade,nome)].coordenada +
arvore[pai].soma;
                                        if(valor <= n){
                                                end = arvore.size();
                                                noatual = noatual + 1;
                                                arvore.push_back(Arvore());
                                                arvore[end].nome = nome;
                                            arvore[end].no = noatual;
                                        arvore[end].pai = pai;
                                                arvore[end].soma = valor;
                                        }
                                }
                                end = arvore.size() - 1;
                                if(n - arvore[end].soma < erroaceitavel){
                                        valormaximo = arvore[end].soma;
                                        a = arvore.size();
                                        return arvore;
                                }
                        }
                } else {
                        return arvore;
                        break;
                }
        }
};

using namespace std;
using namespace cv;

int Sobrepoe()
{
        string titulo = "C:/Users/Eduardo Orru/Documents/Pastas Rodrigo/TCC/sequencias/seqa.avi";
        string seqtxt = "C:/Users/Eduardo Orru/Documents/Pastas Rodrigo/TCC/sequencias/seqa.txt";
        int count = fileCounter("C:/Users/Eduardo Orru/Documents/Pastas Rodrigo/TCC/sequencias/", "seq",
".txt");
        int end, maximo;
        vector<VideoCapture> vid;
        VideoCapture auxvid;
        vector<int> length;
        ifstream file;
```

```
string line;
int auxlength, count1, count2, valor;
Mat sparse, tags;
VideoWriter mov;
int fourcc;
fourcc = mov.fourcc('X', 'V', 'I', 'D');
mov.open("C:/Users/Eduardo Orru/Documents/Pastas Rodrigo/TCC/VideosFinais/videofinal.avi",
fourcc, 16, Size(960, 540), true);
double weight1, weight2;
Mat frame1, frame2;
ofstream finalfile;
vector<int> auxtag;
Mat time;
vector<Mat> channels(3);
stringstream ss1, ss2;
for (int i = 0; i < count; i++) {
        vid.push_back(VideoCapture(titulo));
        length.push_back(int(vid[i].get(CAP_PROP_FRAME_COUNT)));
        end = titulo.length();
        titulo[end - 5] = titulo[end - 5] + 1;
        file.open(seqtxt);
        count1 = -1;
        while (!file.eof()) {
                count2 = -1;
                count1 = count1 + 1;
                getline(file, line);
                if (line.length()) {
                        stringstream iss(line);
                        while (iss >> valor) {
                                count2 = count2 + 1;
                                sparse.push_back(0);
                                sparse.push_back(count1);
                                if (auxtag.size() <= count1) {
                                        sparse.push_back(count2);
                                }
                                else {
                                        sparse.push_back(count2 + auxtag[count1]);
                                }
                                sparse.push_back(valor);
                                tags.push_back(sparse.t());
```

```
                                        sparse.release();
                                }
                        }
                        if (auxtag.size() <= count1) {
                                auxtag.push_back(count2 + 1);
                        }
                        else {
                                auxtag[count1] = auxtag[count1] + count2 + 1;
                        }
                }
                end = seqtxt.length();
                seqtxt[end - 5] = seqtxt[end - 5] + 1;
                file.close();
        }
        Mat1i idx, auxidx;
        Mat result(tags.rows, tags.cols, tags.type());
        Mat finalresult;
        cv::sortIdx(tags.col(1), idx, CV_SORT_EVERY_COLUMN + CV_SORT_ASCENDING);
        for (int i = 0; i < tags.rows; i++) {
                tags.row(idx(i, 0)).copyTo(result.row(i));
        }
        int numrow = 0;
        Mat auxresult;
        for (int i = 0; i < result.rows; i++) {
                if (numrow == result.at<int>(i, 1)) {
                        auxresult.push_back(result.row(i));
                }
                else {
                        if (auxresult.rows) {
                                cv::sortIdx(auxresult.col(2), auxidx, CV_SORT_EVERY_COLUMN +
CV_SORT_ASCENDING);
                                for (int j = 0; j < auxresult.rows; j++) {
                                        finalresult.push_back(auxresult.row(auxidx(j, 0)));
                                }
                                auxresult.release();
                        }
                        while (numrow < result.at<int>(i, 1)) {
                                numrow = numrow + 1;
                        }
                        auxresult.push_back(result.row(i));
```

```
		}
	}
	cv::sortIdx(auxresult.col(2), auxidx, CV_SORT_EVERY_COLUMN + CV_SORT_ASCENDING);
	for (int j = 0; j < auxresult.rows; j++) {
		finalresult.push_back(auxresult.row(auxidx(j, 0)));
	}
	for (int i = 0; i < count - 1; i++) {
		maximo = i;
		for (int j = i; j < count; j++) {
			if (length[j] > length[maximo]) {
				maximo = j;
			}
		}
		if (i != maximo) {
			auxlength = length[i];
			auxvid = vid[i];
			length[i] = length[maximo];
			vid[i] = vid[maximo];
			length[maximo] = auxlength;
			vid[maximo] = auxvid;
		}
	}
	int col = 0;
	finalfile.open("C:/Users/Eduardo Orru/Documents/Pastas Rodrigo/TCC/VideosFinais/tagfinal.txt");
	for (int j = 0; j < finalresult.rows; j++) {
		if (col == finalresult.at<int>(j, 2)) {
			finalfile << finalresult.at<int>(j, 3) << " ";
		}
		else {
			col = 0;
			finalfile << endl;
			finalfile << finalresult.at<int>(j, 3) << " ";
		}
		col = col + 1;
	}
	for (int i = 0; i < length[0]; i++) {
		vid[0] >> frame1;
		for (int j = 1; j < count; j++) {
			if (length[j] > i) {
				weight1 = (double)(j) / (j + 1);
```

```
                        weight2 = (double)1 / (j + 1);
                        vid[j] >> frame2;
                        addWeighted(frame1, weight1, frame2, weight2, 0, frame1);
                }
            }
            for (int j = 0; j < finalresult.rows - 4; j = j + 4) {
                if (finalresult.at<int>(j, 1) == i) {
                        ss1 << finalresult.at<int>(j + 2, 3);
                        ss2 << finalresult.at<int>(j + 3, 3);
                        time = conversor(ss1.str(), ss2.str());
                        split(frame1, channels);
                        for (int k = 0; k < 3; k++) {
                                time.copyTo(channels[k](Rect(finalresult.at<int>(j + 1, 3),
finalresult.at<int>(j, 3), 80, 19)));
                        }
                        merge(channels, frame1);
                }
                else if (finalresult.at<int>(j, 1) > i) {
                        break;
                }
            }
            mov << frame1;
        }
        finalfile.close();
        mov.release();
        return 0;
}

using namespace std;
using namespace cv;

Mat conversor(string hora, string min){
        Mat
time,doispontos,algarismo0,algarismo1,algarismo2,algarismo3,algarismo4,algarismo5,algarismo6,algarismo7,al
garismo8,algarismo9;
        if(hora.length() == 1){
                hora = "0" + hora;
        }
        if(min.length() == 1){
                min = "0" + min;
```

```
        }
        doispontos = imread("C:/Users/Eduardo Orru/Documents/Pastas
Rodrigo/TCC/algarismos/doisponto.bmp",CV_LOAD_IMAGE_GRAYSCALE);
        algarismo0 = imread("C:/Users/Eduardo Orru/Documents/Pastas
Rodrigo/TCC/algarismos/algarismo0.bmp",CV_LOAD_IMAGE_GRAYSCALE);
        algarismo1 = imread("C:/Users/Eduardo Orru/Documents/Pastas
Rodrigo/TCC/algarismos/algarismo1.bmp",CV_LOAD_IMAGE_GRAYSCALE);
        algarismo2 = imread("C:/Users/Eduardo Orru/Documents/Pastas
Rodrigo/TCC/algarismos/algarismo2.bmp",CV_LOAD_IMAGE_GRAYSCALE);
        algarismo3 = imread("C:/Users/Eduardo Orru/Documents/Pastas
Rodrigo/TCC/algarismos/algarismo3.bmp",CV_LOAD_IMAGE_GRAYSCALE);
        algarismo4 = imread("C:/Users/Eduardo Orru/Documents/Pastas
Rodrigo/TCC/algarismos/algarismo4.bmp",CV_LOAD_IMAGE_GRAYSCALE);
        algarismo5 = imread("C:/Users/Eduardo Orru/Documents/Pastas
Rodrigo/TCC/algarismos/algarismo5.bmp",CV_LOAD_IMAGE_GRAYSCALE);
        algarismo6 = imread("C:/Users/Eduardo Orru/Documents/Pastas
Rodrigo/TCC/algarismos/algarismo6.bmp",CV_LOAD_IMAGE_GRAYSCALE);
        algarismo7 = imread("C:/Users/Eduardo Orru/Documents/Pastas
Rodrigo/TCC/algarismos/algarismo7.bmp",CV_LOAD_IMAGE_GRAYSCALE);
        algarismo8 = imread("C:/Users/Eduardo Orru/Documents/Pastas
Rodrigo/TCC/algarismos/algarismo8.bmp",CV_LOAD_IMAGE_GRAYSCALE);
        algarismo9 = imread("C:/Users/Eduardo Orru/Documents/Pastas
Rodrigo/TCC/algarismos/algarismo9.bmp",CV_LOAD_IMAGE_GRAYSCALE);
        double minVal,maxVal;
        Point minLoc;
        Point maxLoc;
        minMaxLoc(doispontos, &minVal, &maxVal, &minLoc, &maxLoc);
        Rect r(minLoc,Size(16,19));
        doispontos = doispontos(r);
        algarismo0 = algarismo0(r);
        algarismo1 = algarismo1(r);
        algarismo2 = algarismo2(r);
        algarismo3 = algarismo3(r);
        algarismo4 = algarismo4(r);
        algarismo5 = algarismo5(r);
        algarismo6 = algarismo6(r);
        algarismo7 = algarismo7(r);
        algarismo8 = algarismo8(r);
        algarismo9 = algarismo9(r);
        for(int i = 0; i < 2; i++){
```

```
switch(hora[i]){
        case '0':
                if(i == 0){
                        time = algarismo0;
                } else {
                        hconcat(time,algarismo0,time);
                }
                break;
        case '1':
                if(i == 0){
        time = algarismo1;
  } else {
        hconcat(time,algarismo1,time);
                }
                break;
        case '2':
                if(i == 0){
        time = algarismo2;
  } else {
        hconcat(time,algarismo2,time);
  }
                break;
        case '3':
                if(i == 0){
        time = algarismo3;
  } else {
        hconcat(time,algarismo3,time);
  }
                break;
        case '4':
  if(i == 0){
        time = algarismo4;
  } else {
        hconcat(time,algarismo4,time);
        }
                break;
        case '5':
  if(i == 0){
        time = algarismo5;
  } else {
```

```
                hconcat(time,algarismo5,time);
            }
                        break;
                case '6':
            if(i == 0){
                time = algarismo6;
            } else {
                hconcat(time,algarismo6,time);
            }
                        break;
                case '7':
            if(i == 0){
                time = algarismo7;
            } else {
                hconcat(time,algarismo7,time);
            }
                        break;
                case '8':
            if(i == 0){
                time = algarismo8;
            } else {
                hconcat(time,algarismo8,time);
            }
                        break;
                case '9':
            if(i == 0){
                time = algarismo9;
            } else {
                hconcat(time,algarismo9,time);
            }
                        break;
        }
}
hconcat(time,doispontos,time);
for(int i = 0; i < 2; i++){
        switch(min[i]){
                case '0':
            hconcat(time,algarismo0,time);
            break;
      case '1':
```

```
                hconcat(time,algarismo1,time);
                break;
            case '2':
                hconcat(time,algarismo2,time);
                break;
            case '3':
                hconcat(time,algarismo3,time);
                break;
            case '4':
                hconcat(time,algarismo4,time);
                break;
                    case '5':
                hconcat(time,algarismo5,time);
                break;
            case '6':
                hconcat(time,algarismo6,time);
                break;
            case '7':
                hconcat(time,algarismo7,time);
                break;
            case '8':
                hconcat(time,algarismo8,time);
                break;
                    case '9':
                hconcat(time,algarismo9,time);
                break;
        }
    }
        return time;
}
```

Apêndice B

Configuração de IP e de Firewall

No roteador ARRIS utilizado neste projeto o procedimento para fixar o IP de um dispositivo na rede é o seguinte: em Lan Setup é necessário selecionar Client List e clicar na opção add de Reserved IP Client List, feito isso irá aparecer o modal mostrado na Figura 3.14 a seguir.

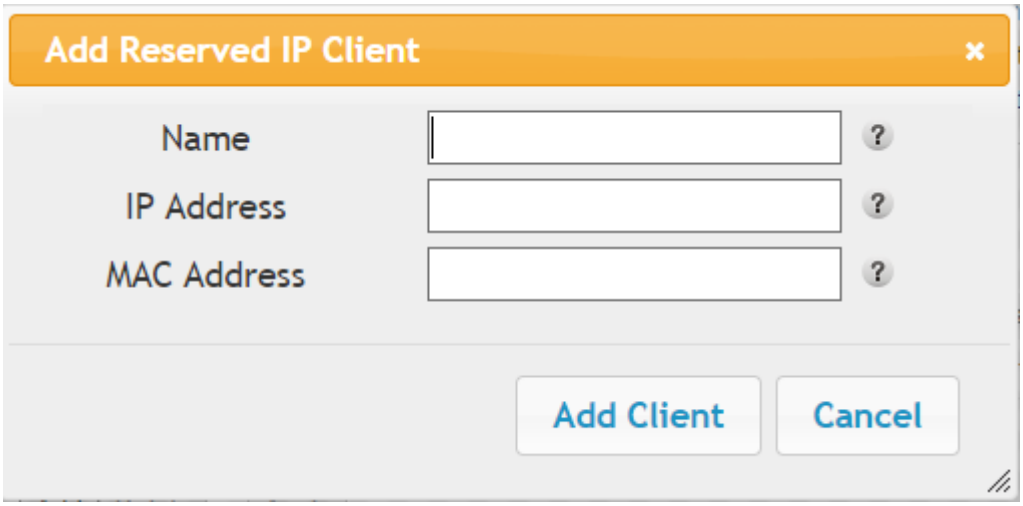

Figura B.1: Modal de Reserva de IP

Neste modal devem ser preenchidas três informações. A primeira é um nome qualquer para a máquina na qual o IP deve ser reservado, a segunda é o IP escolhido para a mesma, que deve estar dentro da faixa de IP da rede local e por fim o endereço MAC da placa de rede da máquina, no caso desse projeto, da Raspberry.

Depois de fixado o IP local da Raspberry, foi liberado no *Firewall* do roteador as portas 22000 (SSH) e a porta 5900 (VNC) para permitir acesso externo para esses serviços em específico na Raspberry. No roteador utilizado no projeto o procedimento é o descrito a seguir: Nas configurações de *Firewall* existe a opção *Virtual Server/Port Fowarding*, depois de entrar nessa opção basta selecionar a opção add. Quando isso é executado o modal mostrado na Figura 3.15 é aberto.

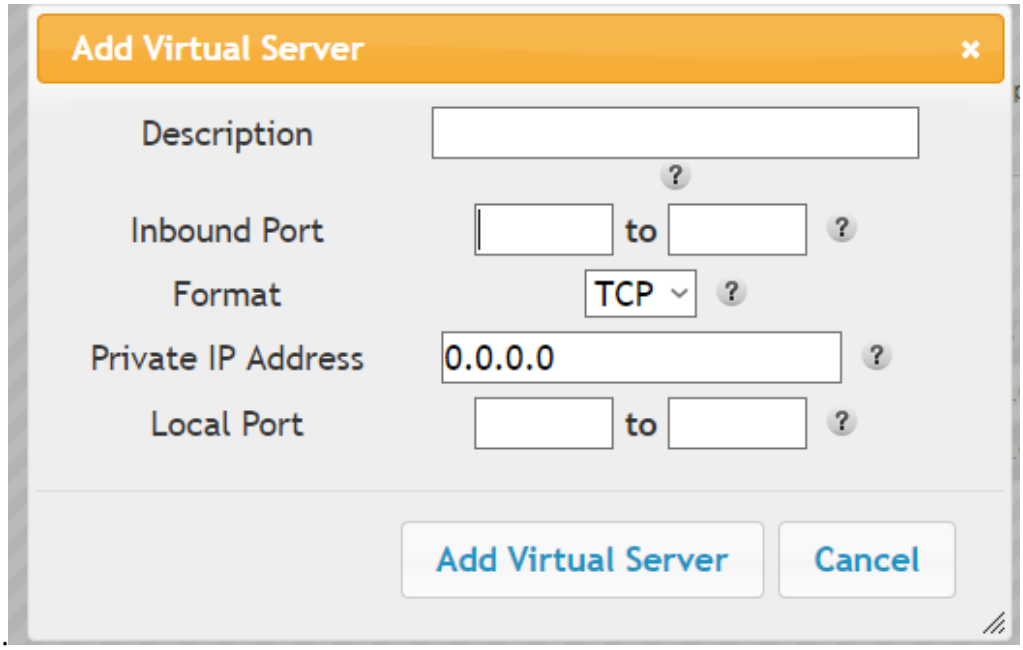

Figura B.2: Modal de *PortFowarding*

Neste modal deve ser preenchido uma descrição para identificar o serviço que está sendo liberado, o *range* de porta externo a ser liberado, no caso do projeto foram 22000-22000 e 5900-5900, o formato que foi mantido em TCP, o IP interno da máquina configurado anteriormente (Raspberry) e as portas locais, que no caso do projeto foram utilizadas as mesmas portas.