UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CAMILA ALVES PEREIRA

SÉRGIO MURILO MOREIRA DE OLIVEIRA

DETECÇÃO DE PESSOAS EM IMAGENS, IMPLEMENTANDO TÉCNICAS DE VISÃO COMPUTACIONAL EM UM *RASPBERRY PI*

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2016

CAMILA ALVES PEREIRA

SÉRGIO MURILO MOREIRA DE OLIVEIRA

DETECÇÃO DE PESSOAS EM IMAGENS, IMPLEMENTANDO TÉCNICAS DE VISÃO COMPUTACIONAL EM UM *RASPBERRY PI*

Trabalho de Conclusão de Curso como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Erikson F. de Morais

PONTA GROSSA

2016

TERMO DE APROVAÇÃO

Detecção de pessoas em imagens, implementando técnicas de visão computacional em um *Raspberry PI*

por

CAMILA ALVES PEREIRA

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 25 de maio de 2016 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. A candidata foi arguida pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Erikson Freitas de Morais
Prof.(a) Orientador(a)

_____
Geraldo Ranthum
Membro titular

_____
Richard Duarte Ribeiro
Membro titular

_____
Profª. Dra. Helyane Bronoski Borges
Responsável pelo Trabalho de Conclusão de Curso

_____
Profª. Dra. Mauren Louise Sguario
Coordenadora do curso

TERMO DE APROVAÇÃO

Detecção de pessoas em imagens, implementando técnicas de visão computacional em um *Raspberry PI*

por

SÉRGIO MURILO MOREIRA DE OLIVEIRA

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 25 de maio de 2016 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Erikson Freitas de Morais
Prof.(a) Orientador(a)

_____
Geraldo Ranthum
Membro titular

_____
Richard Duarte Ribeiro
Membro titular

_____
Profª. Dra. Helyane Bronoski Borges
Responsável pelo Trabalho de Conclusão de Curso

_____
Profª. Dra. Mauren Louise Sguario
Coordenadora do curso

Dedico aos meus pais Helton e Marli.

Dedico aos meus pais Sérgio e Edeutrudes.

AGRADECIMENTOS

Agradecemos a Deus primeiramente, por nos prover proteção, sabedoria e alento para que pudéssemos concluir este projeto e curso.

Eu, Camila, agradeço aos meus pais, Helton e Marli, por me apoiarem e estarem sempre comigo nos momentos de dificuldade.

Agradeço ao meu irmão, Diego, que participou da fase de teste do projeto.

Agradeço ao meu amigo Sérgio Murilo Moreira de Oliveira por ter aceito fazer parte desse momento.

Eu, Sérgio, agradeço aos meus pais, Sérgio e Edeutrudes, bem como aos meus irmãos, Elaine e Robson, por todo o apoio.

Agradeço à minha amiga Camila Alves Pereira por ter aceito trilhar esse caminho ao meu lado.

Ao meu amigo Silivan pela amizade mais sincera que alguém pode almejar.

Aos meus amigos Francismar, Hélio e Juliano, bem como a todos os meus amigos e integrantes da Banda Escola Lyra dos Campos, por me cederem espaço e colaborarem com a realização dos testes deste projeto.

Agradecemos também ao nosso orientador Erikson Freitas de Morais por todo seu empenho e dedicação, por suas sugestões, correções e por seu apoio.

E por último, porém não menos importante, a todos os professores que passaram por nossa trajetória.

RESUMO

PEREIRA, Camila Alves; OLIVEIRA, Sérgio Murilo Moreira de. **Detecção de Pessoas em Imagens, Implementando Técnicas de Visão Computacional em um *Raspberry Pi***, 2016. 58f Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas), Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2016.

Nos dias atuais, é cada vez maior o número de dispositivos eletroeletrônicos presentes em nossas vidas. Com o avanço tecnológico desses dispositivos, é comum que muitos permaneçam conectados à rede elétrica em *stand-by*, estado no qual continuam consumindo baixas quantidades de energia, que somadas, diante do grande número de dispositivos, acabam por gerar grande desperdício. É também cada vez maior o número de ambientes monitorados por câmeras. A observação desses dois fatos motivou o desenvolvimento desse projeto, resultando em um sistema capaz de detectar a presença de pessoas em imagens. O sistema foi desenvolvido na linguagem C++, implementando técnicas de visão computacional, com auxílio da biblioteca *OpenCV*, na plataforma *Raspberry Pi*, com o objetivo de, futuramente, processar imagens adquiridas por câmeras de vigilância e controlar a distribuição de energia elétrica do ambiente monitorado, desligando-a para aparelhos cuja função não seja necessária na ausência de pessoas, bem como religando-a caso a presença de uma pessoa seja detectada.

Palavras-chave: Visão Computacional. Processamento de Imagens. Economia de Energia. Raspberry Pi. OpenCV.

ABSTRACT

PEREIRA, Camila Alves; OLIVEIRA, Sérgio Murilo Moreira de. **Detecção de Pessoas em Imagens, Implementando Técnicas de Visão Computacional em um *Raspberry Pi***, 2016. 58f Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas), Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2016.

Nowadays, there's an increasing number of electronic devices present in our lives. With technological advancement of these devices, it's common that many of them remain connected to the power grid in stand-by, state in which continue to consume low amounts of energy, which together, due the large number of devices, end up generating a big waste. It is also increasing the number of environments monitored by cameras. The observation of these two facts motivated the development of this project, resulting in a system capable of detect the presence of people in images. The system was developed in C ++, implementing, with the help of OpenCV library, computer vision techniques in the Raspberry Pi platform, with the goal of, hereafter, process images acquired by surveillance cameras and control the power distribution of the monitored environment, turning it off for devices whose function is not required in the absence of people, as well as rebind it, case the presence of a person be detected.

Keywords: Computing Vision. Image Processing. Power Saving. Raspberry Pi. OpenCV.

LISTA DE ILUSTRAÇÕES

LISTA DE ACRÔNIMOS

FPS	Frame Por Segundo
GMM	Gaussian Mixture Model
GUI	Interface Gráfica do Usuário
MMG	Modelo de Mistura de Gaussianas
MoG	Mixture of Gaussians
OpenCV	Open Source Computer Vision Library
RGB	Red-Blue-Green

SUMÁRIO

1 INTRODUÇÃO

Atualmente, é cada vez maior o número de equipamentos eletrônicos em nossas vidas. Na correria do dia-a-dia, muitos desses equipamentos permanecem ligados, mesmo que em *stand-by*[1], ou seja, à espera de um comando, para que voltem a desempenhar sua função.

Em *stand-by* tais equipamentos continuam consumindo uma baixa quantia de energia. Parece insignificante, porém, em locais como escritórios, escolas, repartições públicas, indústrias e até mesmo em nossas casas, a grande concentração de equipamentos gera enormes prejuízos, tantos financeiros quanto ambientais (SILVA FILHO, 2008).

Estudos realizados na Inglaterra apontaram que, em 2004, aparelhos em *stand-by* foram responsáveis por 8% de todo o consumo residencial de energia elétrica (DEPARMENT OF TRADE AND INDUSTRY – UNITED KINGDOM, 2006, p. 44), e estudo similares realizados na França, apontaram que aparelhos em *stand-by* são responsáveis por 3% a 10% do consumo mundial de eletricidade em casas e escritórios (BERTOLDI, AEBISCHER, EDLINGTON, HERSHBERG, LEBOT, LIN, ... WEBBER, 2002). De acordo com os estudos, reduzir esse consumo reduziria em pelo menos 1% a emissão mundial de dióxido de carbono (CO_2).

1.1 JUSTIFICATIVA

Com a crescente preocupação com a segurança, dados os crescentes índices de violência, é cada vez mais comum a implementação de sistemas de vigilância onde câmeras de vídeos são utilizadas para capturar imagens do ambiente (CASTRO, PEDRO, 2009).

Com isso, surgiu a questão que, se houvesse um sistema capaz de processar imagens, detectando a presença de atividade humana, aproveitando as câmeras já presentes em um ambiente, seria possível desenvolver um sistema que possibilitaria economizar energia, através do desligamento da distribuição elétrica para ambientes

[1] Palavra inglesa que significa de prontidão, de sobreaviso ou à espera. Fonte: (ARAÚJO, PEREIRA, PAULINO, BARBALHO, SILVA, p. 5, 2014)

vazios, automaticamente desligando dispositivos em *stand-by*, cujo funcionamento não seja necessário na ausência de pessoas.

Surgiu então a ideia de desenvolver um sistema de processamento de imagens, que pudesse ser executado em um *hardware*[2] limitado e de baixo custo, com desempenho que possibilitasse a utilização do sistema no mundo real, a fim de extrair das imagens informações necessárias para saber se há ou não a presença de pessoas em determinado ambiente.

1.2 ESCOPO DO TRABALHO

O escopo deste trabalho consiste no desenvolvimento de um algoritmo na linguagem *C++*, implementando conceitos de visão computacional com o auxílio da biblioteca *OpenCV*[3]*,* capaz de processar imagens capturadas por uma câmera, e detectar a presença de pessoas em tais imagens.

O algoritmo foi executado em um *Raspberry Pi*[4].

A versão do *Raspberry Pi* utilizada no desenvolvimento do projeto foi *Raspberry Pi B+*, cujas as especificações são:

- Tamanho: 85 x 85mm;
- Processador: Broadcom SoC 700 MHz;
- Memória RAM: 512 MB.

A câmera utilizada foi a *Raspicam*, câmera desenvolvida exclusivamente para uso com o *Raspberry Pi*, modelo *Raspicam NoIR* versão 1.3 de 5 *Megapixel*.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

O objetivo geral do projeto foi desenvolver um sistema capaz de detectar a presença de pessoas em imagens.

---

[2] Palavra inglesa que se refere às partes físicas que compõem um aparelho. Fonte: (OFICINADANET, 2016)

[3] *Open Source Computer Vision Library* – Coleção de bibliotecas relacionadas a Visão Computacional. Fonte: (OPENCV, 1999)

[4] Computador de baixo custo do tamanho de um cartão de crédito. Fonte: (RASPBERRY, 2016)

1.3.2 Objetivo Específico

Os objetivos específicos do projeto foram:

- Desenvolver um sistema capaz de processar imagens capturadas por uma câmera, detectando pessoas presentes nas imagens*;*
- Desenvolver o sistema de forma que o mesmo possa ser executado em um *Raspberry Pi on the fly.*

1.4 ESTRUTURA DO TRABALHO

O primeiro capítulo refere-se à introdução, onde são apresentadas de forma resumida a problemática, a justificativa do tema, o escopo e os objetivos do trabalho.

O segundo capítulo, compreende o Referencial Teórico, subdividido em Visão Computacional, *Raspberry Pi*, *OpenCV*, Detecção, Suavização Gaussiana, Mistura de Gaussianas, Segmentação de Imagens, Morfologia Matemática.

No terceiro capítulo são apresentados trabalhos relacionados ao tema.

No quarto capítulo são descritos a metodologia e os resultados dos experimentos efetuados durante o desenvolvimento do trabalho, sendo subdividido em Aquisição, Modelagem de *Background*, Segmentação, Detecção e Testes Efetuados.

O quinto capítulo se refere à Conclusão do trabalho, subdivido em Conclusão e Trabalhos Futuros, seguido das Referências Bibliográficas.

2 REFERENCIAL TEÓRICO

2.1 VISÃO COMPUTACIONAL

Para o ser humano, identificar objetos em uma imagem é uma tarefa simples. O cérebro processa as imagens captadas pelos olhos, e, em uma fração de segundo, somos capazes de identificar, diferenciar e classificar todos os elementos presentes na imagem captada (TRAFTON, 2014).

Porém, quando a imagem é captada por uma máquina, o cenário é diferente, pois além de a qualidade da imagem captada ser proporcional à qualidade do equipamento utilizado na captação, independente do meio de captação utilizado, as máquinas sempre interpretam a imagem da mesma forma, ou seja, como uma matriz, onde cada uma das posições carrega um valor a ser convertido em um sinal de luz no momento da exibição da imagem.

A Visão Computacional dedica seus esforços ao propósito de fazer com que as máquinas sejam capazes de interpretar imagens captadas de forma similar à efetuada pelo ser humano (CARVALHO, SILVA, REBELLO, VIANA, 2003).

Visão Computacional é a ciência e tecnologia das máquinas que enxergam. De acordo com Ballard e Brown (BALLARD, BROWN, p. 2, 1982):

> "Visão Computacional é a iniciativa de automatizar e integrar uma ampla gama de processos e representações utilizadas para a percepção visual. Ela abrange partes de muitas técnicas úteis por si só, tais como processamento de imagem (transformação, codificação e transmissão de imagens) e classificação de padrões estatísticos (teoria da decisão estatística aplicada a padrões gerais, visual ou não). Mais importante para nós, ele inclui técnicas para modelagem geométrica e processamento cognitivo. "

Esta pode ser considerada uma ciência muito recente. Em seu artigo, Milano e Honorato afirmam que:

> "Uma das primeiras menções sobre Visão Computacional data de 1955, onde Selfridge destacou "... *eyes and ears for the computer*" (DAVID, SELFRIDGE, TELEPHONE apud MILANO, HONORATO, p. 1, 1962).

Ainda sobre o assunto, Neves, Vieira Neto e Gonzaga afirmam:

> "A Visão Computacional procura integrar as áreas do processamento digital de imagens e inteligência artificial, tendo como objetivo a obtenção de algoritmos capazes de interpretar visual de imagens. Suas aplicações estão presentes em diversos segmentos tecnológicos que envolvem análise de imagens, reconhecimento de padrões e controla inteligente, abrangendo múltiplas áreas do conhecimento, tais como agronomia, astronomia, biologia, biometria, medicina e muitas outras. Constitui, portanto, uma área multidisciplinar com muitas aplicações práticas" (NEVES, VIEIRA NETO, GONZAGA, p. 1, 2012).

2.2 *RASPBERRY PI*

Raspberry Pi é um computador de baixo custo do tamanho de um cartão de crédito, conforme a Figura 1.

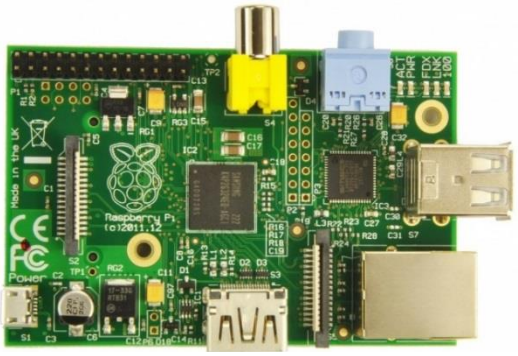

Figura 1 - *Raspberry Pi*
Fonte: (HACKERBOARDS, 2015)

Pode ser conectado a um monitor de computador ou TV, utilizando um teclado padrão e um mouse. Oferece ainda suporte a outros periféricos, conforme o diagrama apresentado na Figura 2.

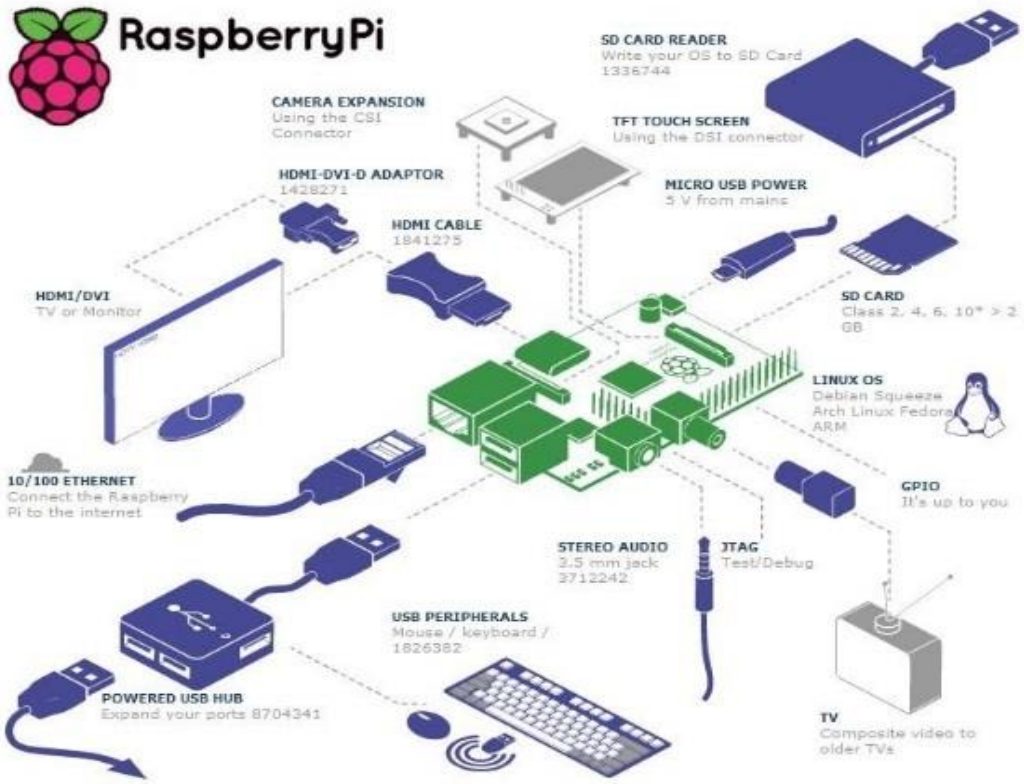

Figura 2 - Diagrama de conectividade do *Raspberry Pi*
Fonte: (HACKERBOARDS, 2015)

Teve seu desenvolvimento inicializado em 2006, no laboratório de computação da Universidade de Cambridge – Reino Unido, com o objetivo de permitir que pessoas de todas as idades pudessem explorar o mundo da computação, aprendendo a programar em diversas linguagens. Apesar de seu tamanho, é capaz de fazer tudo que se pode esperar de um *desktop* normal, como navegar na internet, assistir vídeos, desenvolver planilhas e até jogar (RASPBERRY ABOUT US, 2016).

Preocupados com o decrescente número de criança interessadas em aprender sobre programação nos anos 2000, bem como a queda nas habilidades das crianças que demonstravam interesse em aprender sobre programação às crianças dos anos 1990, os criadores do *Raspberry Pi*, Eben Upton, Rob Mullins, Jack Lang e Alan Mycroft idealizaram um computador pequeno e acessível às crianças, que pudesse inseri-las no universo da programação.

Em 2008, conforme os processadores desenvolvidos para celulares foram se tornando mais acessíveis, bem como suficientemente potentes para oferecer um desempenho satisfatório, o projeto do *Raspberry Pi* começou a se tornar viável.

Eben, Rob, Jack e Alan fizeram então uma parceria com Pete Tomas, para tornar o projeto real, e três anos depois o *Raspberry PI* modelo B começou a ser produzido em massa, vendendo mais de 2 milhões de unidades nos 2 anos seguintes.

O *Raspberry* Pi utiliza o sistema operacional *Raspibian*, uma versão não oficial do *Debian Wheezy*[5] otimizada para funcionar com instruções avançadas da arquitetura ARM v6 presente no processador do *Raspberry Pi*.

2.3 *OPENCV*

Open Source Computer Vision Library (OpenCV) é uma biblioteca multiplataforma de livre uso, tanto acadêmico e comercial. Originalmente teve seu desenvolvimento iniciado pela Intel em 1999 (OPENCV ABOUT, 1999).

Oferece módulos de processamento de imagens e vídeo i/o[6], Interface Gráfica do Usuário (GUI), Álgebra Linear (os computadores interpretam imagens como matrizes de caracteres), Estrutura de Dados, controle de mouse e teclado, bem como

[5] *Debian* é um grupo de desenvolvedores o qual desenvolve e distribui versões livres do sistema operacional Linux. *Whezzy* é uma dessas versões, lançada em 2013. Fonte: (DEBIAN, 2015)
[6] Está expressão é muito utilizada na computação, e significa a abreviatura de Input/Output, ou Entrada/Saída em nosso bom português. Fonte: (OFICINADANET, 2015)

mais de 2.500 algoritmos de Visão Computacional, com funções que permitem filtragem de imagens, calibração de câmeras, reconhecimento de objetos, entre outros, processando imagens em tempo real.

2.4 DETECÇÃO

O processamento de imagens digitais vem se tornando objeto de grande interesse, pois viabiliza diversas aplicações em categorias bem distintas: (1) aprimoramento de informações pictóricas para interpretação humana; e (2) a análise automática por computador de informações extraídas de uma cena.

Os elementos básicos de um sistema de processamento de imagem digital são apresentados na Figura 3. Podem ser tomados como base de qualquer sistema de processamento, desde o mais simples e de baixo custo até o mais sofisticado, pois abrange as principais operações possíveis de se efetuar em uma imagem: aquisição, processamento, armazenamento e exibição. Vale ressaltar que uma imagem pode também ser transmitida à distância utilizando meios de comunicação disponíveis (MARQUES FILHO, VIEIRA NETO, 1999, cap. 1, p. 2).

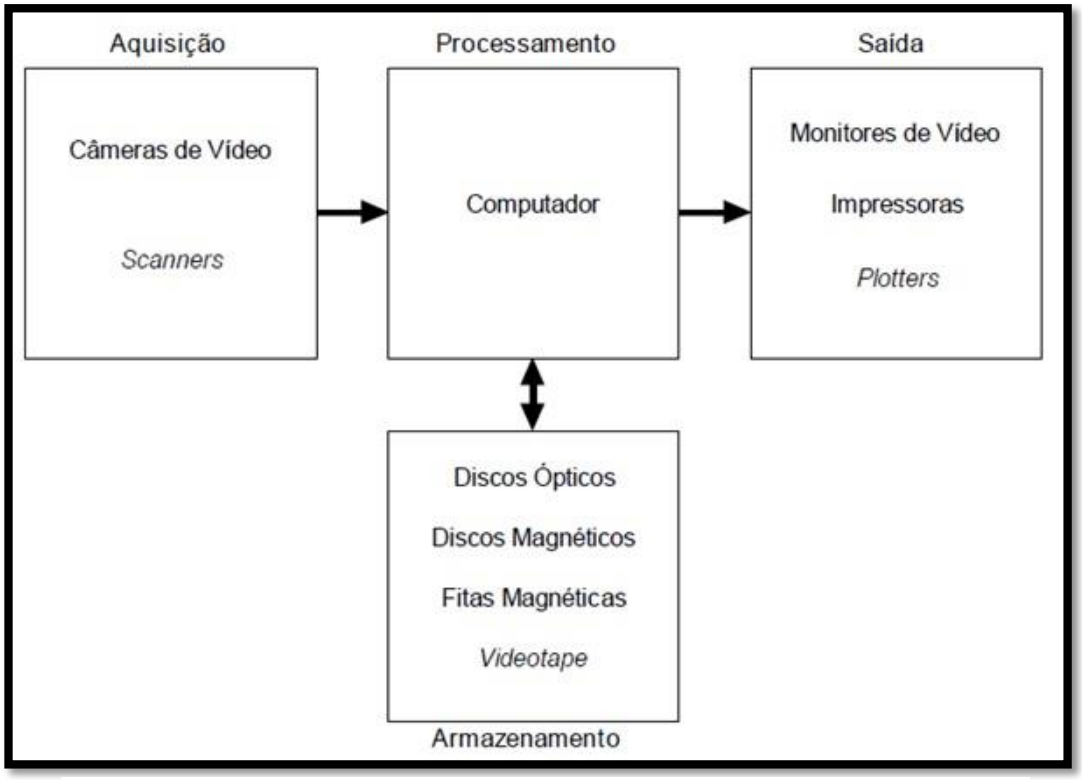

Figura 3 - Elementos de um sistema de processamento de imagens
Fonte: (MARQUES FILHO, VIEIRA NETO, 1999, cap. 1, p. 2)

O termo imagem digital, ou simplesmente imagem, refere-se à função bidimensional de intensidade da luz $f(x, y)$, em que x e y denotam as coordenadas espaciais e o valor de f em qualquer ponto (x, y) é proporcional ao brilho (ou nível de cinza) da imagem naquele ponto.

A Figura 4 ilustra a convenção dos eixos utilizada na representação de uma imagem digital monocromática para o processamento.

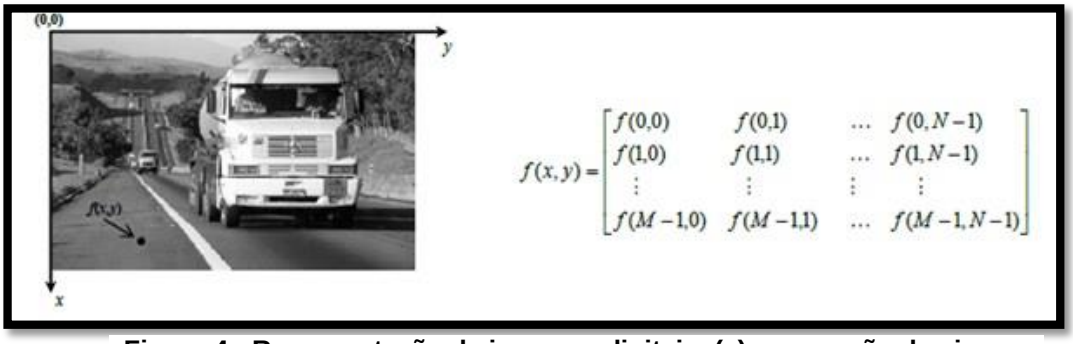

Figura 4 - Representação de imagens digitais: (a) convenção de eixos e (b) matriz de pixels
Fonte: (CUNHA, SETTI, GONZAGA, 2013)

Cada *pixel*[7] possui uma cor que varia de acordo com o nível de cinza $f(x, y)$, quantizado em uma escala de acordo com a resolução de imagem. Quanto mais alta a resolução da imagem maior o número de cores que ela pode apresentar. Por exemplo, em imagens coloridas *Red-Blue-Green* (RGB), cada pixel armazena três valores referentes a intensidade de luz, um para cada plano de cor representado por $f(x, y)$ = (R, G, B), ou seja, as imagens coloridas possuem 24 bits (3 planos de 8 bits), possuindo assim mais de 16 milhões de cores diferentes, ao passo que uma imagem de um bit, ou imagem binária, apresenta apenas dois níveis de cinza, preto ou branco (CUNHA, SETTI, GONZAGA, p. 3, 2013).

A primeira etapa de um sistema automático de processamento de imagens é a detecção de objetos, pois cada sistema desenvolvido possui o propósito de detectar algo. Tradicionalmente, a técnica mais empregada é a subtração de imagens. Sendo cada imagem uma matriz de *pixels*, subtrai-se uma da outra *pixel*-a-*pixel*, resultando em uma imagem que classifica os *pixels* com maiores resultados como objetos em movimento, enquanto *pixels* com menores resultados como plano de fundo.

[7] Termo proveniente de contração da expressão *Picture element* (“elemento da imagem” em inglês), corresponde à menor unidade de uma imagem digital, ou seja, um elemento dento da matriz (x, y). Fonte: (SIQUEIRA, p. 10, 2011)

A Figura 5 ilustra a subtração em *frames*[8], onde o *background model* (modelo de plano de fundo) é subtraído do *current frame* (frame atual). O frame resultante é submetido a um processo de *threshold* (processo de segmentação de imagem, que transforma uma imagem colorida em uma imagem binária, ou seja, preto e branco) e exibido como *foreground mask* (plano frontal da imagem) (CUNHA, SETTI, GONZAGA, p. 4, 2013).

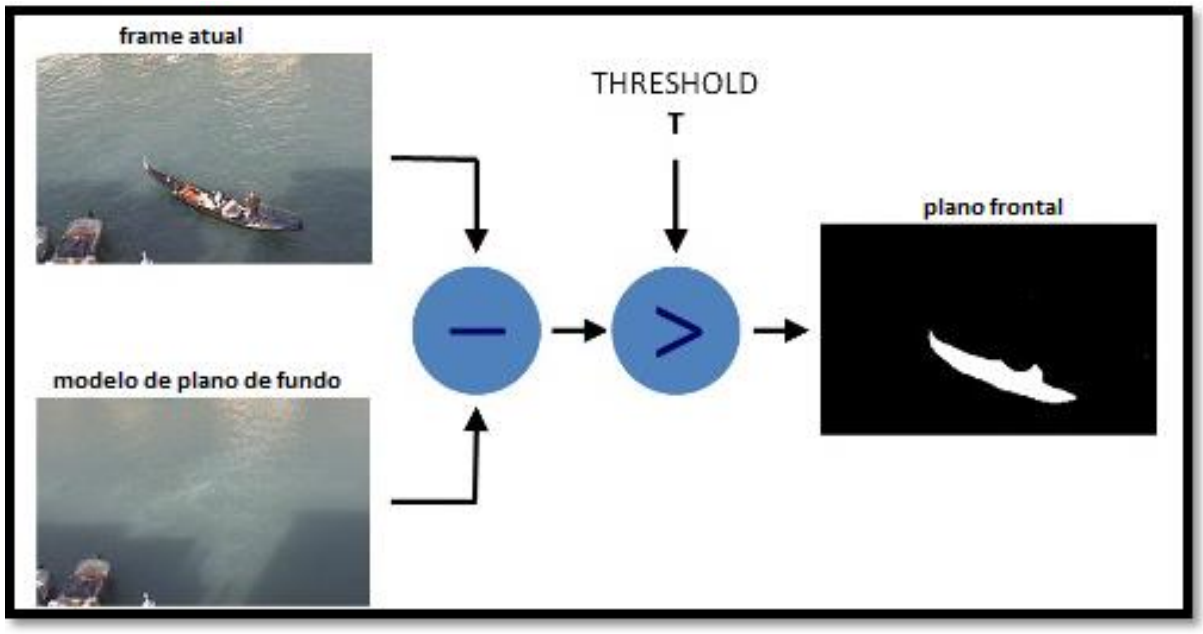

Figura 5 - Subtração de *frames*

Fonte: (OpenCV, 2016)

Efetuadas as operações descritas, o plano frontal exibirá os objetos que se movimentam na imagem.

A modelagem do *background* é a parte mais importante dos algoritmos de subtração de plano de fundo. As imagens obtidas do mundo real apresentam problemas que interferem na precisão do modelo. Ruídos (variações aleatórias de brilho e cor não condizentes com a realidade. Produzidos pelos dispositivos de entrada), variações na luminosidade da cena e variação na posição de objetos na cena, são problemas que comumente ocorrem.

O desempenho da subtração de plano de fundo é diretamente dependente da técnica utilizada na modelagem do *background*. Como na maioria dos estudos a

[8] (Em Português: quadro ou moldura) é cada um dos quadros ou imagens fixas de um produto audiovisual. Fonte: (AUMONT, MARIE, 2001)

câmera é assumida como fixa e estática (sem qualquer movimentação), o modelo de *background* deve construir e manter uma representação estatística da cena que a câmera captura.

Após a análise dos resultados apresentados no projeto similar (CUNHA, 2013), referenciado na seção Trabalhos Relacionados, concluiu-se que o método mais adequado aos objetivos do projeto era o de Subtração de *Background*.

Foi também analisada a documentação da biblioteca *OpenCV* referente à métodos de subtração do plano de fundo, através da qual concluiu-se que o método de subtração de plano de fundo mais adequado ao desenvolvimento do projeto seria o MoG (*Mixture of Gaussians* - Mistura de Gaussianas).

2.5 SUAVIZAÇÃO GAUSSIANA

Durante o processo de aquisição de uma imagem digital, podem ocorrer diversos problemas, devido a limitação de *hardware*, durante o processo de quantização[9], digitalização[10] ou transmissão, entre outros, contaminando a imagem adquirida com ruídos.

Por isso é comum que imagens digitais sejam submetidas a filtros, cujo o objetivo é salientar determinados aspectos e atenuar ruídos.

Filtros podem ser classificados de acordo com a frequência de detalhes eliminada ou mantida na imagem, sendo divididos em:

> "Filtro passa-baixas: Permite que os sinais com frequência abaixo de uma frequência determinada passem para a saída, eliminando todos os sinais com frequências superiores. Filtro passa-altas: Funciona de maneira inversa ao passa-baixas. Deixando passar para a saída apenas os sinais cujas frequências estejam acima de um certo valor. Filtro passa-faixa: Permite a seleção de apenas uma faixa de frequências, ou seja, apenas essa faixa (intervalo) selecionada passará para a saída do filtro. Filtro rejeita-faixa: Atua de forma inversa ao filtro passa-faixa, eliminando os sinais contidos em um determinado intervalo de frequências definido" (VITORINO, p. 2-3).

O processo de suavização é um dos primeiros passos de vários métodos utilizados em sistemas de visão computacional. O filtro gaussiano, apresentado na seção seguinte, é um dos filtros de suavização mais utilizados atualmente.

---

[9] Processo de atribuição de valores discretos para um sinal cuja amplitude varia entre infinitos valores. No processamento de imagens, o valor que tais valores podem assumir é determinando pelo nível de cor. Fonte: (FARIA, p.11, 2010)

[10] Processo pelo qual uma imagem ou sinal analógico é transformado em código digital. Fonte: (DIGITALIZAX, 2016).

2.5.1 Filtro Gaussiano

É um tipo de filtro de suavização de imagem que utiliza funções gaussianas no cálculo da transformação a ser aplicada em cada *pixel*, podendo ser utilizado como um filtro passa-baixas, apresentado nas Figuras 6 e 7.

O filtro gaussiano em 1-D tem a forma:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

Onde σ é o desvio padrão. É assumido que a distribuição tem média zero (i.e. está centrada em x = 0). A distribuição para $\sigma = 1$ é ilustrada na Figura 6 (OLIVEIRA JUNIOR, p. 3-4, 2006).

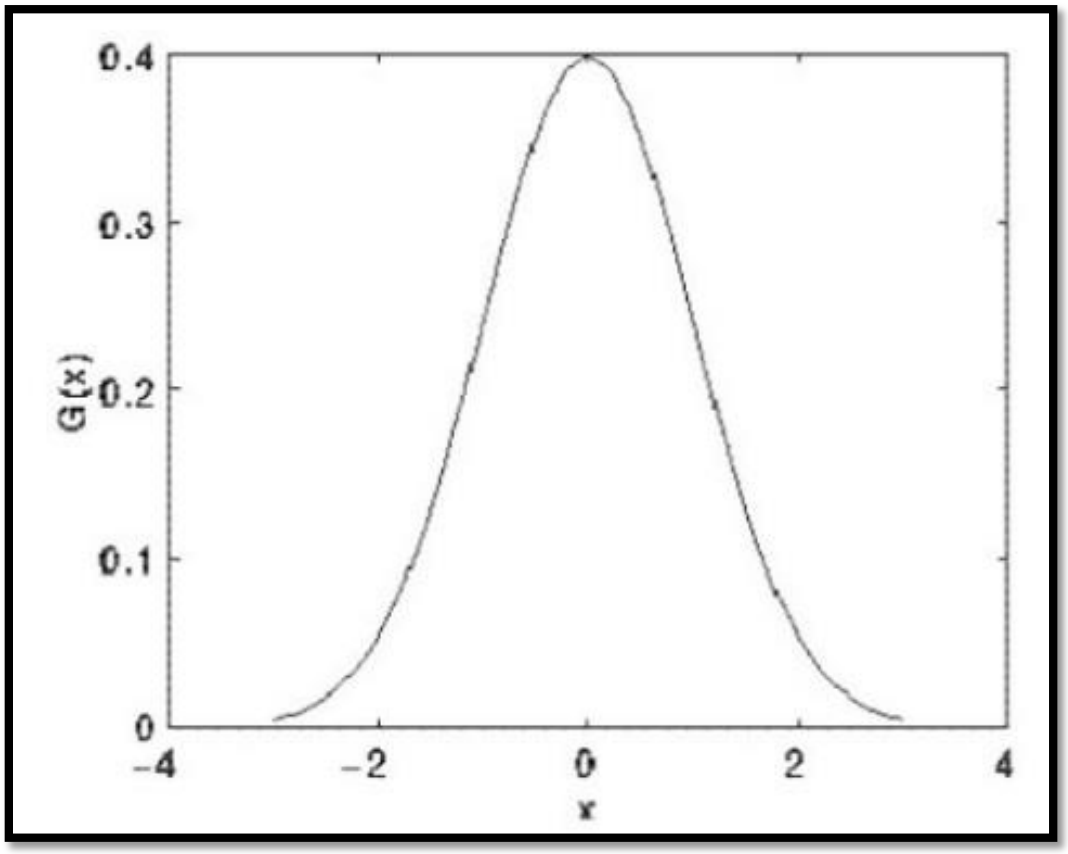

Figura 6 - Forma 1-D da distribuição Gaussiana com média zero (0) e, o σ, desvio padrão um (1)

Fonte: (OLIVEIRA JUNIOR, 2006).

Em sua isotrópica (i.e. circularmente simétrica), é dada pela equação: $G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$, demonstrada na Figura 7:

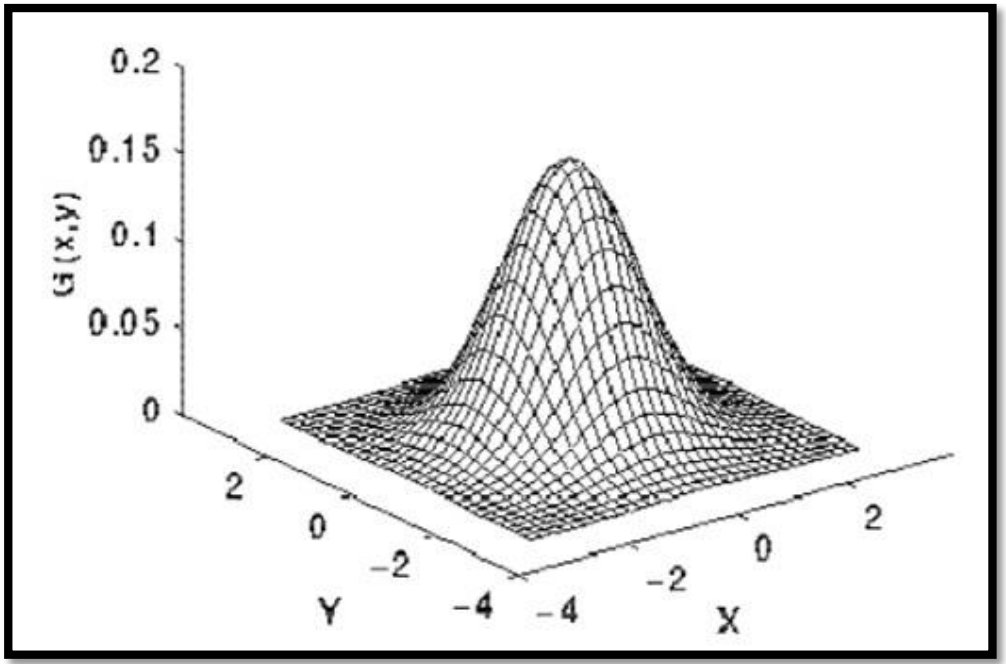

Figura 7 - Representação da função gaussiana em 2D com média (0, 0) e desvio padrão σ = 1

Fonte: (OLIVEIRA JUNIOR, 2006).

A Figura 8 demonstra o efeito da aplicação do filtro de suavização gaussiano com desvio padrão baixo (StDev = 3), bem como com desvio padrão alto (StDev = 10).

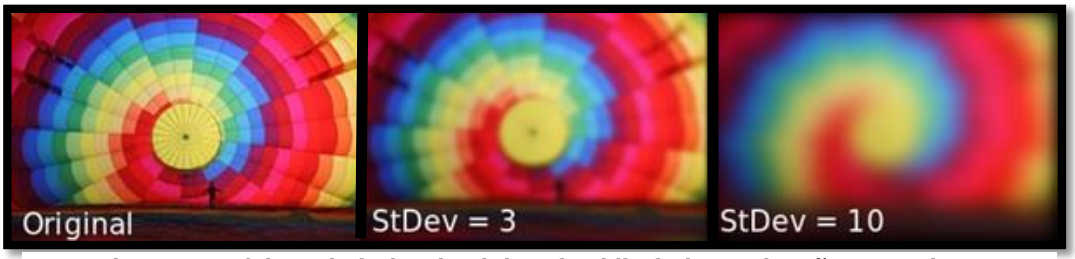

Figura 8 - Efeitos da baixa (meio) e alta (direita) suavização gaussiana

Fonte: (WIKIPEDIA, 2016)

2.6 MISTURA DE GAUSSIANAS

Mistura de Gaussianas (ou do inglês *Mixture of Gaussians* - MoG) são largamente utilizadas no processamento de sinais e análise de dados dos mais diversos tipos.

Dada uma variável X de dimensão d e uma mistura com K componentes, a função de probabilidade de uma mistura de gaussianas pode der definida por: $\Phi(X|\ominus_k) - \sum_{i=1}^{K} \pi\phi(X|\theta_i)$, onde cada θ_i corresponde ao conjunto de parâmetros definidos pela i-ésima componente da mistura, $\pi_i \in [0,1]$ com $i \in (1,2,\dots,K)$ e $\sum_{i=1}^{k} - 1$. O vetor $\ominus_k - (\pi_1,\dots,\pi_k,\theta_1,\dots,\theta_k)$ é o conjunto dos parâmetros da mistura. Cada componente $\phi(X|\theta_i)|$ da mistura é uma função de densidade de probabilidade Gaussiana definida por: $\phi(X - x|\theta_i) - \frac{1}{(2\pi^{\frac{d}{2}})|\sum i|^{\frac{1}{2}}}\, e^{\frac{1}{2}(x\ \mu_1)^t \sum_1 1(x\ \mu_1)}$, onde μ_i é a média, $\sum_I$ é a matriz de covariância e $\theta_i - (\mu_i, \sum_i)$representa os parâmetros de uma Gaussiana.

Especificamente, dado um conjunto de dados S com N instâncias s_t e dimensão d, os parâmetros de $\ominus_k$são estimados maximizando o logaritmo da seguinte função de verossimilhança: $L(\ominus_k |S) - \log \prod_{t=1}^{N} \Phi(s_t |\ominus_k) - \sum_{t=1}^{n} \log \sum_{i=1}^{k} \pi_i\, \phi(s_t|\, \theta_i)$ (SILVA FILHO, DREWS-JR, MARCOLINO, p. 2, 2013).

Para cada propósito pode ser criado um GMM (*Gaussian Mixture Model* – Traduzido para o português como MMG – Modelo de Mistura de Gaussianas), ou seja, uma soma ponderada das densidades de M componentes Gaussianos como representados na equação, $p(x|\lambda) = \sum_{i=1}^{M} w_i g(x|\mu_{i\,\sum_i})$, onde x é um D-dimensional vetor de valores contínuos (ou seja, medidas ou características), w_i, i = 1, ..., M, a densidade dos componentes Gaussianos. A densidade de cada componente é uma função Gaussiana D-variável da formula, $g(x|\mu_i, \sum_i) = \frac{1}{(2\pi)^{\frac{D}{2}}|\sum_i|^{\frac{1}{2}}} \exp\{-\frac{1}{2}(x - \mu_i)^1 \sum_i^{-1}(x - \mu_i)$, com vetor médio μ_i e matriz de covariância $\sum_i$. O peso das misturas satisfaz a constante $\sum_{i=1}^{M} w_i = 1$.

O MMG completo é parametrizado pelos vetores médios, pelas matrizes de covariância e pelo peso da mistura das densidades de todos os componentes. Esses parâmetros são coletivamente representados pela notação $\lambda = \{w_i, \mu_i, \sum_i\}$ i = 1, ..., M" (REYNOLDS, p.1, 1992).

No processamento de imagens, a Mistura de Gaussianas é geralmente empregada na obtenção do plano de fundo (*background*) de uma imagem à qual se deseja extrair informações, sendo aplicada em cada *pixel*, a fim de detectar a diferença entre duas imagens. Os *pixels* correspondentes onde não houve mudança são "descartados", ou seja, considerados parte do plano de fundo, recebendo o valor 0 (preto) no processo *threshold*, enquanto os *pixels* correspondentes onde houveram

mudanças resultam no contorno dos objetos, recebendo o valor 255 (branco) no processo de *threshold* (SILVA, GONZAGA, p.2, 2006).

A Figura 9 mostra, na janela à esquerda, a imagem real captada pela câmera. Na janela central, o modelo de plano de fundo, e na janela à direita, o plano frontal após processamento por MoG.

Figura 9 – Imagem real, plano de fundo e plano frontal
Fonte: Autoria Própria

No processamento de imagens, o número de gaussianas é correspondente à resolução das imagens envolvidas no processamento, logo, é conhecido, colaborando para a redução do custo computacional, valendo ressaltar que o custo computacional é diretamente proporcional ao número de gaussianas a serem processadas.

2.7 SEGMENTAÇÃO DE IMAGENS

Segmentar uma imagem, significa agrupar *pixels* em regiões salientes da imagem, ou seja, regiões correspondentes às superfícies individuais, objetos ou partes naturais de objetos.

Existem diversas técnicas de segmentação, que levam em considerações diferentes atributos da imagem, como cor, brilho, movimento ou disparidade em informações (JEPSON, FLEET, 2007).

As técnicas de segmentação utilizadas em imagens digitais em níveis de cinza (tipo de imagem digital utilizado nesse projeto) geralmente são baseadas em uma das propriedades dos valores de níveis de cinza: similaridade e descontinuidade.

Na similaridade, as principais abordagens são baseadas em limiares (*Thresholding*), aglomeração (*Clustering*), junção e separação (*Split & Merge*) e crescimento de regiões (*Region Growing*).

2.7.1 *Thresholding*

Segmentar a imagem tomando como base os tons de cinza que compõe seus objetos. Dadas as características dos objetos os quais se deseja isolar, é estabelecido um limiar. A imagem é então dividida em dois grupos de *pixels*, os maiores e os menores que recebe mesmo valor fixo. Isto posto, uma imagem $g(x,y)$ submetida ao processo de *thresholding* pode ser definida por como: (BERTHOLDO, p.23, 2007)

$$g(x,y) = \begin{cases} 1\ se\ f(x,y) > T \\ 0\ se\ f(x,y) \leq T \end{cases}$$

Resultando em uma imagem $g(x,y)$ binária, ou seja, com somente dois valores possíveis para cada *pixel*.

Alternativamente, é o possível isolar somente um dos grupos separados pelo limiar, sem que os valores originais dos *pixels* restantes sejam alterados, de acordo com a função:

$$g(x,y) = \begin{cases} 1\ se\ f(x,y) > T \\ f(x,y) se\ f(x,y) \leq T \end{cases}$$

Resultando em uma imagem $g(x,y)$ onde os níveis de cinza $f(x,y)$ maiores que o limiar T são saturados, enquanto os valores menores ou iguais a T manterão as cores originais.

É possível ainda truncar os valores dos níveis de cinza de uma imagem, de forma a limitar os valores $g(x,y)$ ao intervalo (0, T), conforme a fórmula:

$$g(x,y) = \begin{cases} T\ se\ f(x,y) > T \\ f(x,y) se\ f(x,y) \leq T \end{cases}$$

A Figura 10 demonstra o efeito do processo de *threshold* resultando da primeira fórmula previamente demonstrada.

Figura 10 - À esquerda, a imagem original. À direita, a imagem submetida ao *threshold*
Fonte: (WIKIPEDIA, 2016)

2.8 MORFOLOGIA MATEMÁTICA

A morfologia matemática é um ramo de estudo dedicado à compreensão das estruturas geométrica das entidades presentes em uma imagem.

O princípio básico da morfologia matemática consiste em extrair as informações relativas à geometria e à topologia de um conjunto desconhecido (uma imagem), pela transformação através de outro conjunto completamente definido, chamado elemento estruturante. Portanto, a base da morfologia matemática é a teoria de conjuntos. Por exemplo, o conjunto de todos os *pixels* pretos em uma imagem binária descreve completamente a imagem (uma vez que os demais pontos só podem ser brancos). Em imagens binárias, os conjuntos em questão são membros do espaço inteiro bidimensional Z^2, onde cada elemento do conjunto é um vetor 2-D cujas coordenadas são coordenadas (x, y) do *pixel* preto (por convenção) na imagem. Imagens com mais níveis de cinza podem ser representadas por conjuntos cujos elementos estão no espaço Z^3. Neste caso, os vetores têm três elementos, sendo os dois primeiros as coordenadas do *pixel* e o terceiro nível de cinza (MARQUES FILHO, VIEIRA NETO, 1999, cap. 5, p. 139).

As duas operações morfológicas básicas consistem em dilatação e erosão. Para sua compreensão, é necessário que algumas definições úteis de conjuntos sejam apresentadas.

Sejam A e B conjuntos em Z^2, cujos componentes são $a = (a_1, a_2)$ e $b = (b_1, b_2)$, respectivamente. A translação de A por $x = (x_1, x_2)$, denotada $(A)_x$ é definida como: $(A)_x = \{c | c = a + x, para\ a\ \epsilon\ A\}$. A reflexão de B, denotada $\hat{B}$, é definida como: $\hat{B} = \{x | x = -b\ para\ b\ \epsilon\ B\}$. O complemento do conjunto A é: $A^c = \{x | x\ \notin A\}$. Finalmente, a diferença entre dois conjuntos A e B, denotada A – B, é definida como: $A - B = \{x | x\ \in A, x\ \notin B\} = A \cap B^c$ (MARQUES FILHO, VIEIRA NETO, 1999, cap. 5, p. 140).

2.8.1 Dilatação

"Sejam A e B conjuntos no espaço Z^2 e seja∅ o conjunto vazio. A dilatação de A e B, denotada A $\oplus$ B, é definida como: $A \oplus B = \left\{x \middle| \left(\hat{B}\right)_x \cap A \neq \emptyset\right\}$. Portanto, o processo de dilatação consiste em obter a reflexão de B sobre sua origem e depois

deslocar esta reflexão de x. A dilatação de A por B é, então, o conjunto de todos os x deslocamentos para os quais a interseção de $\left(\hat{B}\right)_x$ e A inclui pelo menos um elemento diferente de zero. Com base nesta interpretação, a equação anterior pode ser escrita como: $A \oplus B = \left\{x \middle| \left[\left(\hat{B}\right)_x \cap A\right] \subseteq A\right\}$. O conjunto B é normalmente denominado elemento estruturante.

A Figura 11 mostra, à esquerda a imagem original e à direita a imagem dilatada.

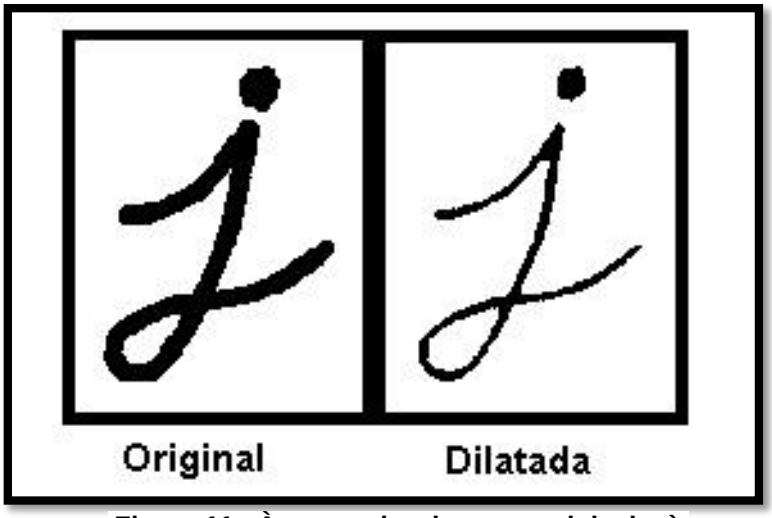

Figura 11 - À esquerda a imagem original e à direita a imagem dilatada.
Fonte: (OpenCV, 2016)

2.8.2 Erosão

Sejam A e B conjuntos no espaço Z^2. A erosão de A por B, denotada A Θ B, é definida como: $A \Theta B = \{x|(B)_x \subseteq A\}$ o que, em outras palavras significa dizer que a erosão de A por B resulta no conjunto de pontos x tais que B, transladado de x, está contido em A. A dilatação e a erosão são operações duais entre si com respeito a complementação e reflexão. Ou seja, $(A \Theta B)^c = A^c \Theta \hat{B}$. A prova desta dualidade está demonstrada a seguir: Partindo da definição de erosão, temos: $(A\Theta B)^c = \{x|(B)_x \subseteq A\}^c$. Se o conjunto $(B)_x$ está contido no conjunto A, então $(B)_x \cap A^c = \emptyset$. Portanto, a equação anterior torna-se: $(A\Theta B)^c = \{x|(B)_x \cap A^c = \emptyset\}^c$. Porém, o complemento do conjunto dos x's que satisfazem $(B)_x \cap A^c = \emptyset$ é o conjunto dos x's tais que $(B)_x \cap A^c \neq \emptyset$. Logo, $(A\Theta B)^c = \{x|(B)_x \cap A^c \neq \emptyset\} = A^c \oplus \hat{B}$ (MARQUES FILHO, VIEIRA NETO, 1999, cap. 5, p. 142 e 143).

A Figura 12 mostra, à esquerda a imagem original e à direita a imagem erodida.

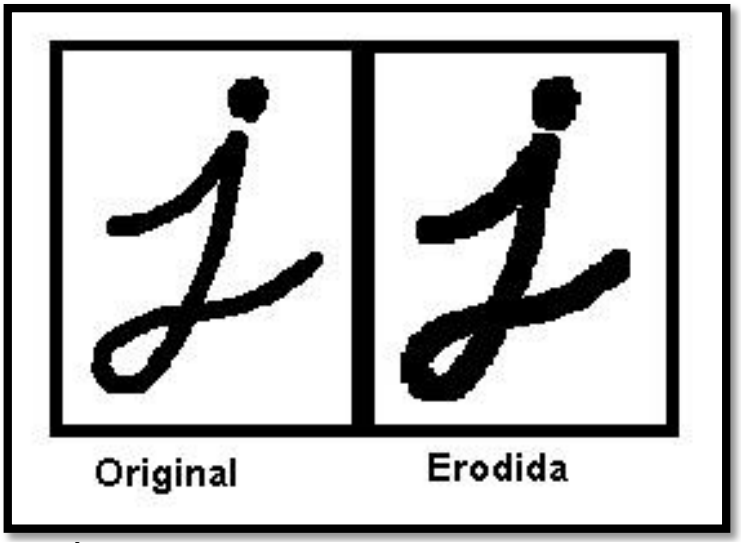

Figura 12 - À esquerda a imagem original e à direita a imagem erodida
Fonte: (OpenCV, 2016)

3 TRABALHOS RELACIONADOS

Há uma vasta gama de trabalhos cujo foco é a localização de pessoas ou determinados objetos em imagens digitais.

Porém, o projeto difere quanto aos dispositivos utilizados. Enquanto a maioria dos trabalhos pesquisados é implementada em componentes de *Hardware* robustos, a implementação do processamento de imagens em plataformas de menor desempenho, como o *Raspberry Pi*, é um campo cujo potencial ainda é pouco explorado.

O trabalho de Sonaglio e Moecke, intitulado como “Detecção de Movimento para Sistema Automático de Vigilância Por Vídeo”, constitui um projeto desenvolvido utilizando câmeras de baixo custo para captar imagens, que são armazenadas somente quando há detecção de movimento significativo, facilitando a análise das imagens por parte de um operador humano, bem como evitando a necessidade de acompanhamento constante das imagens.

Os objetivos do projeto, de acordo com os autores, foram:

> “...realizar a detecção de movimento baseado apenas em vídeo, sendo capaz de eliminar informações irrelevantes captadas na aquisição da imagem, como ruído de aquisição e aquisição e condições” (SONAGLIO, MOECKE, 2009, p. 2).

O protótipo do sistema foi desenvolvido utilizando a ferramenta Matlab, e o VCEL, algoritmo que detecta quando a imagem está muito escura ou clara e ajusta o tempo de exposição à luz, melhorando o contraste da imagem e minimizando os efeitos de variação na iluminação ambiente do ambiente.

Nos testes efetuados, ocorreram falhas de detecção quando as pessoas estavam muito distantes da câmera, ou utilizavam roupas que se assemelhavam ao plano de fundo da imagem. De acordo com os autores, tais falhas provavelmente seriam reduzidas, caso câmeras de maior resolução fossem utilizadas.

Ocorreram ainda falsos alarmes, devido a movimentação de vegetação, sombras e variação na luminosidade no ambiente. Os falsos alarmes foram significativamente reduzidos com a implementação do VCEL.

Enquanto o sistema apresentado neste projeto foi programado manualmente em C++, o sistema de Sonaglio e Moecke foi desenvolvido com o auxílio da ferramenta Matlab. Não é mencionado o equipamento utilizado no processamento das imagens, porém, através da interpretação do contexto do sistema, concluiu-se que o mesmo foi

implementado nos mesmos servidores onde as imagens capturadas pelo sistema são armazenadas, ou seja, em um *hardware* de maior robustez.

O sistema funciona de forma similar ao desenvolvido neste projeto, porém com objetivos diferentes.

Outro projeto com etapas similares de detecção de *background* e segmentação dos objetos alvo (nesse caso, caros), desenvolvendo código em Linguagem C++, fundamentando nos conceitos de Visão Computacional, bem como utilizando a biblioteca *OpenCV*, foi desenvolvido por Cunha, sob o título de "Sistema automático para obtenção de parâmetros de tráfego veicular a partir de imagens de vídeo usando *OpenCV*".

De acordo com o autor, a meta e objetivos do projeto foram:

> "...desenvolver um sistema de coleta automática de dados do tráfego veicular a partir do pós-processamento de imagens de vídeo em rodovias. Para tanto, foram propostos os seguintes objetivos:
> - A partir do estado de técnica, definir o melhor método para detecção segmentação de veículos em imagens de rodovias;
> - Estabelecer, a partir do estado da técnica, o melhor processo para rastreamento de veículos em imagens de vídeo e rodovias;
> - Desenvolver uma ferramenta de software que permita a coleta automática de parâmetros de corrente de tráfego, a partir de imagens de vídeos em rodovias, usando técnicas de segmentação e rastreamento desenvolvido nas etapas anteriores.
>
> Assim, o método proposto para se atingir a meta e os objetivos consiste nas seguintes etapas:
> - Definir um método para gerar imagens atualizadas de *background* em tempo de execução do vídeo;
> - Definir um método de segmentação de veículos;
> - Extrair os parâmetros de tráfego a partir dos veículos segmentados nas imagens" (CUNHA, 2013, p. 16 e 17).

O autor enfatiza a fundamental importância da dedicação ao aprendizado da linguagem de programação, visto que C++, em conjunto com Matlab, tem sido o meio mais eficaz de comunicação entre pesquisadores da área de Visão Computacional.

Quanto à modelagem de background, o autor concluiu primeiramente que uma imagem adequada de background aumenta a chance de uma melhor segmentação dos frames da corrente de tráfego. Investigou os seis modelos de geração de background conceituados na literatura. Media, Mediana, Mistura de Gaussianas, Moda, Probabilidade e Scoreboard, concluindo após testes, que Scoreboard era o modelo mais eficiente.

Quanto aos métodos de segmentação, investigou três métodos: subtração de fundo, mapa de texturas e segmentação baseada em texturas, concluindo-se que o

Background Subtraction era o mais adequado para a segmentação dos veículos nas imagens para o sistema proposto.

Por fim, foram aplicadas as técnicas estudadas, resultando num sistema que, por um lado se mostrou eficiente no que diz respeito a representação da velocidade dos veículos, com taxa de acerto de 92,7%, de acordo com autores, bem como uma boa aderência da distribuição das velocidades do método, de acordo com os testes estatísticos de Qui-quadrado e Kolmogorov-Smirnov.

Porém, não apresentou resultados satisfatórios na taxa de fluxo de tráfego, quando comparados às coletas manuais, necessitando maior robustez na identificação dos veículos no diagrama.

O sistema apresentou ainda, uma ferramenta iterativa que permite extrair o cumprimento dos veículos.

Contrastando com o sistema defendido neste projeto, o sistema desenvolvido por Cunha contou com *hardware* robusto, sendo implementado em um computador Core i7 3,4 GHz com 8 Gb de memória RAM. Foi desenvolvido de forma similar, e submetido a diversos testes estatísticos. Com objetivos similares, diferiu em relação aos objetos de interesse, bem como em relação aos objetivos, apresentando resultados satisfatórios no geral, bem como informações importantes, que em muito colaboraram no desenvolvimento de nosso projeto.

4 METODOLOGIA

Após elucidados os objetivos do projeto, foram analisados projetos similares. Definiu-se que o trabalho seria dividido nas seguintes etapas: aquisição, modelagem de *background*, e por fim segmentação e detecção.

4.1 AQUISIÇÃO

Para utilizar o *Raspberry Pi*, é necessário instalar seus componentes, bem como configurá-lo para o primeiro uso. Tais configurações podem ser realizadas de acordo com as instruções fornecidas pelo fabricante (RASPBERRYPI WHAT YOU WILL NEED, 2015).

A instalação do módulo *Rapicam*, câmera utilizada no desenvolvimento do projeto, exige configuração adicional (CAMERAMODULE, 2015). As especificações detalhadas da câmera podem ser encontradas no site do fabricante (CAMERA, 2015).

Como um dos objetivos é que o sistema seja compatível com diversos tipos de câmera, o código referente ao projeto foi desenvolvido de forma que a alocação da câmera é a única coisa que precisa ser modificada, caso uma nova câmera seja utilizada. A biblioteca *OpenCV* oferece uma função automática de busca por dispositivos de captura de imagem, cuja implementação é feita de forma simples, com uma única linha de código, exemplificada a seguir:

“CvCapture* capture = cvCaptureFromCAM(CV_CAP_ANY);”.

O código acima instancia um objeto “CvCapture”, chamado “capture”, criando uma sessão de câmera “cvCaptureFromCAM”, cujo o parâmetro “(CV_CAP_ANY)” faz com que o sistema busque e utilize o primeiro dispositivo USB de captura de imagem encontrado.

Uma lista de câmeras compatíveis com *Raspberry Pi* pode ser encontrada no endereço presente na seguinte referencia (RPi USB WEBCAMS, 2015).

Devido à ausência de recursos financeiros, não foi possível efetuar testes com outros modelos de câmera.

Durante o desenvolvimento do projeto, a única câmera com a qual o código foi testado foi a *Raspicam*. Está câmera utilizada com a biblioteca *OpenCV,* exige a

adição de uma biblioteca exclusiva, o que altera também a forma como a câmera é utilizada dentro do código. No exemplo a seguir, é possível comparar a alocação da *Raspicam* com a alocação genérica de câmera anteriormente citada:

“raspicam::RaspiCam_CvCamera;”.

Iniciaram-se os testes de captura de imagem, a fim de descobrir quais as resoluções suportadas pela câmera, a qualidade das imagens adquiridas, bem como o impacto das resoluções no processamento, vista a capacidade limitada do *hardware*.

O primeiro teste foi realizado na resolução máxima da câmera, 1920x1080 *pixels*, cujo resultado pode ser observado na Figura 13. Foram efetuadas duas marcações em preto na parede (circuladas em vermelho na imagem) para que fosse possível comparar diferenças na área de imagem captada pelas diferentes resoluções. A taxa de atualização obtida foi de aproximadamente 1,5 *frames* por segundo (FPS), considerada baixa em se tratando de um vídeo.

Em seguida, foi efetuado o redimensionamento da imagem. Ou seja, a imagem ainda era adquirida em 1920x1080, e então redimensionada para a resolução de 640x480 *pixels*. Foi observada baixa perda de qualidade na imagem, porém, a taxa de atualização em Frames por Segundo (FPS) se manteve similar ao teste anterior. A imagem redimensionada pode ser observada na Figura 14.

Figura 13 - Imagem capturada pela Raspicam na resolução de 1920x1080 *pixels*
Fonte: Autoria Própria

Figura 14 - Imagem capturada na resolução 1920x1080 pixels e redimensionada para a resolução 640x480 pixels
Fonte: Autoria Própria

Os testes seguintes foram efetuados decrescendo gradativamente as resoluções utilizadas.

Ao fim dos testes, concluiu-se que a resolução de captura ideal é de 640x480, pois apresentou quebras quase imperceptíveis na sequência de exibição de frames, levando à conclusão que a taxa de FPS foi superior à 20. Além disso, foi observado um ligeiro aumento no campo visual da câmera, de acordo com as marcas no cenário de testes, bem como baixa perda na qualidade de definição da imagem adquirida.
As conclusões anteriormente descritas podem ser observadas na Figura 15.

Figura 15 - Imagem capturada na resolução de 640x480 pixels
Fonte: Autoria Própria

4.2 MODELAGEM DE *BACKGROUND*

Como descrito na seção 2.5 Suavização Gaussiana, diversos problemas podem comprometer a qualidade da imagem adquirida. O processo de modelagem de *background*, como o nome sugere, é responsável por criar o modelo de plano de fundo que será utilizado nas operações conseguintes. Logo, quanto mais preciso for o modelo criado, maior será a taxa de sucesso na detecção dos objetos desejados.

Visando corrigir problemas de aquisição, antes da modelagem de *background,* as imagens captadas são submetidas a um filtro de suavização gaussiano.

Sobre isso, Gonzalez e Woods afirmam que:

> "Os filtros lineares de suavização são utilizados para borrar e remover ruídos. Durante tarefas de pré-processamento, as imagens são borradas com o objetivo de remover de pequenos detalhes antes da extração de objetos (grandes) e conectar pequenas descontinuidades em linhas ou curvas. A redução de ruído pode ser obtida ao borrar uma imagem com um filtro linear e também pela filtragem não linear" (GONZALEZ, WOODS, 2010 p. 119).

Na Figura 16, é possível observar o resultado da aplicação de um filtro de suavização gaussiano.

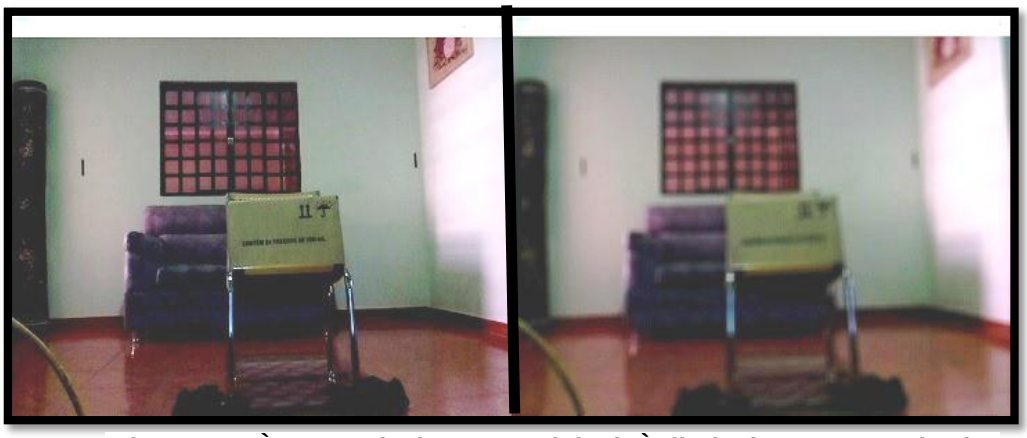

Figura 16 - À esquerda, imagem original. À direita imagem suavizada.
Fonte: Autoria Própria

Após efetuado o processo de suavização, o modelo de *background* foi gerado, com base nos conceitos apresentados no tópico Mistura de Gaussianas.

Como o plano de fundo pode sofrer diversas alterações, como modificação na localização de objetos ou adição de objetos ao ambiente, é necessário que o mesmo seja atualizado.

A biblioteca *OpenCV* oferece funções para modelagem de plano de fundo, cuja taxa de aprendizado, ou seja, a frequência de atualização do plano de fundo pode

ser determinada no momento da implementação. Dentro dos propósitos do projeto foi determinada uma taxa de aprendizado que fizesse com que um objeto se tornasse parte do plano de fundo após aproximadamente 5 minutos e 50 segundos. Houveram variações nesse tempo quando o objeto não permaneceu totalmente estático. O resultado pode ser observado na Figura 17.

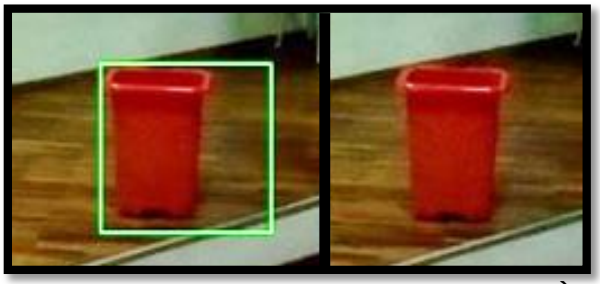

Figura 17 – À esquerda o objeto detectado pelo sistema. À direita o objeto como parte do plano de fundo.
Fonte: Autoria Própria

4.3 SEGMENTAÇÃO E DETECÇÃO

Após a modelagem de *background*, é necessário segmentar a imagem, conforme descrito na seção 2.7 Segmentação de Imagens, a fim de separar os objetos desejados do modelo de plano de fundo.

O projeto implementa o método de *thresholding* binário, descrito pela primeira fórmula previamente descrita na seção 2.7.1 *Thresholding*, cujos resultados podem ser vistos na Figura 19.

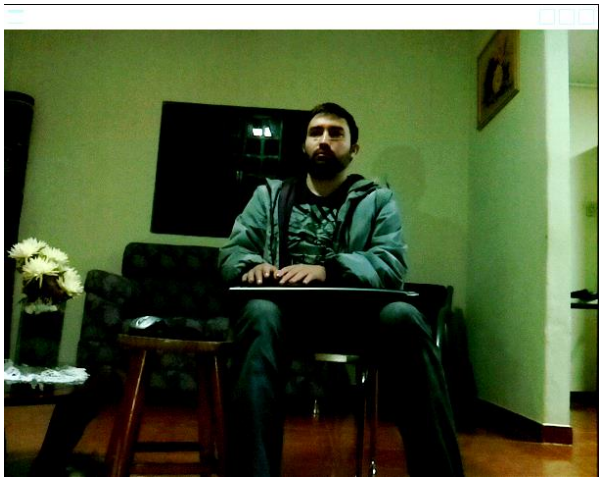

Figura 18 - Imagem real
Fonte: Autoria Própria

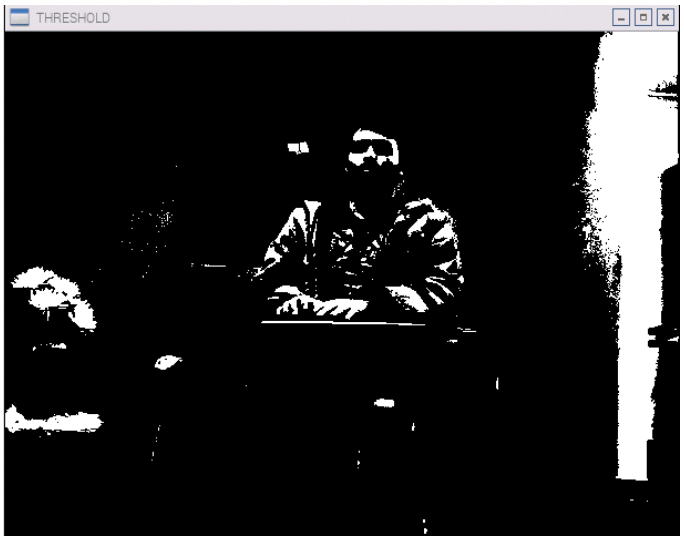

Figura 19 - Imagem submetida ao processo de thresholding
Fonte: Autoria Própria

Após efetuado o *Thresholding*, é possível observar que o resultado converte as cores mais escuras da imagem para preto, enquanto as mais claras para branco.

As cores que serão transformadas em preto, bem como quais serão transformadas em branco, variam de acordo com a sensibilidade aos níveis de cinza, definida no momento da aplicação do *threshold*. A fim de tornar o objeto detectando mais “sólido”, é aplicada a morfologia matemática de erosão, descrita na seção 2.8 Morfologia Matemática.

O resultado obtido pode ser observado na Figura 20.

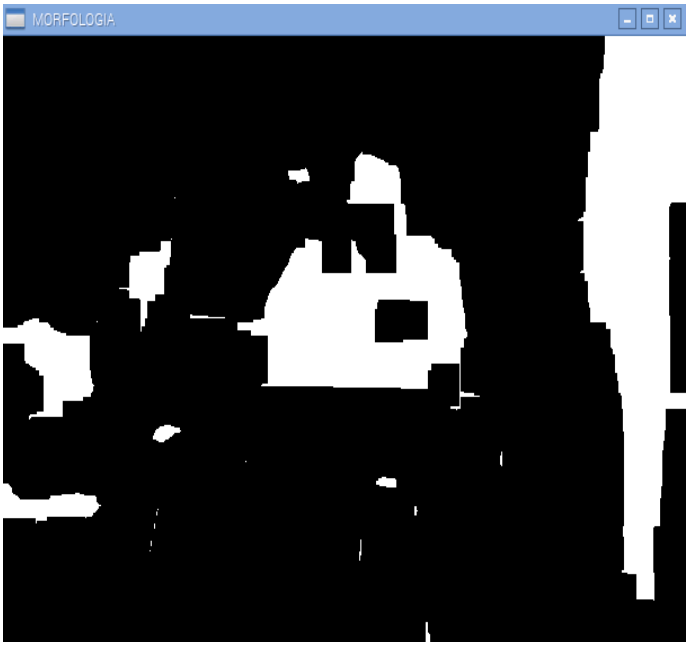

Figura 20 - Imagem submetida à morfologia matemática
Fonte: Autoria Própria

Efetuados os processos de segmentação, a imagem está pronta para que a detecção seja efetuada. O método de detecção utilizado foi a detecção de contornos, considerando apenas o contorno externo dos objetos detectados na imagem. Determinou-se que os objetos detectados seriam filtrados de acordo com o perímetro de seu contorno (em *pixels*).

Nos testes efetuados tomou-se como parâmetro que os objetos detectados deveriam ocupar, no mínimo, 5% e, no máximo, 25% da área da imagem.

O perímetro equivalente às áreas acima mencionadas foi calculado através da seguinte fórmula:

$$\sqrt{\left[(A * L) * \left(\frac{P}{100}\right)\right]} * 4$$

Onde:

A = Altura da imagem capturada (em *pixels*).

L = Largura da imagem capturada (em *pixels*).

P = Proporção da imagem (em porcentagem).

As proporções foram escolhidas a fim de evitar que *blobs*[11] muito pequenos (como os gerados por pequenos animais que por ventura cruzassem a imagem) ou muito grandes (como uma lagartixa que, por ventura possa caminhar sobre a câmera, obstruindo seu campo visual) fossem detectados.

Durante o desenvolvimento do projeto, foi observado que a versão atual da biblioteca *OpenCV* não oferece suporte para que a função de balanço automático de branco na imagem seja desativada na *Raspicam*.

Devido a essa função, mudanças na iluminação do ambiente ocasionam mudanças no padrão de cores captadas pela câmera, gerando ruídos na imagem que não puderam ser tratados, gerando falsas detecções.

De acordo com a documentação da biblioteca *OpenCV*, a função não é suportada na versão 2.4 utilizada no projeto, nem na versão 3.0, que encontra-se em fase beta de teste (OPENCV READING AND WRITING VIDEO, 2015).

[11] Do Inglês *Binary Large Object, Basic Larg Object*, *BLOB* ou *BLOb*, que significa objeto grande binário ou objeto grande básico na tradução literal. É uma coleção de dados binários armazenados como uma única entidade em um sistema de gerenciamento de banco de dados. Blobs geralmente são objetos de imagem, áudio ou outros objetos multimídia. Fonte: (SISTEMAEMBARCADOS, 2016).

4.4 TESTE EFETUADOS

a) Detecção simples de um *blob* na imagem.

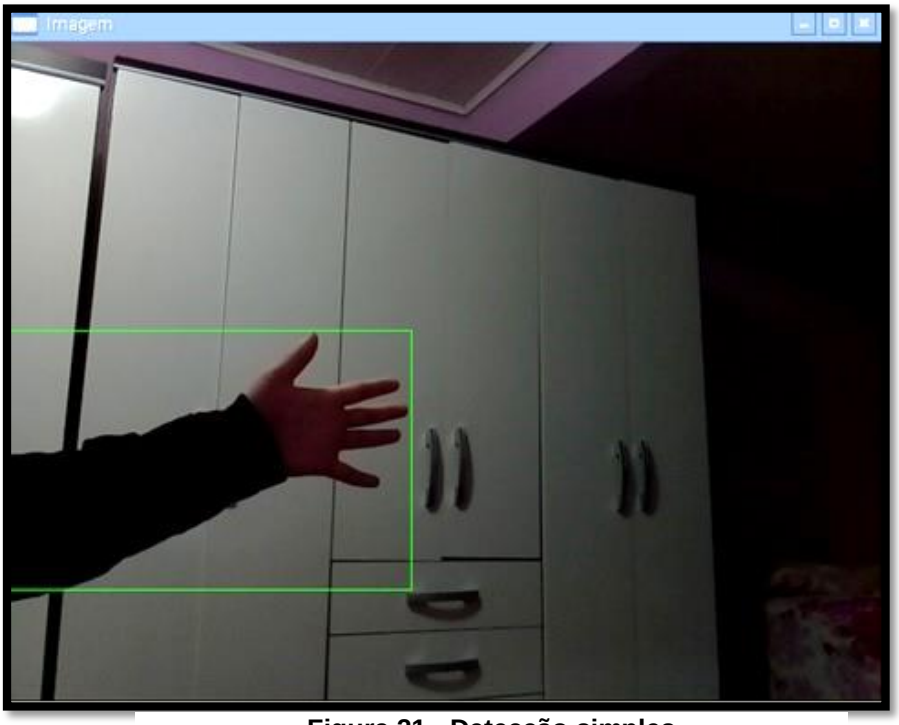

Figura 21 - Detecção simples
Fonte: Autoria Própria

b) *Blob* extrapolando o limite de 25%, sendo descartado como detecção.

Figura 22 - Blob extrapolando o limite de 25%
Fonte: Autoria Própria

c) Tamanho máximo que um *blob* pode ocupar dentro da imagem para ser considerado uma detecção.

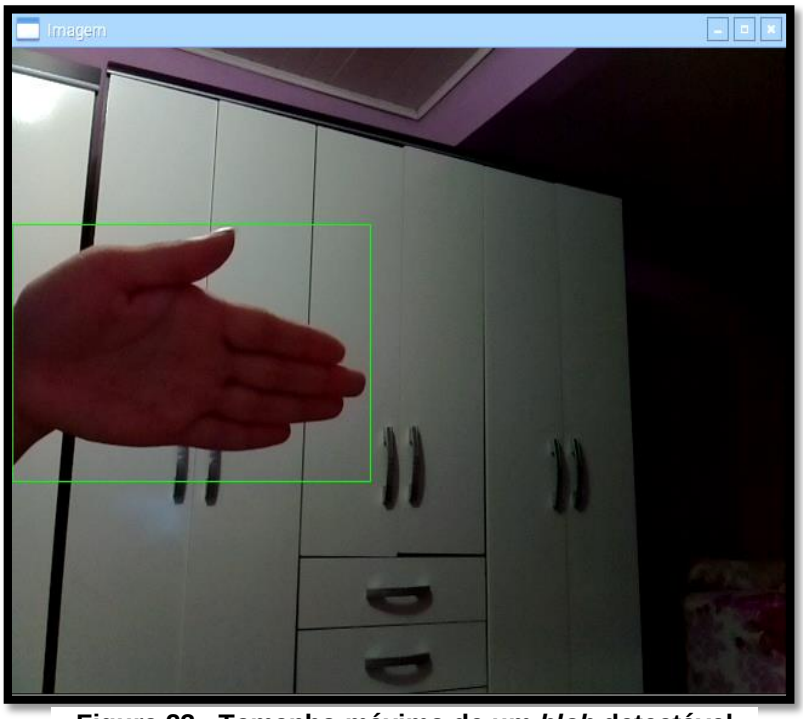

Figura 23 - Tamanho máximo de um *blob* detectável
Fonte: Autoria Própria

d) Tamanho mínimo que um *blob* deve ocupar dentro da imagem para ser considerado uma detecção.

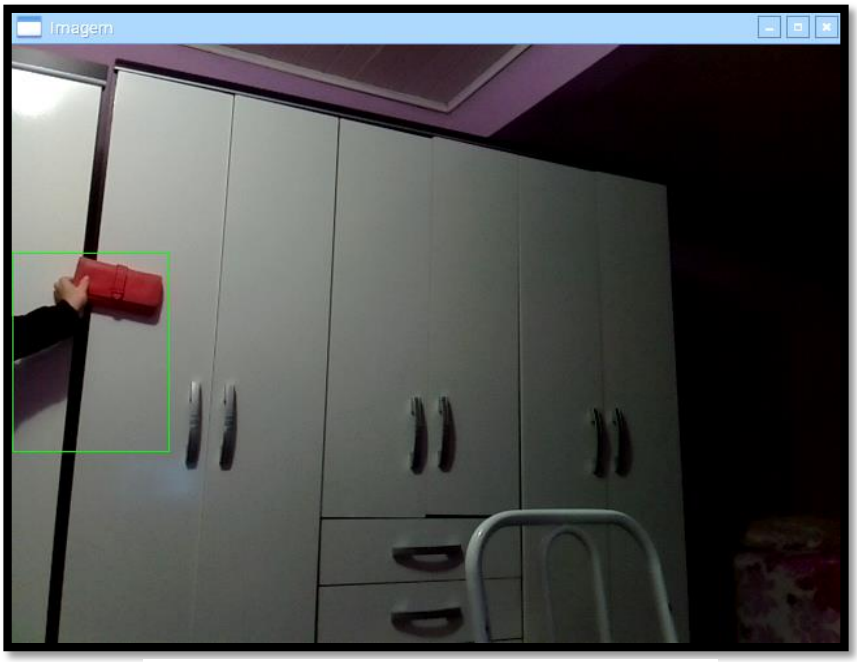

Figura 24 - Tamanho mínimo
Fonte: Autoria Própria

e) Detecção de sombras: sombras de objetos sólidos, cujo tons de cinza são mais escuros, não são detectadas. Na Figura 25, a sombra foi produzida por um objeto sólido não transparente.

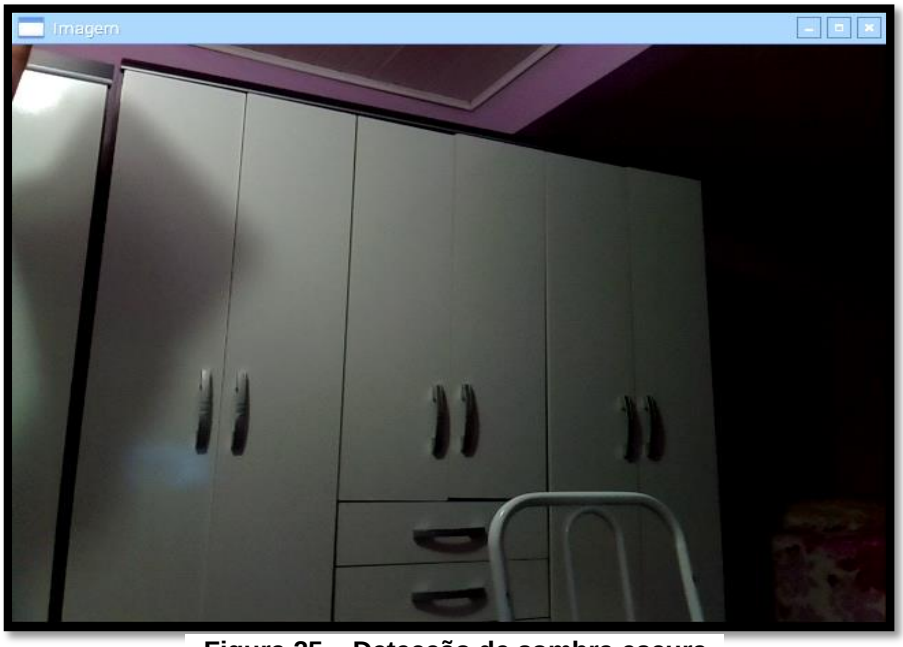

Figura 25 – Detecção de sombra escura
Fonte: Autoria Própria

f) Detecção de sombras: sombras cujos tons de cinza sejam mais claros, podem ocasionar falsa detecção. Na Figura 26, a sombra foi produzida por um objeto semitransparente.

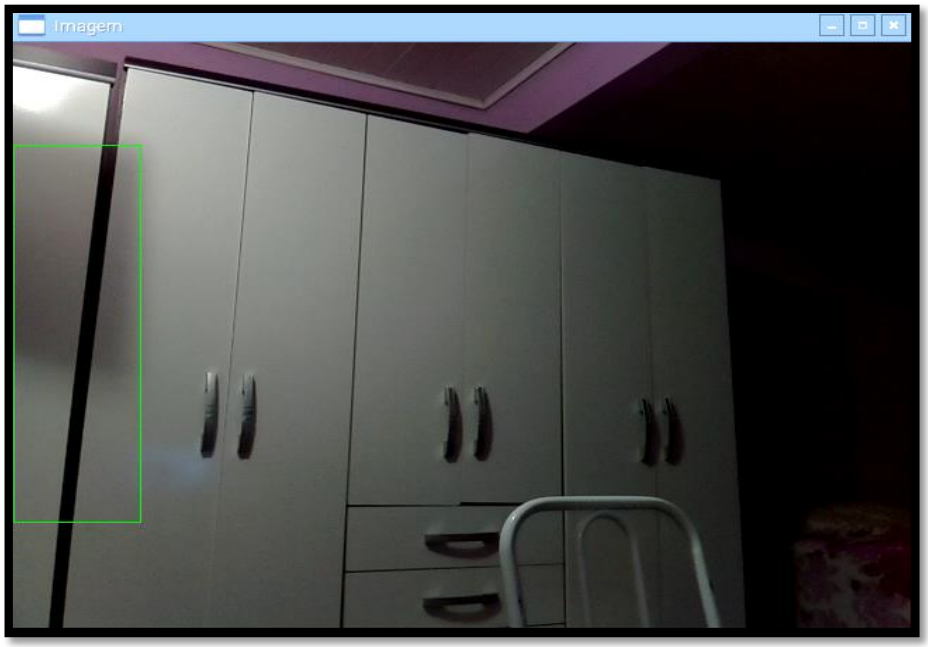

Figura 26 - Detecção de sombra clara
Fonte: Autoria Própria

g) Erro de detecção causado pelo balanço automático de branco da câmera, devido a uma modificação na iluminação do cenário. A detecção falsa desapareceu após aproximadamente 3 segundos.

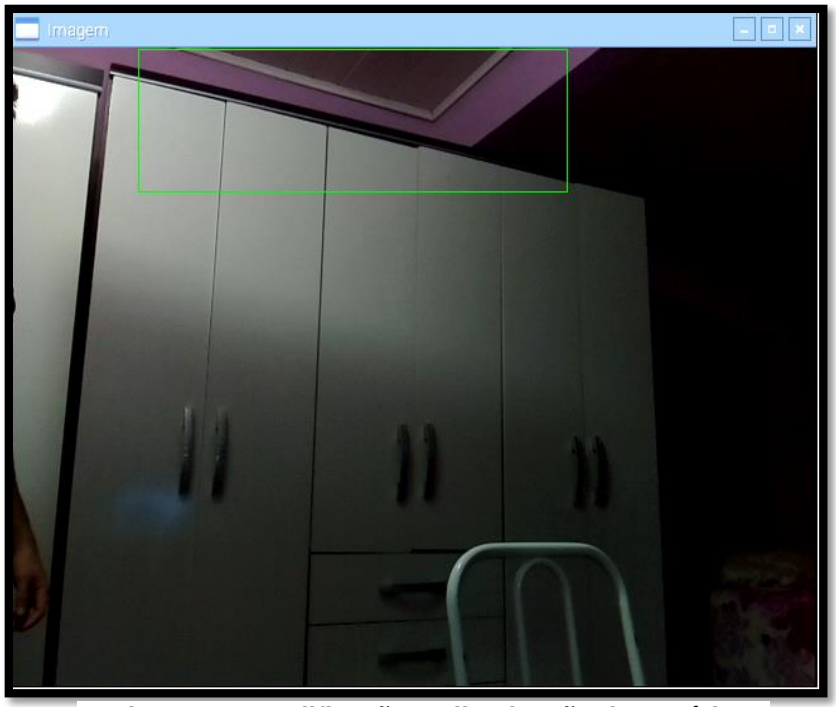

Figura 27 - Modificação na iluminação do cenário
Fonte: Autoria Própria

h) Detecção bem-sucedida.

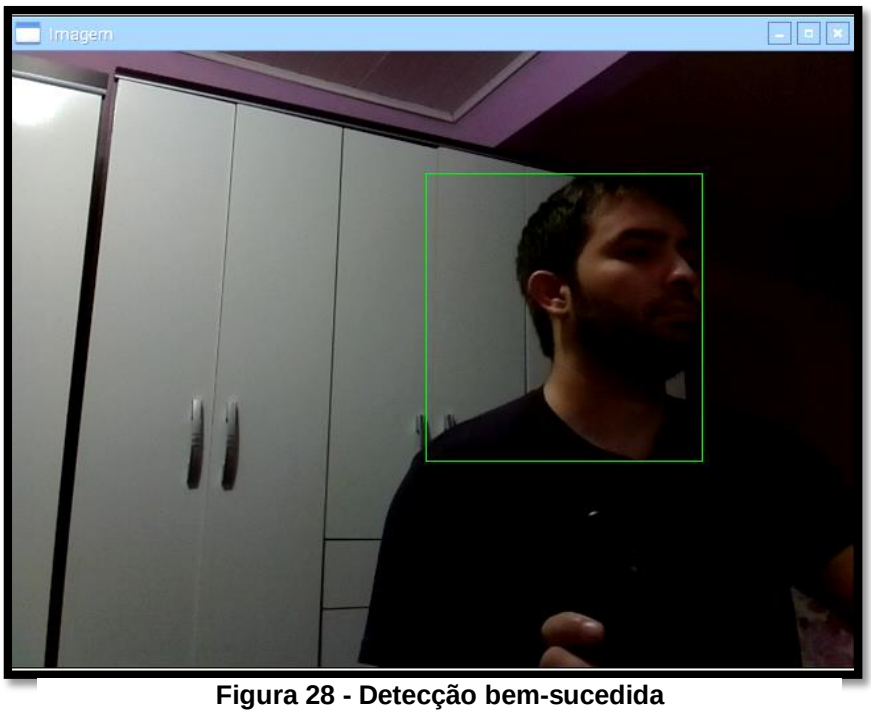

Figura 28 - Detecção bem-sucedida
Fonte: Autoria Própria

i) Detecção anulada pela mudança na iluminação da cena.

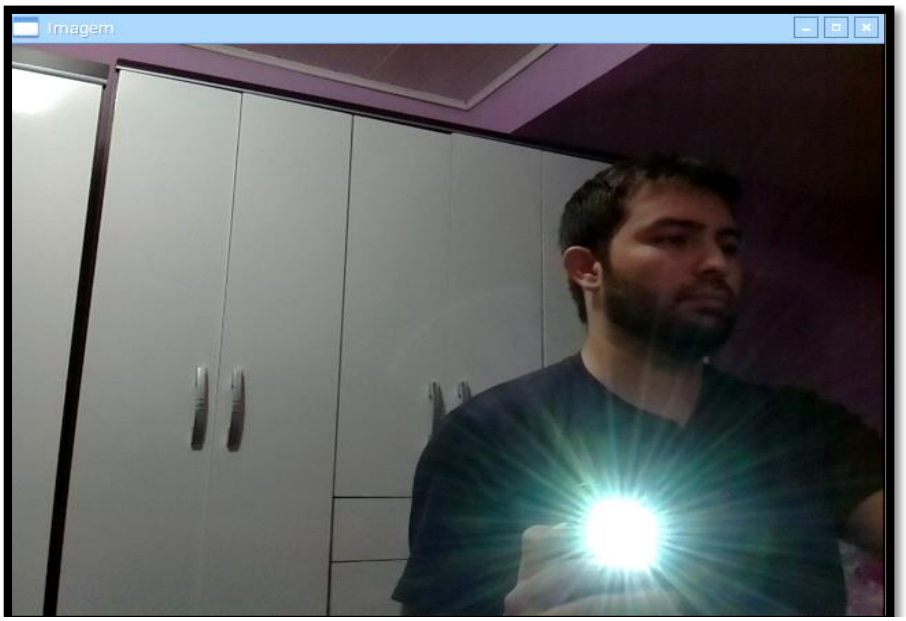

Figura 29 - Detecção anulada
Fonte: Autoria Própria

j) Detecção normal.

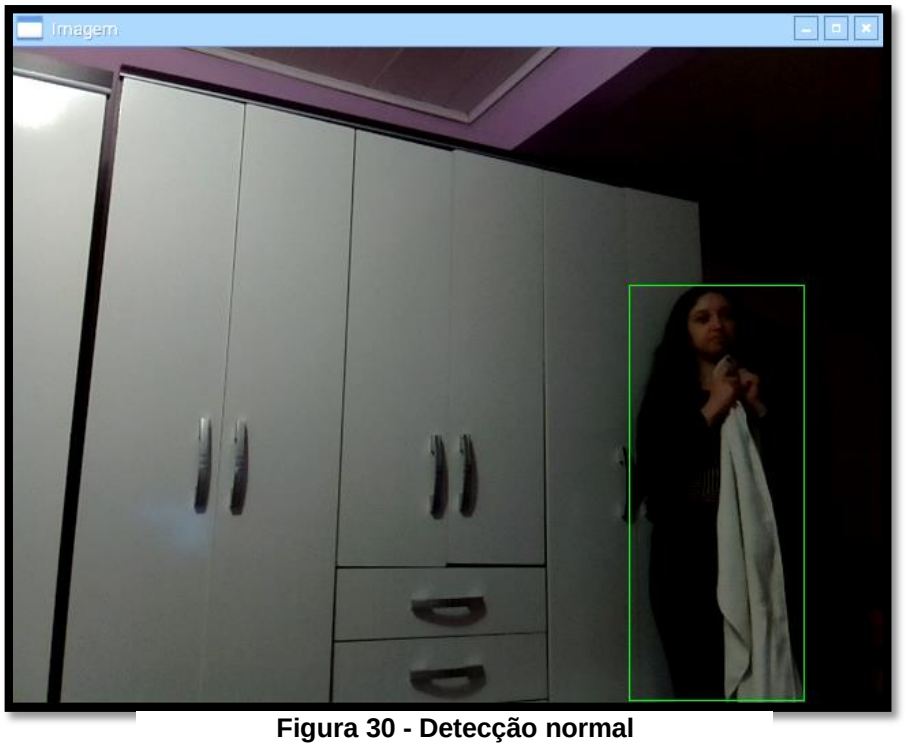

Figura 30 - Detecção normal
Fonte: Autoria Própria

k) Falha na detecção de cores similares ao plano de fundo.

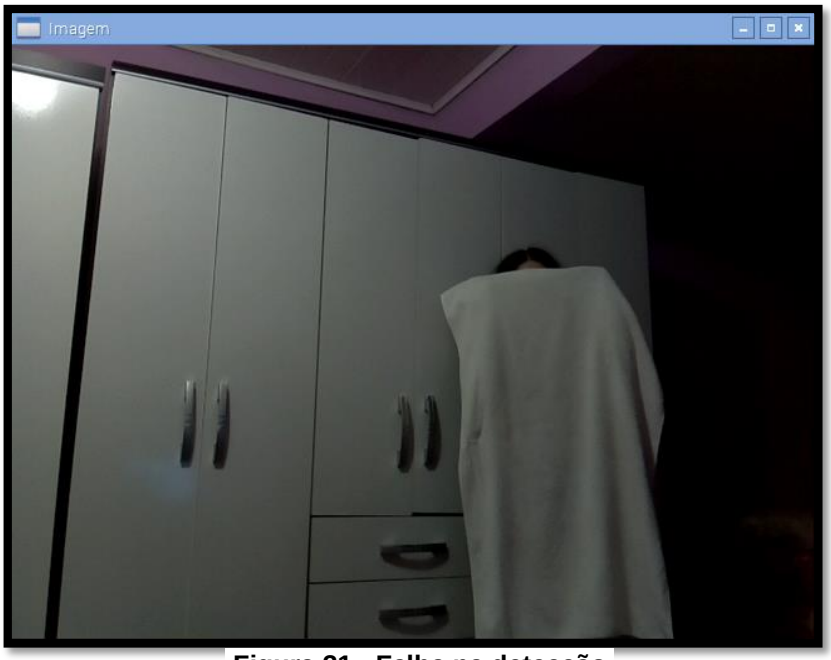

Figura 31 - Falha na detecção
Fonte: Autoria Própria

l) Pessoas próximas situadas a grande profundidade na imagem sendo detectadas como um único *blob*.

Figura 32 – Detecção de um único blob
Fonte: Autoria Própria

m) Após aproximadamente 6 minutos sem que houvesse movimentos consideráveis, as pessoas anteriormente detectadas se tornaram parte do plano de fundo.

Figura 33 – Atualização do plano de fundo
Fonte: Autoria Própria

n) Detecção múltipla. Ao ser movimentada, a porta deixou de fazer parte do plano de fundo gerando uma detecção.

Figura 34 - Detecção múltipla.
Fonte: Autoria Própria

o) Abertura de uma porta externa, alterando a iluminação do cenário, ocasionando falsas detecções. As detecções falsas desaparecem em aproximadamente 15 segundos.

Figura 35 – Detecção falha por alteração na iluminação do ambiente
Fonte: Autoria Própria

p) Falha na detecção de indivíduo com roupas similares ao plano de fundo, bem como falsa detecção devido a mudança na iluminação do ambiente.

Figura 36 – Detecção falha
Fonte: Autoria Própria

q) Detecção bem-sucedida de indivíduo cuja cor das roupas se assemelhava às do plano de fundo, porém, com contorno em cores diferentes.

Figura 37 - Detecção bem-sucedida
Fonte: Autoria Própria

r) Detecção múltipla, onde *blobs* próximos são considerados um único elemento

Figura 38 - Detecção múltipla
Fonte: Autoria Própria

s) Teste da versão sem *GUI* sem detecção.

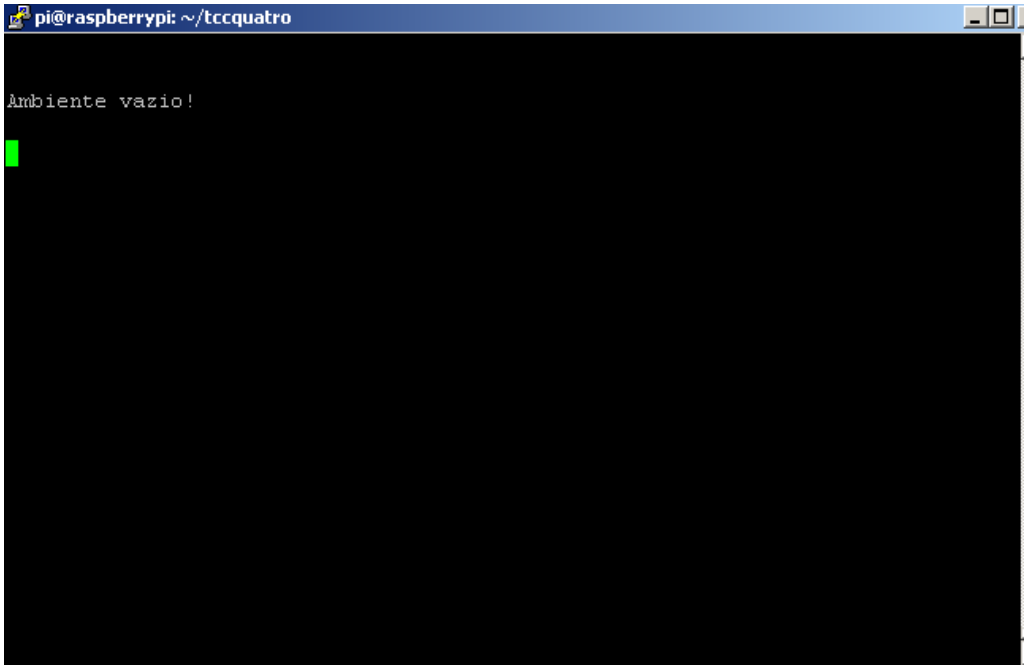

Figura 39 - Teste da versão sem *GUI* sem detecção
Fonte: Autoria Própria

t) Teste da versão sem *GUI* com detecção.

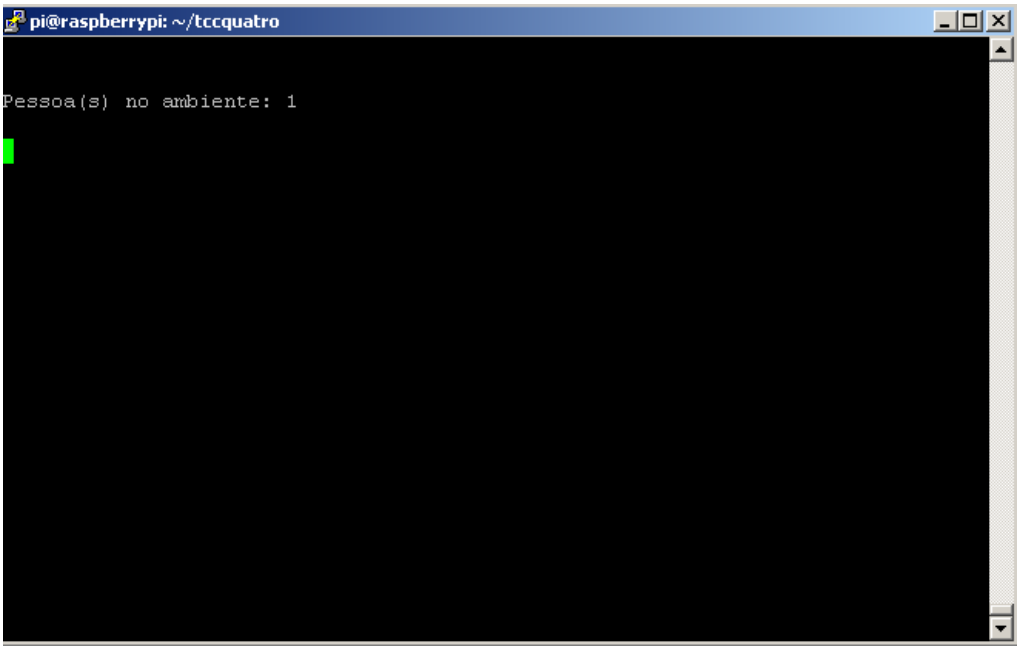

Figura 40 - Teste da versão sem *GUI* com detecção

Fonte: Autoria Própria

Os testes foram efetuados em ambientes internos. Nos primeiros testes realizados com pouca iluminação, poucas detecções ocorreram. Foram então ajustados os parâmetros do nível de *threshold* aplicado à imagem, para que o sistema detectasse objetos com tons mais escuros de cinza.

Após as devidas modificações, foi observada significativa melhoria na capacidade do sistema, porém, o sistema se tornou mais sensível às mudanças na iluminação do ambiente, fazendo com que aumentasse o número de falsas detecções.

Em ambientes amplos, indivíduos situados em maiores profundidades dentro da imagem não foram detectados, se fazendo assim necessária a modificação nos parâmetros, para que o sistema detectasse *blobs* menores. Feitos os devidos ajustes, o sistema apresentou maior precisão na realização das detecções.

Porém, quando as pessoas estavam em pontos muito distantes ou escuros da imagem, bem como quando usavam roupas similares às cores do plano de fundo, o sistema não foi capaz de detectá-las.

5 CONCLUSÃO E TRABALHOS FUTUROS

A ideia defendida neste trabalho considera que é possível projetar um sistema capaz um ambiente, utilizando câmeras presentes no mesmo, a fim de determinar quando há ou não atividade humana, para que, futuramente, seja possível controlar a distribuição de energia elétrica em ambientes monitorados com base nas informações fornecidas pelo sistema. A meta e objetivos foram atingidos em sua totalidade. A seguir serão apresentadas as conclusões referentes a cada objetivo proposto.

5.1 CONCLUSÃO

Quanto ao primeiro objetivo, foi estabelecida a fundamentação teórica, bem como a revisão bibliográfica referente a cada componente e processo necessários ao funcionamento do sistema. O sistema desenvolvido foi capaz de processar as imagens obtidas. Foi também capaz de detectar com precisão próxima de cem por cento a presença de pessoas isoladas nas imagens processadas, quando as condições iluminação eram ideais. Porém, apresentou erros de detecção diante de mudanças na iluminação do ambiente, bem como erros na contagem de indivíduos detectados. Quando muito próximos, dois ou mais indivíduos foram contabilizados como um só. E quando muito próximos à câmera, foram descartados das detecções. Falsas detecções ocorreram diante de mudanças na iluminação do ambiente. Em alguns testes, sombras foram detectadas, e pessoas com roupas cujas cores estavam muito próximas às cores do plano de fundo ou se encontravam muito distantes da câmera, se tornaram invisíveis ao sistema.

Quanto ao segundo objetivo, o sistema foi executado *on the fly* com sucesso no *Raspberry Pi*. Em relação ao desempenho, algumas observações devem ser efetuadas.

A taxa de aprendizado do plano de fundo pode ser ajustada, para que o mesmo seja atualizado com maior ou menor velocidade. Quanto menor a taxa de atualização, mais preciso é o modelo de plano de fundo, bem como maior é o tempo necessário para que objetos adicionados ao cenário se tornem parte do plano de fundo. Com menor taxa de aprendizado, observou-se ainda aumento no tempo necessário para que o sistema comece a efetuar detecções de maneira efetiva. Na versão final do sistema, foi utilizada uma taxa de atualização tal que, um objeto

estático na imagem levasse cerca de 5 minutos e 50 segundos para se tornar parte do plano de fundo. Observou-se que, quando iniciado, o tempo necessário para que o sistema comece a efetuar detecções de forma efetiva é similar ao que os objetos estáticos levam para se tornarem parte do plano de fundo.

Testou-se ainda o desempenho do sistema sem GUI, ou seja, sem exibição das imagens, apenas captando-as e extraindo as informações relevantes. Observou-se com isso, que o tempo para que o sistema começasse a efetuar detecções de forma efetiva, com os mesmos parâmetros anteriormente utilizados, sofreu uma queda de cerca de 50 segundos, o que significa uma melhoria de aproximadamente de 15% no desempenho do sistema.

Efetuaram-se testes modificando os parâmetros de atualização do plano de fundo, onde foram obtidos resultados similares.

5.2 TRABALHOS FUTUROS

Diante das conclusões alcançadas mediante os resultados dos testes, o primeiro passo na continuação do desenvolvimento do projeto, é testar o sistema desenvolvido com o maior número de câmeras possíveis, a fim de verificar quais trechos de código devem ser alterados, quais são as especificações mínimas que uma câmera deve possuir para atender aos requisitos do sistema, bem como quais câmeras são compatíveis com o mesmo.

Quanto ao funcionamento do sistema, sugere-se a implementação de técnicas mais elaboradas de detecção, a fim de diminuir o número de falsas detecções, bem como de problemas como a detecção de objetos diferentes como se fossem um só.

Sugere-se ainda a adaptação do sistema à versão 3.1 da biblioteca OpenCV, lançada quando o desenvolvimento do projeto já se encontrava em curso. Suas funcionalidades apresentam diversas melhorias e correções em relação à versão utilizada no projeto.

Quanto ao desempenho, sugere-se a implementação do sistema na versão 3.0 do Raspberry Pi, cujo *hardware* apresenta significativa melhoria, apresentando o triplo do desempenho médio em comparação com a versão B+ utilizada no projeto, conforme testes disponíveis no site do fabricante (RASPBERRY PI 3). A versão 3,

conta ainda com conectividade WI-FI nativa, o que expande os horizontes do projeto, permitindo a utilização de *Câmeras IP*[12].

E por fim, integrar ao sistema um dispositivo de controle da distribuição elétrica, como um conjunto de relês programáveis, por exemplo, para que o sistema tenha, de fato, acesso ao controle da distribuição elétrica nos ambientes monitorados.

[12] É uma câmera de vídeo que pode ser acessada e controlada por via de qualquer rede IP, como a LAN, Intranet ou Internet. Fonte: (WIKIPEDIA, 2015)

REFERÊNCIAS

ARAÚJO, Raphaela Cristina Andrade de; PEREIRA, Neuma Caroline Santos; PAULINO, Maria Do Socorro Moura; BARBALHO, Isabela Loreny Pierre; SILVA, Rosa Adeys. **A percepção dos funcionários de um complexo educacional sobre o consumo de energia elétrica no modo stand-by,** p. 5, 2014.

AUMONT, Jacques; MARIE, Michel. **ISBN 85-308-0703-0. Dicionário teórico e crítico de cinema**. Paris: Editora Papirus, p. 136-137, 2001.

BALLARD, Dana H; BROWN, Christopher M. **ISBN 0-13-165316-5. Computer Vision.** New Jersey: Editora Prentice Hall, p. 2, 1982.

BERTHOLDO, Flávio Augusto Rocha. **Técnicas de limiarização para melhorar a qualidade visual de documentos históricos**, p. 23, 2007.

BERTOLDI, Paolo; AEBISCHER, Bernard; EDLINGTON, Charles; HERSHBERG Craig; LEBOT, Benoit; LIN, Jiang; MARKER, Tony; MEIER, Alan; NAKAGAMI, Hidetoshi; SHIBATA, Yoshiaki; SIDERIUS, Hans Paul; WEBBER, Carrie. **Standby Power Use: How Big is the Problem? What Policies and Technical Solutions Can Address It?**, 2002.

Câmera. Disponível em: <https://www.raspberrypi.org/documentation/hardware/camera/README.md>. Acesso em 22 ago. 2015.

Câmera module. Disponível em: <https://www.raspberrypi.org/help/camera-module-setup>. Acesso em 10 dez. 2015.

CARVALHO, Antonio A. de; SILVA, Romeu R. da; REBELLO, João Marcos A.; VIANA, Alexandre F. **O Mundo das Imagens Digitais**, 2003

CASTRO, Rafael Barreto de; PEDRO, Rosa Maria Leite Ribeiro. **REDES DE VIGILÂNCIA: a experiência da segurança e da visibilidade articuladas às câmeras de monitoramento urbano,** 2009.

CUNHA, André Luiz Barbosa Nunes de. **Sistema automático para obtenção de parâmetros do tráfego veicular a partir de imagens de vídeo usando OpenCV;** 2013.

CUNHA, André Luiz Barbosa Nunes de; SETTI, José Reynaldo; GONZAGA, Adilson. **Comparação dos modelos de geração de background em processamento de vídeos de tráfego veicular,** p. 2-4, 2013.

DAVID, E.E.; SELFRIDGE, O.G.; TELEPHONE, Bell apud MILANO, Danilo de; HONORATO, Luciano Barrozo. **Visão Computacional,** p. 2, 1962.

Definição de Hardware. Disponível em: <https://www.oficinadanet.com.br/artigo/hardware/definicao_de_hardware>. Acesso em 05 mar. 2016.

DEPARMENT OF TRADE AND INDUSTRY – UNITED KINGDOM. **NR3 1BQ**. **The Energy Challenge,** p. 44; 2006.

Digitalização de Documentos esse é nosso serviço. Disponível em: <https://digitalizax.wordpress.com/>. Acesso em 05 mar. 2016.

FARIA, Diogo. **Análise e Processamento de Imagem.** p. 11. 2010.

GONZALEZ, Rafael C; WOODS, Richard E. **ISBN 0-201-18075-8**. **Digital Image Processing,** 2º Ed. New Jersey: Editora Prentice Hall, p. 119, 2010.

Hackerboards Raspberry Pi clone beefs up CPU, adds SATA. Disponível em: <http://hackerboards.com/sbc-mimics-raspberry-pi-has-faster-cpu-adds-sata/>. Acesso em 12 ago. 2015.

JEPSON, A.D.; FLEET, D.J. **Image Segmentation,** 2007.

MARQUES FILHO, Ogê; VIEIRA NETO, Hugo. **ISBN 8574520098. Processamento Digital de Imagens,** Rio de Janeiro: Brasport, cap. 1. p. 2. cap. 5, p. 139-143, 1999.

NEVES, Luiz Antônio Pereira; VIEIRA NETO, Hugo; GONZAGA, Adilson. **SBN: 978-85-64619-09-8. Avanços em Visão Computacional,** Curitiba: Omnipax Editora, p. 1, 2012.

I/O - Parte 1 - O que é? Disponível em: <https://www.oficinadanet.com.br/artigo/335/io_-_parte_1_-_o_que_e>. Acesso em 1 set. 2015.

O que é o projeto Debian? Disponível em: <https://www.debian.org/doc/manuals/project-history/ch-intro.pt.html>. Acesso em 19 mar. 2016.

O que são Blobs binários e porque você deveria conhecer. Disponível em: <http://www.sistemasembarcados.org/blobs-binarios.html>. Acesso em 02 abr. 2016.

OLIVEIRA JUNIOR, Luciano Lucas de. **Filtros Compostos e Adaptativos: o filtro de Gaussiano, Laplaciano do Gaussiano e do Gabor (Harmônico-Gaussiano),** p. 3-4, 2006.

OpenCV About. Disponível em: <http://opencv.org/about.html>. Acesso em: 1 set. 2015.

OpenCV Eroding and Dilating. Disponível em: <http://docs.opencv.org/2.4/doc/tutorials/imgproc/erosion_dilatation/erosion_dilatation.html>. Acesso em 21 mar. 2016.

OpenCV How to Use Background Subtraction Methods. Disponível em: <http://docs.opencv.org/3.1.0/d1/dc5/tutorial_background_subtraction.html#gsc.tab=0>. Acesso em: 06 fev. 2016.

OpenCV Reading And Writing Video. Disponível em: <http://docs.opencv.org/3.0-beta/modules/videoio/doc/reading_and_writing_video.html>. Acesso em 12 dez. 2015.

Raspberry About Us. Disponível em: <https://www.raspberrypi.org/about>. Acesso em 12 ago. 2015.

Raspberry What you will need. Disponível em <https://www.raspberrypi.org/help/quick-start-guide/>. Acesso em 05 dez. 2015.

REYNOLDS, Douglas. **Gaussian Mixture Models,** p. 1, 1992.

RPi USB Webcams. Disponível em: <http://elinux.org/RPi_USB_Webcams>. Acesso em 22 ago. 2015.

SILVA FILHO, Antonio Mendes da. **O Consumo de Energia no Modo Standby**, 2008.

SIQUEIRA, Alexandre Fioravante de. **Estudos de imagens provenientes de membranas de borracha natural com aditivos metálicos utilizando técnicas de Fourier, Gabor e Wavelets,** p. 10, 2011.

SILVA FILHO, Sidnei da; DREWS JR, Paulo; MARCOLINO, Luiz F. V. **Mistura de gaussianas: uma abordagem rápida para modelar nuvem de pontos,** p. 2, 2013.

SILVA, Evandro A; GONZAGA, Adilson. **Detecção de veículos em movimento usando modelo de misturas gaussianas e rnas**, p.2, 2006.

SONAGLIO, Simara; MOECKE, **Marcos. Detecção de movimento para sistema automática de vigilância por vídeo,** 2009.

TRAFTON, Anne. **In the blink of an eye**. 2014**.**

VITORINO, Prof. José Carlos. **Filtros de sinais**, p. 2-3.

Wikipédia Gaussian blur. Disponível em: <https://en.wikipedia.org/wiki/Gaussian_blur#cite_note-NixonAguado-2>. Acesso em 05 mar. 2016.

Wikipédia Thresholding (image processing). Disponível em: <https://en.wikipedia.org/wiki/Thresholding_%28image_processing%29>. Acesso em 12 mar. 2016.