Luiz Eduardo Pita Mercês Almeida

Relatório de Estágio Supervisionado Realizado na equipe de Visão Computacional do CPqD

João Pessoa

2018

Luiz Eduardo Pita Mercês Almeida

Relatório de Estágio Supervisionado Realizado na equipe de Visão Computacional do CPqD

Relatório de Estágio Supervisionado submetido ao Departamento de Engenharia Elétrica da Universidade Federal da Paraíba como parte dos requisitos necessários para a obtenção do título de Engenheiro Eletricista.

Universidade Federal da Paraíba

Centro de Energias Alternativas e Renováveis

Curso de Graduação em Engenharia Elétrica

Orientador: Dr. Alexsandro José Virgínio dos Santos

João Pessoa

2018

Luiz Eduardo Pita Mercês Almeida

Relatório de Estágio Supervisionado Realizado na equipe de Visão Computacional do CPqD

Relatório de Estágio Supervisionado submetido ao Departamento de Engenharia Elétrica da Universidade Federal da Paraíba como parte dos requisitos necessários para a obtenção do título de Engenheiro Eletricista.

Trabalho aprovado. João Pessoa, 07 de junho de 2018:

Dr. Alexsandro José Virgínio dos Santos
Orientador

Guilherme Adriano Fôlego
Supervisor CPqD

Dr. Euler Cássio Tavares de Macedo
Professor Convidado

Dr. José Maurício Ramos de Souza Neto
Professor Convidado

João Pessoa
2018

Este trabalho é dedicado a Mércia, Gonzaga, Rafaela, Maria do Carmo (Didi) e a Taisa, meus maiores motivadores e aqueles que sempre me cuidam.

Agradecimentos

Agradeço em primeiro lugar a Deus, sem o qual nada disso seria possível.

Em seguida a minha família, especialmente aos meus pais, que sempre me incentivaram e me apoiaram em minhas escolhas, dando-me forças para continuar mesmo com as barreiras enfrentadas, e à Taisa Del Pino, quem me apoiou nos momentos mais difíceis.

A Geraldo e Silvana Scabelo por terem me recebido e acolhido tão bem na cidade de Campinas.

Ao Prof. Dr. Alexsandro José Virgínio dos Santos, pela orientação, por todo o apoio antes e após o início do estágio, pelo acompanhamento, mesmo à distância, das atividades de estágio e pela contribuição para a realização deste relatório.

A Guilherme Adriano Fôlego, pela supervisão das atividades de estágio, pelo acompanhamento contínuo e disponibilidade em ajudar sempre que preciso. A Filipe Costa pelo suporte e parceria na execução das atividades do grupo de Visão Computacional e a todos os demais colegas do projeto apresentado nesse relatório, Bruno Costa, Adilson Batistel, Ricardo Shiguemi, Filipe Palma, Carlos Daniel, Pedro Bart e Thiago Gallante.

A Norberto Alves, gerente Tecnologias de Fala, Imagem e Mobilidade da Diretoria de Suporte à Decisão e Aplicações (DSDA), pelo acompanhamento e apoio em todos os momentos do estágio e em todas as atividades, coordenando e dirigindo minhas ações.

Aos demais colegas da plataforma de Computação Cognitiva, pelos ensinamentos, em especial ao líder da plataforma Alan Godoy Souza Mello.

A Marcos Alberto da Costa e aos professores Dr. Rogério Gaspar de Almeida, Dr. Helon David de Macêdo Braz e Dr. José Maurício Ramos de Souza Neto, pela prestatividade e proatividade durante e após o processo seletivo de estágio.

Aos professores Dr. Euler Cássio Tavares de Macêdo e Dr. Nady Rocha que me acompanharam ao longo dos anos na graduação como tutores do grupo PET Elétrica.

E a todos aqueles que de alguma forma, direta ou indiretamente, contribuíram para a realização deste estágio.

"..., estejam unidos em amor e alcancem toda a riqueza do pleno entendimento,
a fim de conhecerem plenamente o mistério de Deus, a saber, Cristo.
Nele estão escondidos todos os tesouros da sabedoria e do conhecimento.
(Bíblia Sagrada, Colossenses 2, 2-3)

Resumo

Este relatório descreve as principais atividades de estágio desempenhadas na Fundação Centro de Pesquisa e Desenvolvimento em Telecomunicações (CPqD), no período de 08/03/2018 a 06/06/2018, mais especificamente na Equipe de Visão Computacional da Plataforma de Computação Cognitiva. Dentre os trabalhos desenvolvidos neste setor, o estagiário contribuiu no projeto de inspeção visual de notebooks e teve participação em oficinas internas sobre ferramentas de softwares. As principais atribuições dadas a ele foram realizar experimentações em linguagem *Python* de algoritmos de Visão Computacional e auxiliar no desenvolvimento de biblioteca em C++ contendo os algoritmos avaliados. Para realização dessas atividades foram utilizados softwares como o *Visual Studio Code*, *CMAKE*, *GIT*, *LabView*, *Visual Studio*, ferramentas de programação em *Python*, biblioteca de visão *OpenCV* (*Open Source Computer Vision Library*), *ANDROID Studio* e *Eclipse*. Os resultados da experimentações foram aproveitados no desenvolvimento da biblioteca, que por sua vez, foi compilada e testada com sucesso pelo estagiário em seu ambiente final, isto é, no *LabView*.

Palavras-chave: visão computacional. biblioteca de visão. inspeção visual de notebooks.

Abstract

This report describes the main internship activities developed at CPqD, from 08/03/2018 to 06/06/2018, specifically in the Computer Vision Team of the Cognitive Computing Platform. Among the projects developed by this plataform, the intern helped in the project of visual inspection in notebooks and participated in internals workshops of software's tools. The main tasks assigned to him were to make experiments in Python language of some algorithms of Computational Vision and aid in the development of a C ++ library that contains the algorithms evaluated by the experiments. In order to perform these activities, some softwares were used, such as Visual Studio code, CMAKE, GIT, LabView, Visual Studio, programming tools in Python, OpenCV vision library (*Open Source Computer Vision Library*), ANDROID Studio and Eclipse. The experiments' results were aplied in the developed library, which was run and tested in the final environment, the LabView, by the intern.

Keywords: computational vision. vision library. visual inspection of notebooks.

Lista de ilustrações

Sumário

1 Introdução

Com o avanço das tecnologias digitais e o aumento da capacidade de processamento dos computadores, o sonho da inteligência autônoma tem se aproximado cada vez mais real. A busca por esse sonho inicia-se no contexto dos computadores digitais por Turing, por volta de 1950 e é seguido dos trabalhos em Inteligência Artificial (IA) com objetivo de dotar os computadores com capacidade de processamento de informações similar ao dos organismos biológicos (BALLARD; BROWN, 1982).

Essa busca levou ao desenvolvimento de máquinas capazes de lidar com entradas sensoriais, principalmente sons e imagens. A fim de imitar a visão, surgiram novas áreas na ciência para processar as chamadas imagens digitais, entre elas o Processamento Digital de Imagens (PDI) e posteriormente a Visão Computacional.

A Visão Computacional é o estudo da extração de informações de uma imagem, ou seja, é a descrição explícita de objetos físicos a partir de imagens. Enquanto o PDI realiza o processamento das imagens através de transformações da imagem em outra imagem, a Visão Computacional é o entendimento da imagem, ou seja, se utiliza do PDI como pré-requisito para o reconhecimento, a manipulação e a tomada de decisão sobre objetos em uma imagem (BALLARD; BROWN, 1982).

Nos últimos vinte anos, esse conhecimento tem adquirido maior usabilidade devido ao barateamento das câmeras e aumento da capacidade de processamento dos computadores. Como por exemplo, no campo da inteligência artificial, principalmente nas áreas de aprendizado de máquina e de reconhecimento de padrões, ele é muito empregado para o entendimento da informação gerada a partir de imagens. Outros campos de muita aplicabilidade dessa ciência são na medicina, nas ciências forenses, segurança, tecnologias assistivas, robótica, automação, entre outras (RIOS, 2010).

O estágio realizado na Fundação CPqD (Centro de Pesquisa e Desenvolvimento em Telecomunicações) abordou primordialmente essa área do conhecimento. Devido a sua aplicabilidade e relevância para os dias atuais, a Visão Computacional tem sido recorrente nos produtos, soluções, projetos e pesquisas desenvolvidas no CPqD.

Um desses projetos, foi o desenvolvimento de uma máquina para inspeção automática de notebooks. Esse processo é feito na linha de produção da montadora por operadores humanos os quais procuram defeitos como: teclas erradas, parafusos faltando, presença de *gaps* e ausência de etiquetas. Deviso a isso torna-se um processo minucioso ocorrendo muitas falhas devido a quantidade de defeitos que podem existir e a demanda de produção que deve ser comprida. Com isso, a montadora contratou o CPqD para pesquisar e desenvolver um protótipo que automatize o procedimento com o auxílio da

Visão Computacional.

A máquina desenvolvida consiste em uma câmara onde se coloca o notebook, fotografa-o de diferentes posições e as imagens geradas são comparadas com fotos previamente cadastradas de um notebook sem defeito. Se a diferença for maior que um determinado limiar, indica-se que esse notebook possui defeito.

As fotografias são tiradas por *smartphones* com sistema *ANDROID* a pedido da montadora, que os forneceu. Esses dispositivos ficam dentro da câmara em diferentes posições para fotografar as diferentes partes do notebook: frente, trás, lateral direita, lateral esquerda, superior e inferior. Essas fotos são transmitidas via *wi-fi* para um computador industrial acoplado a câmara. Ele, por sua vez, executa um programa em *LabView* o qual está integrado com uma biblioteca de Visão Computacional executando todo o procedimento de comparação das imagens. Além disso, o *LabView* fornece a interface gráfica para que o usuário possa interagir durante a etapa de cadastro dos notebooks sem defeitos. Esse projeto, principalmente o desenvolvimento da biblioteca de Visão Computacional, é de suma importância por se tratar do principal contexto das atividades efetuadas pelo estagiário.

2 A empresa

O estágio foi realizado na Fundação CPqD - Centro de Pesquisa e Desenvolvimento em Telecomunicações localizado no Parque II do Polo de Alta Tecnologia de Campinas no estado de São Paulo. O Parque Tecnológico foi uma iniciativa do CPqD e se denomina Pólis de Tecnologia, constando com diversas empresas desse setor.

Nesta seção, apresenta-se uma descrição sobre o CPqD e o Pólis de Tecnologia. São elencadas as áreas de atuação do CPqD, bem como as atividades desenvolvidas na Plataforma Tecnológica de Computação Cognitiva e na Gerência de Tecnologias em Processamento de Fala e Imagem e Mobilidade, setores no qual o estagiário foi alocado.

2.1 O CPqD

A Fundação CPqD é uma fundação de direitos privados e portanto uma instituição independente focada na inovação com base nas Tecnologias da Informação e Comunicação (TICs). O CPqD tem por missão contribuir para o desenvolvimento, o progresso e o bem-estar da sociedade brasileira através da inovação das TICs (CPQD, 2018e).

Atualmente o programa de pesquisa e desenvolvimento do CPqD é o maior da América Latina em sua área de atuação, englobando atividades voltadas para futuras transições tecnológicas, projetos direcionados às necessidades do mercado e utilização de tecnologias para o atendimento das políticas públicas de inclusão digital. Suas soluções abrangem diversos setores, como comunicação, multimídia, financeiro, indústrias, administração pública, defesa e segurança (CPQD, 2018b).

Fundado em 1976, o CPqD tem sua origem como o Centro de Pesquisa e Desenvolvimento da Telebrás, estatal que detinha o monopólio nacional dos serviços públicos de telecomunicações. Com a privatização da Telebrás em 1998, o CPqD passou a ser uma fundação de direito privado, ampliando seu setor de atuação para outros segmentos (CPQD, 2018e).

O CPqD tornou-se então em um ecossistema inovador e aberto, permeado por várias frentes colaborativas, que possibilitam a empresa gerar diversas soluções, como tecnologias de produtos, sistemas, consultorias, fornecimento de capital intelectual e pesquisas. Desse modo, ele passou a manter proximidade com *startups*, investidores, incubadoras, aceleradoras, universidades, institutos de pesquisa, grandes empresas, governos e todos os ambientes em que o desenvolvimento e a inovação são fundamentais (CPQD, 2018e). Além disso, passou a apresentar soluções para as mais diversas áreas: indústria, telecomunicações, agronegócios, sistemas financeiros, *utilities* (energia elétrica, gás, água,

saneamento e petróleo), varejo e serviços, e abrange temas de grande repercussão na atualidade como banda larga, *smart grids*, banco do futuro, cidades inteligentes, defesa e segurança, *Internet of Things* (IoT), segurança da informação, inteligência artificial, *blockchain* e responsabilidades socioambientais (CPQD, 2018c).

Para gerenciar e organizar as pesquisas nessas diversas áreas, o programa de P&D do CPqD é inspirado em grandes temas estruturantes centrados nas necessidades das pessoas e dos clientes (CPQD, 2018b). Assim, a base de conhecimento do CPqD é composta de plataformas tecnológicas, as quais são caracterizadas por uma coleção de competências tecnológicas afins, gerenciadas de forma sistemática, tendo em vista a inovação (pesquisa, desenvolvimento e aplicação no mercado) (CPQD, 2018e). As plataformas atualmente existentes na empresa são:

- Sistemas Eletrônicos Embarcados;
- Sistemas de Energia;
- Sensoriamento;
- Comunicações sem fio;
- Computação Avançada;
- Segurança da Informação e Comunicação;
- Redes de Dados;
- Comunicações Ópticas;
- Computação Cognitiva.

Os resultados tecnológicos oriundos das plataformas são transformados em resultados para o mercado, compondo ofertas de software e serviços, licenciando tecnologias para a indústria ou criando novas empresas (CPQD, 2018e). Para isso, o CPqD conta com o suporte de diversas instituições de fomento e empresas parceiras (CPQD, 2018b). Assim, hoje a empresa possui mais de 40 anos desenvolvendo soluções avançadas em TICs e em outras áreas, visando sempre contribuir com o desenvolvimento e o progresso tecnológico do Brasil e promovendo o bem-estar da sociedade (CPQD, 2018e).

2.2 O Pólis

Devido a expansão do número de empresas que atuam em mercados estratégicos de tecnologia como telecomunicações, informática e automação industrial, o CPqD criou

em 1999 o primeiro Centro Empresarial Tecnológico de Campinas, denominado Pólis de Tecnologia (CPQD, 2018d).

O Pólis oferece um gerenciamento eficiente e infraestrutura de serviços para as empresas que operam com alta tecnologia. Entre os serviços proporcionados citam-se serviços de restaurantes, lanchonetes, bancos, correios, reprografia, transporte fretado, anfiteatros, auditórios, clube, entre outros (CPQD, 2018d).

As empresas de tecnologia que hoje compõem o Pólis são: CPqD, Matera Systems, CI&T, Trópico, Dextra, Padtec, Já!, Dex Training, Exxemplo e Elabora. Além disso, há as empresas que prestam serviços como a CECORP, Localiza, Santander, HSBC, Itaú, Nono Miquele Restaurantes, Thiane, entre outras (PÓLIS DE TECNOLOGIA, 2018).

Algumas dessas empresas fazem parte do Universo CPqD, um universo de iniciativas de tecnologias brasileiras criadas pelo CPqD. Essas empresas são organizações com características distintas, criadas a partir da iniciativa direta ou indireta do CPqD, por meio da transferência das tecnologias de produtos geradas pelo CPqD. Elas assumem a responsabilidade pela produção e comercialização desses produtos, disseminando tecnologias inovadoras e produtos e serviços diferenciados de alto valor agregado. Algumas das empresas que fazem parte desse Universo são a Padtec, a Trópico, o Instituto Atlântico e a Já! (CPQD, 2018e).

2.3 A Plataforma de Computação Cognitiva

Essa plataforma é responsável por pesquisar e desenvolver soluções com o uso de Inteligência Artificial (IA). Portanto, define-se como uma plataforma inspirada nas capacidades únicas do cérebro humano de analisar e resolver problemas (CPQD, 2018a).

Ela desenvolve sistemas capazes de ajudar na tomada de decisões e de fazer previsões adequada. Para isso faz uso de sensores, modelos, metodologias, algoritmos e dados de modo a conseguir identificar padrões, reconhecer objetos, visualizar inúmeras possibilidades e validar hipóteses (CPQD, 2018a).

A plataforma divide-se em seis competências em que cada uma delas faz uso de técnicas específicas. São elas: Raciocínio, Aprendizado, Visão, Fala, Diálogo e Sinais. Por sua vez o estagiário atuou primordialmente na competência de Visão, ingressando na equipe de Visão Computacional da Diretoria de Suporte a Decisão e Aplicações (DSDA) onde está inserida a Gerência de Tecnologias em Processamento de Fala e Imagem e Mobilidade.

2.3.1 Competência de Visão

A competência de visão trata-se do conjunto de métodos e tecnologias por meio dos quais sistemas computacionais tornam-se capazes de interpretar imagens, buscando se assemelhar à visão humana. Desse modo, ela permite aos computadores processar imagens e vídeos, reconhecer padrões, extrair e organizar informações das imagens. Diversas técnicas são usadas por essa competência, a exemplo de técnicas para aquisição de imagens, filtragem, segmentação, representação e descrição, morfologia matemática e reconhecimento de objetos (CPQD, 2018a).

O universo de aplicações é bem diversificado. No CPqD existem casos de aplicações para biometria de face, reconhecimento de cenas e objetos (por exemplo gestos, ações e vigilância), reconhecimento de padrões (como emoções, caracteres e inspeção visual), vivacidade (*anti-spoofing*), visão assistiva, entre outros. Além disso, são grupos de interesse a análise forense, análise de imagens médicas e controle de processos (CPQD, 2018a).

3 Objetivos do Estágio

Nesta seção apresentam-se os objetivos gerais e específicos do estágio.

3.1 Objetivos Gerais

Os objetivos gerais do estágio, conforme definido no plano de trabalho, consistem na realização de duas atividades principais: Apoio às atividades de pesquisa e desenvolvimento na área de Inteligência Artificial, com foco em Visão Computacional e participações em *workshops* internos de tecnologias para IA. A primeira atividade, abrange a realização dos seguintes itens:

- rotular bases de imagens;
- realizar experimentos com *OpenCV* (*Open Source Computer Vision Library*);

3.2 Objetivos Específicos

Os objetivos específicos estabelecidos para as primeiras etapas do estágio e como meio de alcançar as metas gerais propostas são:

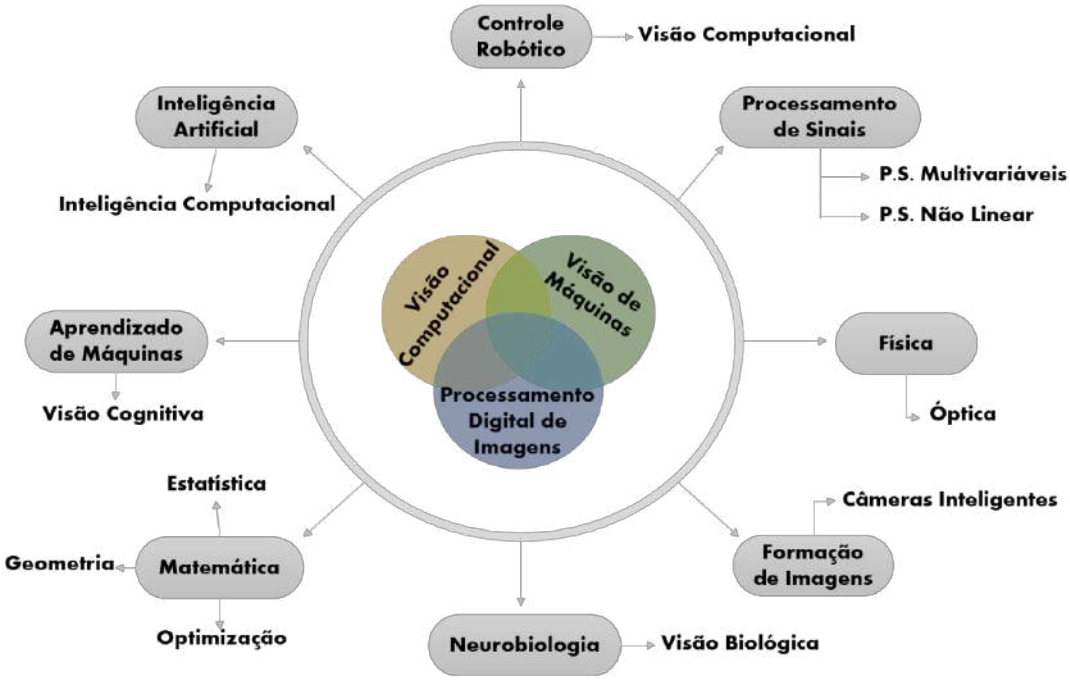
- programar experimentos em linguagem *Python* com *OpenCV* para projeto de inspeção visual de notebooks;
- compilar biblioteca dinâmica (*DLL*) de visão computacional com *OpenCV* em C e C++ para uso em projeto de inspeção visual de notebooks;
- e participar em oficinas de programação em *BASH*, *Docker* e em *JAVA Native Interface* (*JNI*).

4 Fundamentação Teórica

O conjunto de métodos e técnicas que tornam sistemas computacionais capazes de extrair e interpretar informações de imagens teve seus primeiros estudos mais aprofundados no final da década de setenta (BALLARD; BROWN, 1982). A Visão Computacional é fortemente influenciada pela compreensão dos processos de aquisição de imagens e de sua percepção no sistema visual do homem e de outros animais.

A visão computacional engloba e pode ser aplicada a diversas áreas do conhecimento: inteligência artificial (inteligência computacional), aprendizado de máquina (visão cognitiva), processamento de sinais, física (óptica), matemática, neurobiologia (visão biológica) e robótica (RIOS, 2010). Algumas dessas áreas podem ser visualizadas na Figura 1.

Figura 1 – Áreas relacionadas à Visão Computacional, ao PDI e à Visão de Máquinas.



Fonte: Adaptada de RIOS, L.R. S. Visão Computacional. 2010.

Tais aplicações podem fazer parte de sistemas complexos. São exemplos desses sistemas equipamentos para processamento de imagens médicas e aplicações industriais para controle de qualidade ou medição de características de produtos. Além delas, a Visão Computacional pode ser aplicada no reconhecimento de pessoas e padrões, segurança de ambientes, visão de robôs, aplicações militares (mísseis guiados, por exemplo) e várias outras atividades que dependem da visão biológica (RIOS, 2010).

Reconhecer algum objeto, padrão ou comportamento em imagens requer um conhecimento prévio sobre o mundo e capacidade de julgamento para se fazer conclusões e tomar uma decisão. Como exemplo, um sistema que atua no controle de qualidade de uma fábrica precisa ter conhecimento sobre o que é uma peça, as especificações dela, para saber então se é defeituosa ou não. Depois, caso seja defeituosa, decidir se vai para o descarte ou para o conserto. Outro exemplo, em uma linha de produção, uma máquina que inspeciona determinado produto precisa saber quais os defeitos que podem existir nesse produto e o quanto a presença dele é tolerável ou não, antes de decidir se aquele produto deve seguir na linha.

Entretanto, antes de tomar decisões sobre os objetos e dados de uma imagem é preciso extraí-los da imagem. A Visão Computacional faz uso das técnicas do Processamento Digital de Imagens (PDI) para remover ruídos, tratar configurações de iluminação e sombra, a noção de perspectiva, profundidade ou odometria e principalmente, isolar os objetos, elementos de interesse, de uma cena (RIOS, 2010). O PDI utiliza-se de pouco conhecimento da semântica ou conteúdo da imagem, ao passo que os métodos de alto nível correlacionados ao conceito de Visão Computacional envolvem as tarefas como segmentação da imagens em regiões ou objetos de interesse, suas descrições, reduzindo-os a uma forma mais apropriada para representar o conteúdo da imagem e reconhecimento ou classificação desses objetos. As principais características usadas para descrever-los incluem contornos ou bordas, suas dimensões, relacionamento com os outros objetos, cores e texturas (PEDRINI; SCHWARTZ, 2008).

Mesmo com os avanços recentes das tecnologias de captura e processamento de imagens e, principalmente, no surgimento de algoritmos com uso de redes neurais e outras técnicas de IA , não existe uma solução única para os problemas de Visão Computacional. O que existe são algumas definições gerais e soluções dispersas e específicas para um determinado problema, a exemplo da leitura de textos, biometria facial, *anti-spoofing*, reconhecimento de objetos, entre outros (RIOS, 2010). A seguir serão exemplificados algumas situações, técnicas, soluções e aplicações recorrentes em Visão Computacional, de modo a trazer a ideia do que se estuda atualmente nessa área.

4.1 Operações do Processamento Digital de Imagens

Nessa seção serão abordados três técnicas do PDI: detecção de bordas, análise de texturas e computação de fluxo óptico. Tais técnicas são chamadas de iniciais por serem realizadas no princípio das etapas de operação do PDI (RUSSELL; NORVIG, 2010).

Bordas são curvas ou linhas ao longo do plano da imagem em que há uma mudança significativa no brilho da imagem. A ideia por trás da detecção de bordas é simplificar e compactar as informações de uma imagem representada por diversos valores na escala de

cinza. Dessa forma, pode-se isolar apenas os contornos das cenas e dos objetos importantes de uma imagem (GONZALEZ; WOODS, 2006).

Por se tratarem de variações do brilho de uma imagem causadas por uma mudança na forma do conteúdo da imagem, as técnicas de detecção de bordas se baseiam no cálculo de derivadas, podendo utilizar os chamados filtros de aguçamento, há exemplo dos gradientes de *Sobel*, *Roberts* e *Prewitt* (QUEIROZ; GOMES, 2014). O filtro de *Canny* é uma das aplicações mais utilizadas para detecção de bordas, por associar filtros de aguçamento com filtros de suavização, garantindo o reconhecimento dos contornos com pouca interferência de ruídos (CANNY, 1986). Outra alternativa é dada uma imagem binária detectar os contornos das regiões existentes através do uso do Teorema de *Green*, para relacionar á área dessas regiões a curva fechada que as delimita.

Outro termo usado nessa área é a textura, a qual consiste de padrões sobre uma superfície espacialmente repetidos e que podem ser percebidos visualmente. Exemplos desse tipo são as janelas na fachada de um prédio, as listras de uma zebra, conchas na areia da praia, e outros tipos. Ao passo que a detecção de bordas leva em consideração o valor de brilho dos *pixels*, o conceito de textura considera o histograma das orientações de um conjunto de *pixels*. Como exemplo, a textura de tijolos em uma parede possuíra dois picos em seu histograma, um na vertical e outro na horizontal, por sua vez, pintas em um leopardo vão apresentar uma distribuição de orientação mais uniforme no histograma (RUSSELL; NORVIG, 2010).

A detecção de texturas são menos sensíveis a mudanças de iluminação. Tal característica torna essa técnica uma ferramenta importante na detecção de objetos, uma vez que, técnicas como a busca por contornos sofrem grande variação com esse alterações nesse fator (RUSSELL; NORVIG, 2010).

Uma das principais técnicas utilizadas para detecção de movimento em sequências de imagens é o fluxo óptico. O fluxo óptico descreve a velocidade e a direção do movimento de cada *pixel* em relação à imagem anterior (RIOS, 2010). O fluxo óptico pode conter informações úteis sobre o cenário, como exemplo, pode fornecer dados sobre a distância dos objetos, uma vez que objetos mais próximos se movem mais rapidamente que os mais distantes.

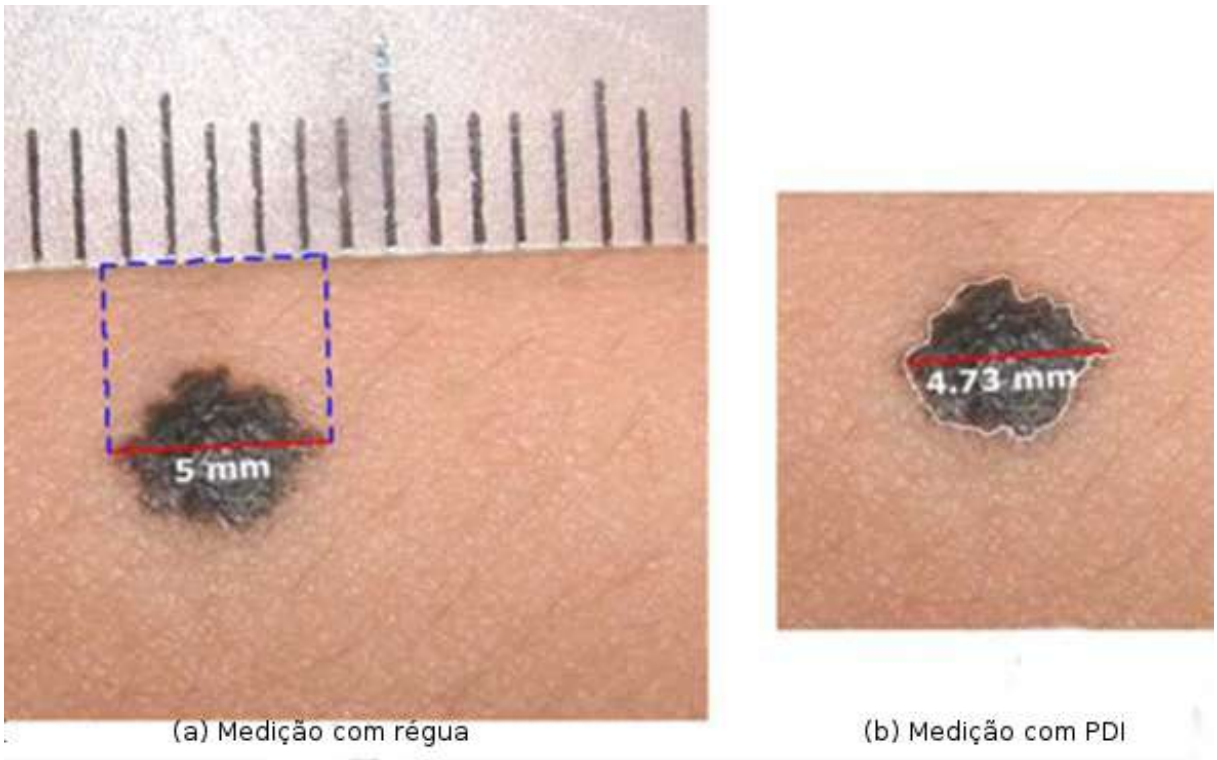
Além dessas técnicas, existe uma diversidade de algoritmos e formas de se reconhecer objetos e características em uma imagem. A grande diversidade de modos de atuação possibilita desenvolver diferentes soluções para um mesmo problema. A seguir serão exemplificados alguns dos desafios e aplicações recorrentes na Visão Computacional.

4.2 Aplicações

Os avanços no campo dos algoritmos cognitivos ao longo dos anos tornou diversas aplicações viáveis. Exemplos de áreas que utilizam a Visão Computacional e o PDI são inúmeras e abrangentes, podendo citar a medicina, a biologia, a automação industrial, sensoriamento remoto, astronomia, microscopia, artes, área militar, arqueologia, segurança, vigilância, agricultura, esportes, entre outras.

Na medicina pode auxiliar os diagnósticos médicos a partir das imagens capturadas de aparelhos de raio X, tomografias computadorizadas, ressonância magnética e ultra-sonografia. O uso de técnicas do PDI e da Visão Computacional facilitam por exemplo na identificação de regiões atingidas pelo câncer, permitindo ao médico fazer o diagnóstico com mais precisão e rapidez, podendo, assim, planejar melhor o tratamento (NEVES; NETO; GONZAGA, 2012). Como exemplo na Figura 2 tem-se uma imagem contendo uma lesão de pele no qual foi desenvolvido um método com auxílio do PDI para calcular o diâmetro da lesão. Em Figura 2-a observa-se a medição de um edema na pela com uso da régua e em Figura 2-b com o uso das técnicas estudadas.

Figura 2 – Exemplo com cálculo do diâmetro da lesão.



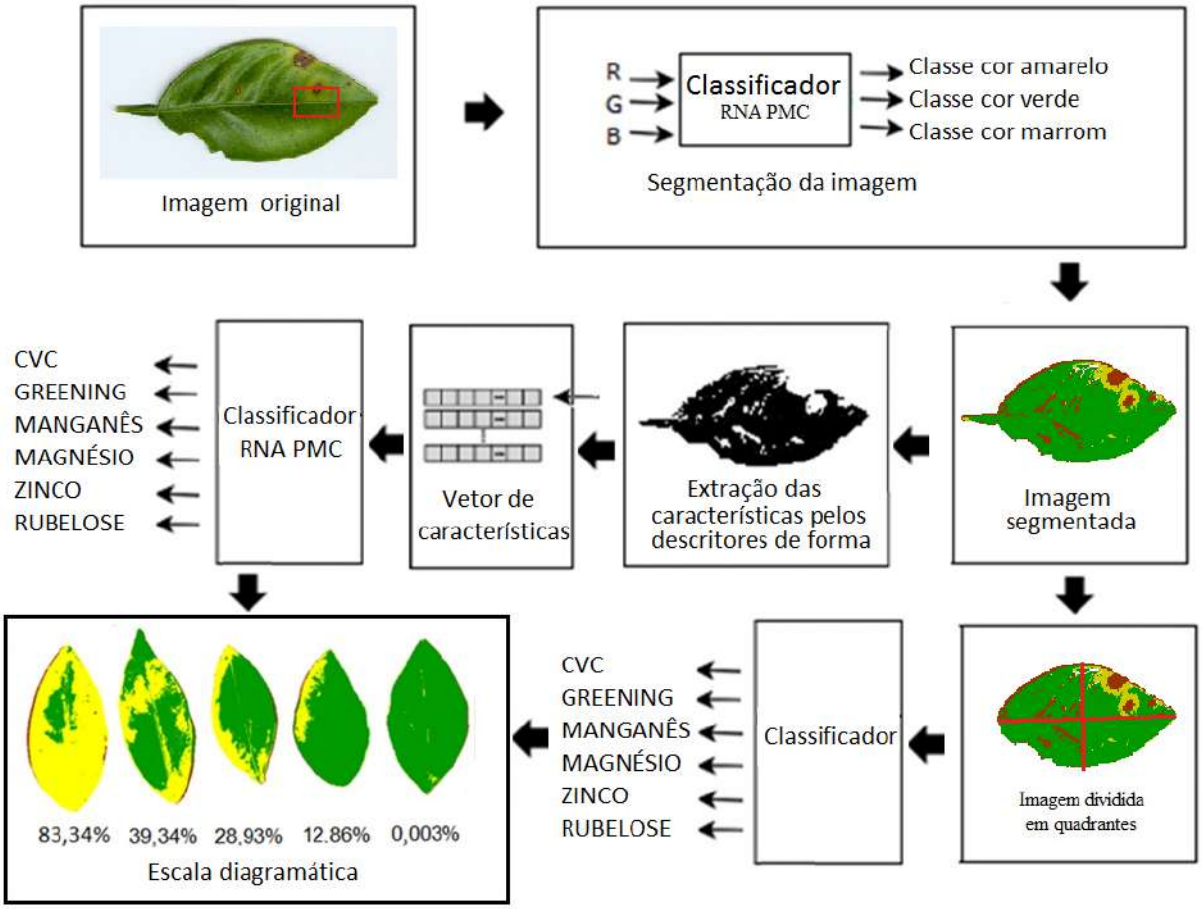
Fonte: Araujo et al. (2012).

Na indústria, pode-se utilizar a Visão Computacional em processos de montagem e inspeção de produtos, em visão robótica e no controle de qualidade. Algumas atividades comuns incluem a verificação de falhas em circuitos impressos, separação de peças por

robôs na linha de montagem, classificação de defeitos em soldas, entre outras (PEDRINI; SCHWARTZ, 2008).

Com a facilidade no acesso a imagens de satélites, o uso dos processos de manipulação e compreensão de imagens digitais nos setores de sensoriamento remoto, agricultura, pecuária, cidades inteligentes tem ganhado novas proporções. Podendo auxiliar no acompanhamento de áreas urbanas, previsão de fenômenos climáticos, monitoramento de áreas atingidas por erosão, previsão de safras, determinação de áreas de desmatamento, análise de composição de solo, extrações de feições cartográficas, análise de pragas (COSTA, 2017). Na Figura 3 visualiza-se a Visão Computacional na agricultura sendo usada para distinguir determinado tipo de doença em folhas das plantas de frutas cítricas, como as das laranjeiras. Outro exemplo é o projeto *Sunroof* da *Google*, que utiliza imagens de satélites para calcular a área de telhados, a insolação e sombreamento sobre eles a fim de estimar o custo e a capacidade energética de uma possível instalação de painéis fotovoltaicos nesses telhados (GOOGLE, 2018).

Figura 3 – Exemplo com identificador de doenças em plantas.

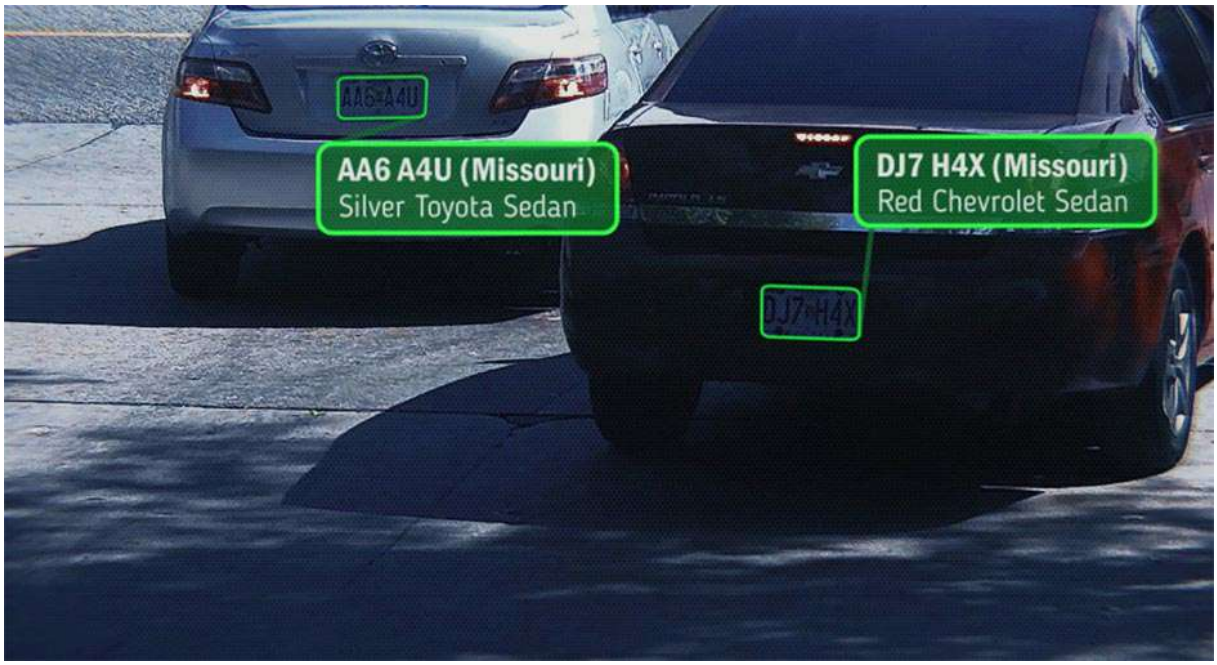


Fonte: Ribeiro, Paiva e Jorge (2014).

A Visão Computacional tem um dos seus maiores usos em sistemas biométricos.

A identificação de impressões digitais possibilita identificar digitais em imagens, garantir uma chave pessoal única que pode ser utilizado em sistemas bancários, cofres, registros, entre outros. O reconhecimento facial permite distinguir indivíduos a partir de imagens ou vídeos, auxiliando o reconhecimento de pessoas em fichas criminais, construção de retratos falados, sistemas de vigilância, além de poder ser utilizado como senha de acesso em dispositivos móveis, cofres e estabelecimentos seguros (BONATO; NETO, 2011). Uma aplicação do reconhecimento facial, detecção de movimento e de esqueleto é a identificação de emoções e comportamentos, que pode ser usada para tratar perfis psicológicos, como polígrafos em entrevistas e para atendimento personalizado com base nas emoções de um clientes.

Figura 4 – Aplicação de reconhecimento automático de placas veiculares e veículos.

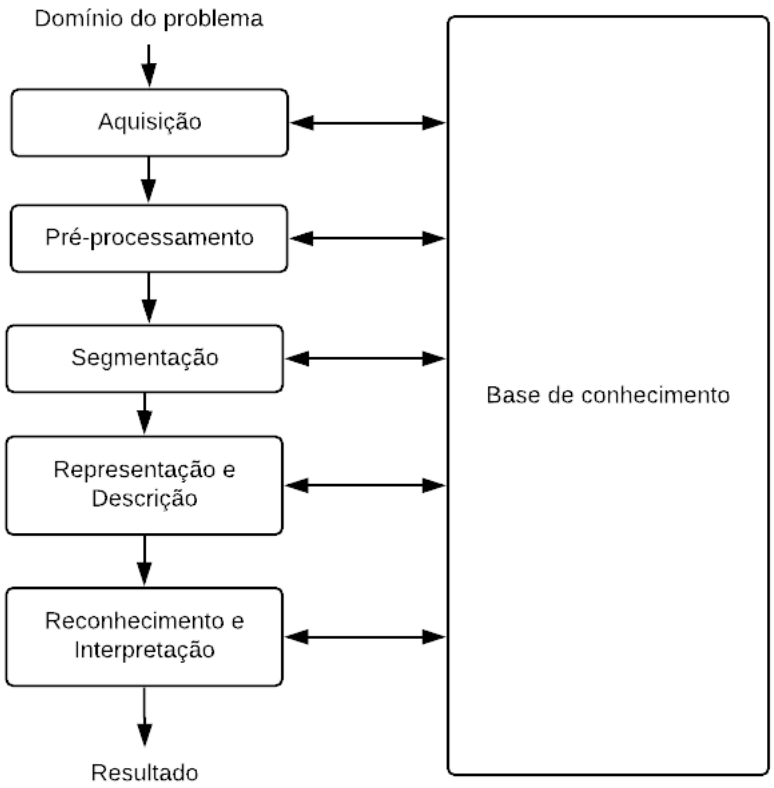


Fonte: OPENALPR (2018).

O Reconhecimento Óptico de Caracteres, mais conhecido por sua sigla em inglês OCR (*Optical Character Recognition*), permite reconhecer caracteres impressos, manuscritos ou datilografados de diferentes idiomas a partir de imagens de documentos, cheques, tela de computadores, entre outros. Isso abre possibilidade para diversas aplicações como digitalização de livros, documentos e relatórios de modo a possibilitar buscas por palavras nos arquivos digitalizados, para correções de provas escritas, aplicativos para cálculo automático de contas escritas no papel, leitura de etiquetas e rótulos, entre outras (NEVES; NETO; GONZAGA, 2012). O OCR é muito utilizados nos problemas de Reconhecimento de Placas de Veículos (*License Plate Recognition - LPR*), servindo para controle em estacionamentos, identificação de placas em radares eletrônicos e em câmeras de segurança, na Figura 4 observa-se essa aplicação do OCR. Outras aplicações são em aplicativos de

acessibilidade que fazem leitura de caracteres para deficientes visuais, como o *PayVoice* desenvolvido pelo CPqD, aplicativo para leitura do valor de máquinas POS (*Point of Sale*) de cartão de crédito (MARTINS, 2018).

Figura 5 – Etapas de um sistema de processamento de imagens.



Fonte: Pedrini e Schwartz (2008).

4.3 Etapas de um sistema de Visão Computacional

Os sistemas de Visão Computacional comumente seguem um conjunto de etapas, como ilustrado na Figura 5. As etapas vão desde o problema até a geração de resultados.

O processo de aquisição consiste na captura da imagem por meio de um dispositivo ou sensor, transformando-a para um formato adequado para sua manipulação. Esses dispositivos podem ser câmeras, tomógrafos médicos, satélites e *scanners*. Dentre os aspectos envolvidos nessa etapa podemos citar a escolha do tipo de sensor, o formato da imagem digital, as configurações de luminosidade (tempo de abertura da lente, ISO, foco, entre outros exemplos), resolução, número de níveis de cinza ou cores da imagem digital (PEDRINI; SCHWARTZ, 2008).

A etapa de pré-processamento objetiva a melhoria da qualidade da imagem vinda da aquisição, para isso se utiliza das técnicas de atenuação de ruídos, correção de contraste e brilho. Ela é seguida pela etapa de segmentação ou limiarização (*thresholding*) que se trata do processo de separar a imagem em regiões de *pixels* similares (RUSSELL; NORVIG, 2010). A ideia utilizada é dividir a imagem nas unidades estruturais da cena ou nas que distinguem os objetos de interesse (RUSS, 1995), separando os objetos da imagem (*foreground*) das informações de fundo (*background*).

Existem diferentes modos e técnicas de se fazer a limiarização, entre os mais utilizados está a segmentação em dois níveis ou binarização. Essa técnica consiste em se definir um único valor de limiar onde os *pixels* que possuírem valores de intensidade de brilho maiores que esse limiar receberão valor 1 e os menores valor 0. Assim a imagem fica segmentada de forma binária, separando as regiões de interesse. A escolha do valor de limiar pode ser efetuado de diferentes maneiras, como por exemplo escolhendo-se o valor médio do nível de brilho, analisando a distribuição do histograma da imagem, através de algoritmos como o de Otsu (OTSU, 1975) ou através de cálculos estatísticos. Além disso, para reduzir o efeito da luminosidade sobre a segmentação pode-se utilizar o *threshold* adaptativo, que define valores de limiares para um conjunto reduzido de pixel ao longo da imagem (OPENCV, 2018a).

Então separa-se essas regiões de interesses com base na extração de características ou propriedades que possam ser usadas para distinguir classes de objetos. O processo de extração é denominado descrição e a estrutura que permite armazenar e manipular essa seleção é chamada de representação. Uma exemplo de técnica de descrição é o uso da Transformada de Hough para distinguir objetos facilmente parametrizados, como as linhas retas e círculos (DUDA; HART, 1972). Para o reconhecimento dessas formas, tal transformada tem se mostrado computacionalmente mais veloz que as técnicas baseadas na correspondência com *templates* (NIXON; AGUADO, 2012).

Além disso, pode-se fazer uso de um pós processamento nas imagens segmentadas, que facilitam a extração de características, sendo muito utilizado a matemática morfológica. As operações morfológicas mais básicas são: a erosão e a dilatação. A erosão remove *pixels* dos limites de um objeto na imagem, enquanto a dilatação adiciona *pixels* a essas bordas. Quando usadas em conjunto com as mesmas definições de vizinhança, *template* ou máscara, tem-se as operações abertura, erosão seguida de dilatação, e fechamento, dilatação seguida de erosão. A abertura é utilizada para remover pequenos objetos de uma imagem preservando a forma e o tamanho dos objetos grandes. Por sua vez o fechamento é utilizado para unificar elementos, eliminando os vazios entre eles (OPENCV, 2018b).

A partir disso pode-se partir para a última etapa que trata do reconhecimento e da interpretação dos componentes de uma imagem. Reconhecimento ou classificação é o processo de atribuição de um identificador ou rótulo aos objetos da imagem, dada

as características presentes nos seu descritores. Já a interpretação atribui um significado aos objetos reconhecidos, isto é, dado a classificação atribuir um resultado ou decisão para a situação descoberta (PERELMUTER et al., 1995). Atualmente, para a etapa de classificação é recorrente o uso de redes neurais e *deep learning*. As redes neurais convolutivas têm se mostrado uma ferramenta de alta eficácia para solucionar os problemas de reconhecimento de padrões e objetos em imagens, inclusive algumas já incorporam, além da classificação, as etapas de pré-processamento, segmentação e descrição.

A base de conhecimento, mostrado na Figura 5 corresponde a uma codificação do domínio do problema, cujo tamanho e complexidade variam conforme a aplicação. Ela deve ser usada para guiar a comunicação entre as etapas de processamento levando o conjunto de processos a executar uma determinada tarefa (PEDRINI; SCHWARTZ, 2008).

5 Atividades Realizadas

Neste capítulo são descritas as atividades práticas realizadas durante o estágio. Abordam-se as atividades de experimentações em linguagens de programação *Python* e C++, seguido do desenvolvimento de biblioteca em C++ para projeto de inspeção visual e de participação em oficinas e cursos internos.

5.1 Experimentações em linguagens *Python* e C++

Para o projeto de inspeção visual em notebook foi necessário identificar alguns defeitos na fabricação desses dispositivos. Para isso o algoritmo de Visão Computacional foi dividido em duas etapas de processamento. Na primeira reconhecia-se nas imagens as regiões de interesse que poderiam apresentar defeitos, como teclado e parafusos. A segunda consistia em utilizar técnicas de comparação entre imagens para comparar um notebook sem defeitos com os defeituosos.

Inicialmente foi desenvolvida uma série de experimentações em *Python* para encontrar as ROI (Regiões de Interesse ou *Region Of Interest*) para as teclas do teclado, para as etiquetas e para os parafusos de um notebook. Em cada uma delas foram testadas diferentes técnicas de processamento digital de imagens com o uso das bibliotecas do *OpenCV*.

5.1.1 Identificação de teclas e etiquetas do notebook

Para realização desse experimento foram utilizadas as bibliotecas *Numpy* e *OpenCV*. O código das experimentações seguiu algumas etapas comuns em algoritmos de visão: pré-processamento, segmentação, reconhecimento de contornos e seleção dos contornos.

Na etapa de pré-processamento utilizou-se o filtro espacial de média sobre a imagem convertida para a escala de cinza a fim de reduzir os ruídos de alta frequência. Essa filtragem facilita a obtenção dos pontos de interesses ao eliminar os ruídos que levam a erros ou imprecisões nas demais etapas. Em seguida foi realizado uma ajuste de contraste na imagem, para equalizar o histograma e distribuir melhor o brilho das imagens. Na Figura 6 pode-se observar as modificações realizadas na imagem original durante a etapa de pré-processamento.

A etapa de segmentação utilizou a função de *Threshold* do *OpenCV* para binarizar a imagem, isto é, segmentá-la em dois níveis. Foram feitos testes com diferentes configurações de segmentação, como através da obtenção de um limiar pelo método de Otsu (OTSU, 1975), ou utilizando uma distribuição gaussiana dos *pixels*, ou ainda com uso de segmentação

Figura 6 – Etapas de pré-processamento.

(a) Imagem original.

(b) Imagem em tons de cinza.

(c) Após o filtro de suavização.

(d) Ajuste de contraste.

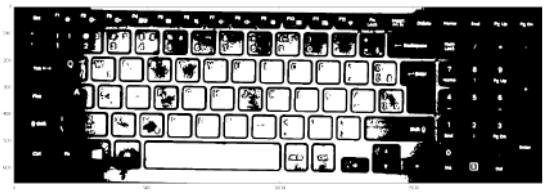
Fonte: Produzido pelo autor.

adaptativa por média. Para a biblioteca, observou-se que o último método possuía uma melhor resposta, uma vez que foi possível obter as ROIs com maior eficiência. Esse processo é necessário para a execução da próxima etapa, que necessita de uma imagem binária como entrada. Na Figura 7, observa-se o resultado de um dos experimentos realizados, pode-se notar quatro diferentes formas de *Threshold*: a primeira definindo o limiar no valor médio do brilho possível, isto é aproximadamente 127; a segunda utilizando o algoritmo de Otsu para definir o limiar; a terceira e a quarta com uso de *Threshold* adaptativo, sendo uma pela média e a outra pela distribuição gaussiana dos *pixels*. O *Threshold* adaptativo calcula um valor de limiar para pequenas porções da imagem.

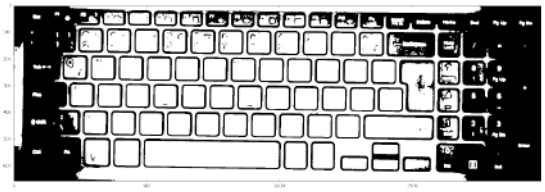
A etapa de obtenção dos contornos se utiliza de uma imagem binarizada para obter os contornos da regiões presentes. Isso é feito através do Teorema de *Green*, que relaciona uma área ao caminho fechado que a determina. Para fazer isso de forma optimizada utilizou-se a função *findContours()* do *OpenCV*. Finalmente esses contornos são selecionados a partir de suas áreas, calculadas também por meio do Teorema de *Green*. Define-se uma

Figura 7 – Testes de segmentação.

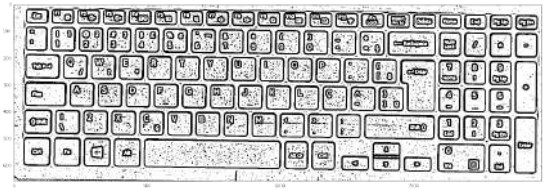
(a) Imagem original.



(b) *Threshold* global (Valor =127).



(c) *Threshold* por limiar calculado por Otsu.



(d) *Threshold* adaptativo mediano.

(e) *Threshold* adaptativo gaussiânico.

Fonte: Produzido pelo autor.

área máxima e uma área miníma e seleciona-se os contorno dentro dessa faixa e os classifica em teclas, *touchpad* e etiquetas. Por fim, é traçado a *Bounding Box* que identificará as *ROIs* selecionadas. Na Figura 8 pode-se visualizar os resultados das etapas descritas, na primeira, observa-se os contornos obtidos e na segunda, o resultado das seleção por áreas e o uso de *Bounding Box* para demarcar as *ROIs*.

Figura 8 – Detecção de teclas e etiquetas.

(a) Resultado da busca por contornos.

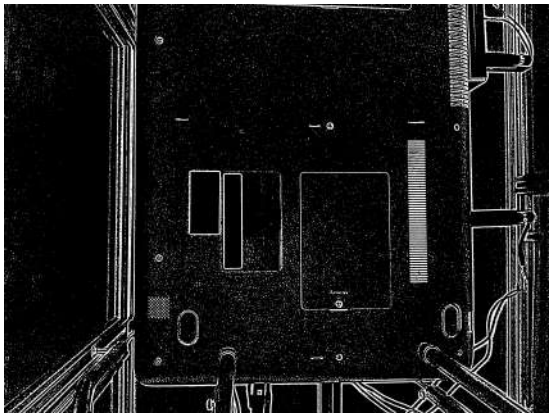
(b) Resultado da seleção por áreas.

Fonte: Produzido pelo autor.

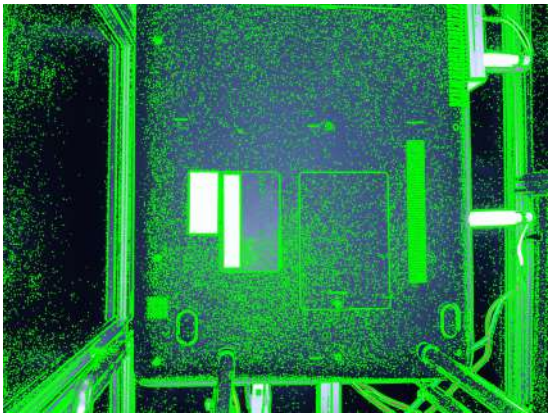
5.1.2 Identificação de parafusos e etiquetas em parte inferior do notebook

Para essa experimentação foram utilizadas as mesmas etapas de pré-processamento, isto é, conversão de escala de cores, filtragem com filtro espacial de média, e segmentação adaptativa por média. Na parte inferior, executa-se dois procedimentos, o primeiro localiza as etiquetas e o segundo, os parafusos. O procedimento para encontrar as etiquetas é similar ao utilizado na parte superior do notebook, com o teclado, utiliza-se apenas a função que encontra contornos e seleciona as áreas marcando as ROIs com a *Bounding Box*. Na Figura 9 estão ilustrado os resultados das etapas de pré-processamento e seleção de *ROIs* para as etiquetas.

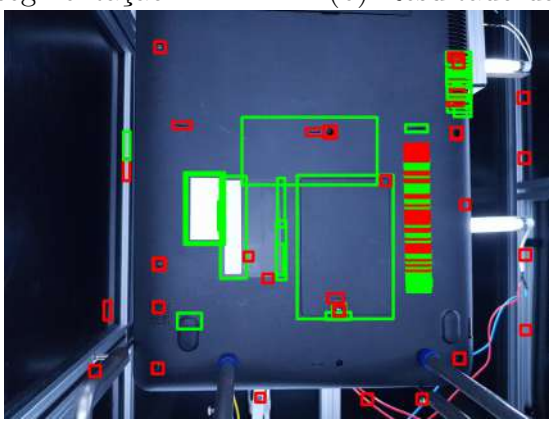
Figura 9 – Pré-processamento e detecção de etiquetas na parte inferior.



(a) Resultado da segmentação.



(b) Resultado da busca por contornos.



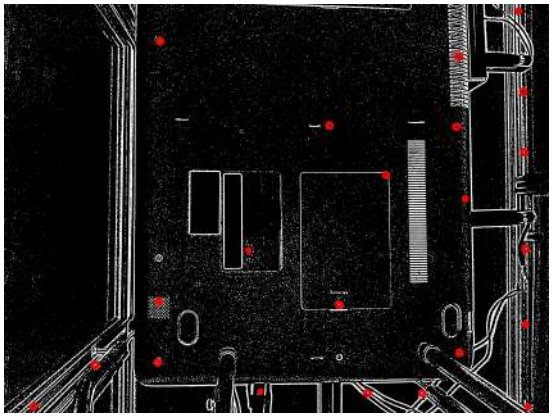
(c) Resultado da seleção de etiquetas em verde.

Fonte: Produzido pelo autor.

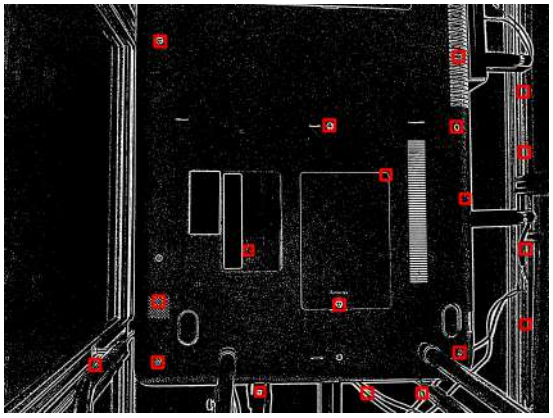
Já para os parafusos, diferentemente do processo anterior, foi utilizado a transformada de *Hough* para encontrar regiões circulares. Assim, foi então calculada a área de cada círculo encontrado convertendo as variáveis das coordenada de *Hough* para coordenadas cartesianas. Com a área, foi possível determinar um faixa de valores que continha somente os círculos próximos ao tamanho dos parafusos. Na Figura 10 observa-se o resultado da

aplicação do Transformada de *Hough* e a seleção das *ROIs* dos parafusos, como também o resultado final obtido para a parte inferior do notebook.

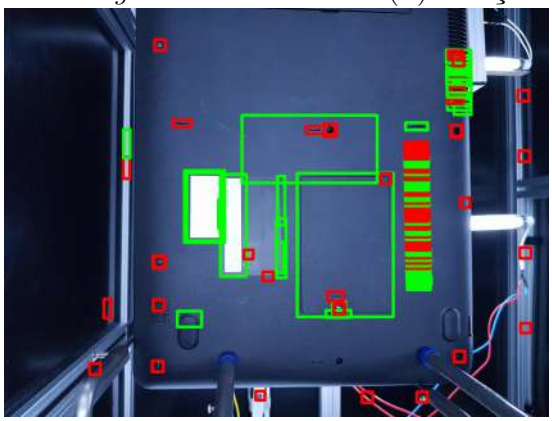
Figura 10 – Detecção de parafusos.



(a)Transformada de *Hough*.



(b) Seleção de ROIs por área.



(c) Resultado final da detecção na parte inferior do notebook.

Fonte: Produzido pelo autor.

5.1.3 Testes com codificação e decodificação

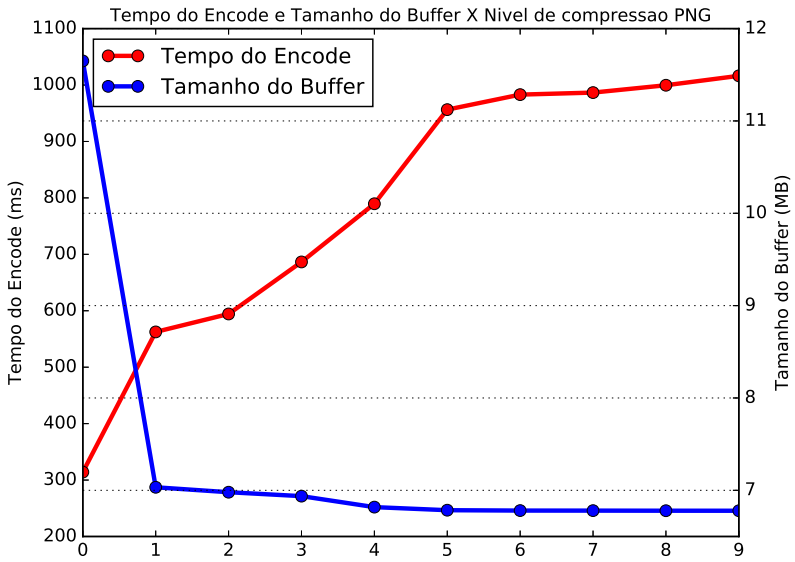
Uma das etapas do projeto consiste em enviar fotos a partir de um dispositivo móvel, por *socket*, ao computador que estará executando a biblioteca de Visão Computacional. No intuito de reduzir o tempo de transmissão das imagens optou-se por codificá-las e comprimi-las. Para isso, foram feitos testes com imagens *JPEG* e *PNG* em tons de cinza e tamanho 4032 por 3024 utilizando diferentes fatores de compressão, avaliando a perda de qualidade nas imagens, levantando curvas para determinar um ponto ótimo de operação.

Os testes foram feitos em linguagem *Python* e C++ considerando os valores de compressão de 0 a 9 no caso de imagens *PNG* e fatores de qualidade de 90 a 100 no caso de imagens *JPEG*. Para codificar e decodificar as imagens foram utilizadas as funções do *OpenCV*, *imencode()* e *imdecode()*. Os testes foram realizados dez vezes para cada fator de

compressão e foi extraído a media do tempo gasto para comprimir e descomprimir com o uso das funções de medição de tempo do *OpenCV*, para o tamanho do *buffer* o resultado obtido foi o mesmo para todas as repetições.

Para imagens em *PNG* foram avaliados apenas a relação entre o fator de compressão, o tamanho do *buffer* e o tempo para comprimir a imagem, como mostrado na Figura 11. Fatores como qualidade não foram avaliados, pois a compressão *PNG* não afeta a qualidade da imagem. Entretanto, observou-se que o tamanho de *buffer* era muito maior se comparado as imagens *JPEG* e o tempo para comprimi-lo era inviável para a aplicação.

Figura 11 – Relação tamanho do buffer e tempo de compressão para imagens PNG.

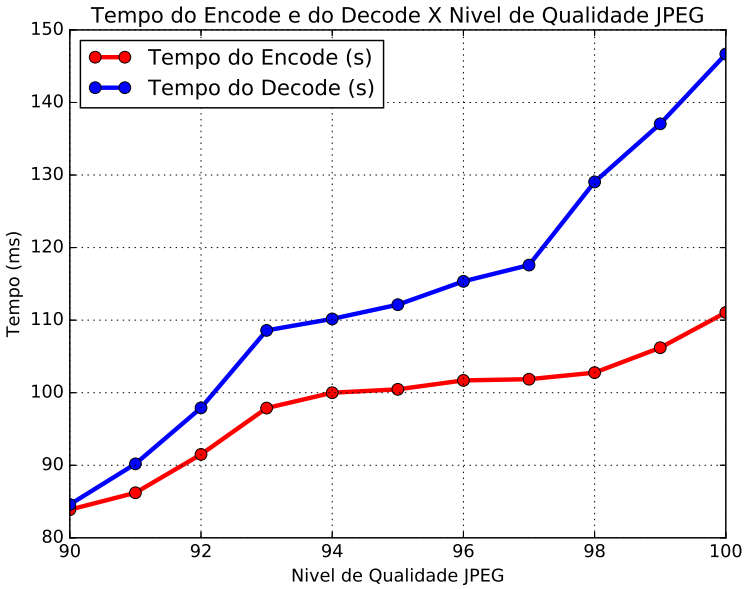


Fonte: Produzido pelo autor.

Nas imagens *JPEG*, por sua vez, foram feitos testes para avaliar não só o tempo de codificação e decodificação ou a redução no tamanho do *buffer*, como também os efeitos sobre a qualidade da imagem. Para avaliar esse efeito numericamente foi feita a diferença absoluta *pixel* a *pixel* da imagem original com a imagem codificada. A partir disso determinou-se a soma de todas as diferenças, a diferença máxima, a média das diferenças, a mediana das diferenças e o desvio padrão das diferenças. Os testes foram realizados em três imagens distintas dentro da câmara protótipo do projeto, mas com mesma resolução e em tons de cinza.

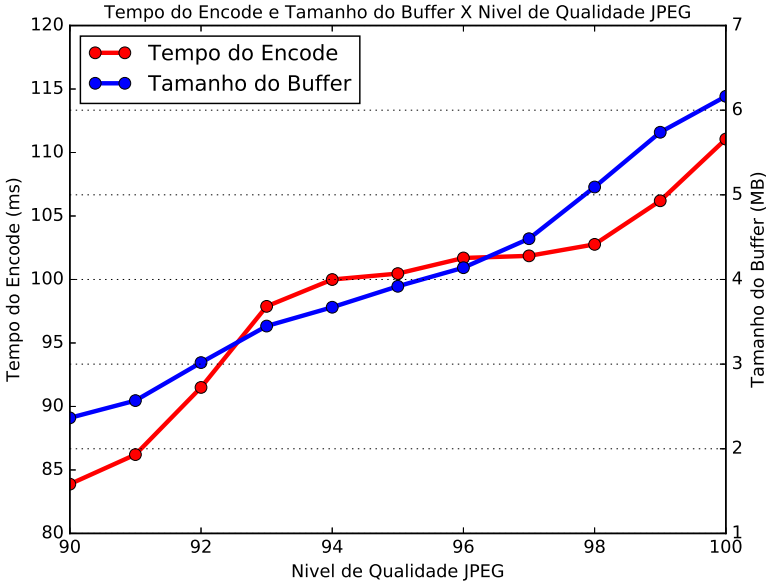
Na Figura 12 observa-se o tempo desprendido para codificar em comparação com a decodificação. Nota-se que os tempos dos dois processos são similares e menores que os tempos obtidos para as imagens *PNG*, sendo essa a maior vantagem no uso de imagens *JPEG* nesse projeto. No caso do *PNG* o tempo para decodificação não foi considerado por serem aproximadamente constantes e próximos a 160 ms para os teste realizados.

Figura 12 – Comparação tempo de codificação e decodificação por fator de qualidade em imagem JPEG.



Fonte: Produzido pelo autor.

Figura 13 – Comparação tempo de codificação e tamanho do buffer por fator de qualidade em imagem JPEG.



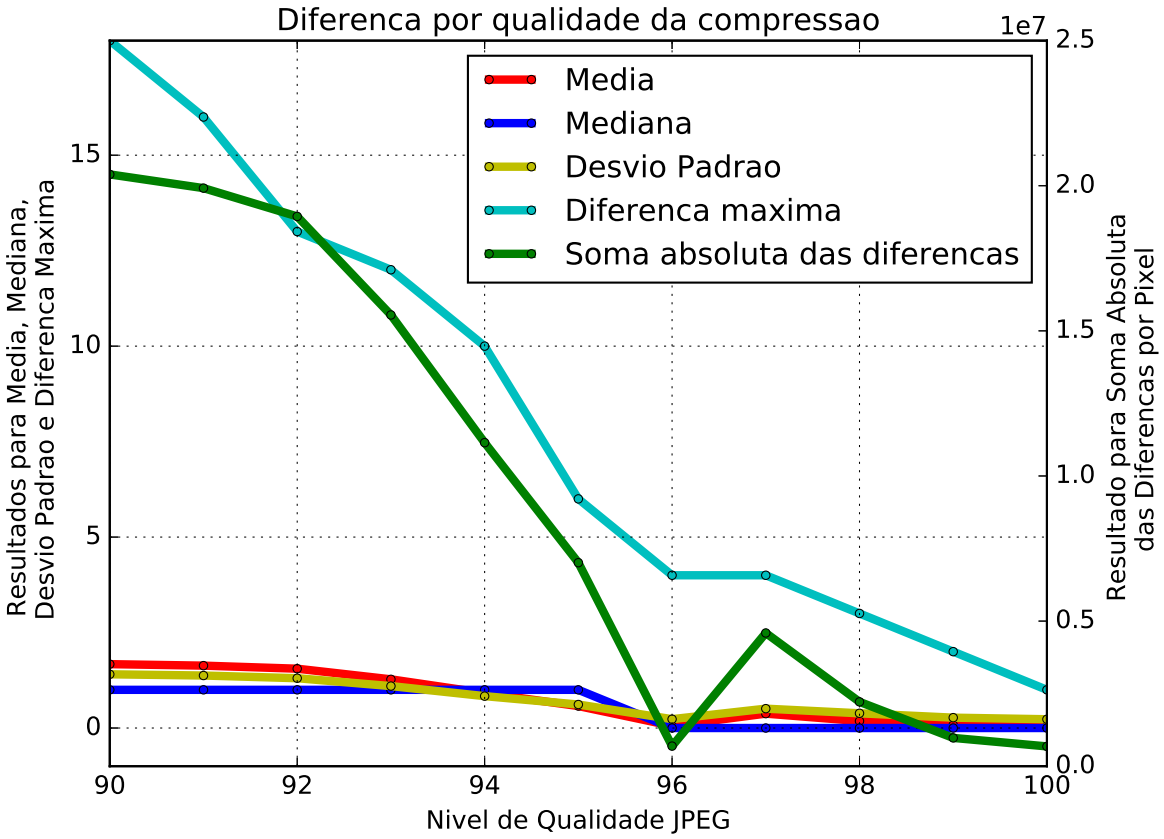
Fonte: Produzido pelo autor.

Na Figura 13 nota-se a relação da codificação com a redução do tamanho do *buffer*, dado que seu tamanho original, imagem sem compressão, é de aproximadamente 6,2 MB. Essa redução em relação ao valor de qualidade 100 é próxima de 7% quando o fator de

qualidade é 99 e chega a aproximadamente 60% com fator 90.

Por fim, na Figura 14 está exposto os valores das comparações estatísticas feitas na subtração *pixel* a *pixel* da imagem original e da imagem comprimida. Além disso, se percebe um comportamento diferenciado para o fator de qualidade 96, principalmente na análise da curva da soma absoluta da diferença *pixel* a *pixel* entre os *pixels* da imagem original e da imagem comprimida. Para esse, apesar de investigado, não foi encontrado uma justificativa coerente. Para os demais fatores de qualidade, o comportamento das curvas estatísticas geradas foram coerentes com o comportamento esperado para o processo de compressão de imagens *JPEG*.

Figura 14 – Diferenças estatísticas entre imagem original e comprimida em imagem JPEG.



Fonte: Produzido pelo autor.

Os resultados apresentados nessa experimentação foram apenas preliminares. Durante o desenvolvimento do projeto outros testes foram feitos utilizando outros dispositivos, a exemplo de um dispositivo *ANDROID* onde a compressão da imagem será executada de fato. Entretanto, esses resultados serviram para demonstrar que a possibilidade do uso da codificação e decodificação da imagens é uma alternativa válida para reduzir o tamanho do *buffer* a ser transmitido diminuindo, também, o tempo de transmissão.

5.2 Desenvolvimento de biblioteca de Visão Computacional

Após as experimentações foi preciso integrar todos os experimentos na biblioteca de visão. Para isso foi preciso adequar cada experimento e reescrever os códigos desenvolvidos em linguagem *Python* para C++. Durante essa etapa, o estagiário colaborou nas adequações das experimentações na interface da biblioteca e programou algoritmos para funções específicas, como por exemplo a que classifica uma determinada *Bounding Box* nos tipos etiqueta, teclas, serigrafia e *touchpad*.

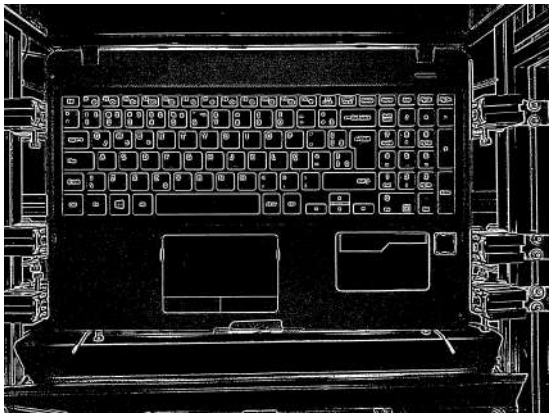
Para o algoritmo exemplificado foi utilizado nas etapas de pré-processamento um filtro morfológico de dilatação para facilitar a busca pela região do *touchpad*. Essa operação expande as regiões brancas de uma imagem binária unificando os contornos falhos ou mal definidos, isso foi realizado, pois, após a binarização da imagem o retângulo em torno do *touchpad* apresentou-se falho, como o visualizado na Figura 15-a. Após a dilatação seleciona-se as regiões de maiores áreas e determina que entre elas as mais próximas do ponto médio do eixo x da imagem são classificadas como *touchpad*, todas as regiões com áreas menores que um valor pré-determinado são do tipo serigrafia, as regiões acima do *touchpad* são teclas, com exceção das marcadas como serigrafia e as demais são classificadas como etiquetas. Na Figura 15 pode-se observar cada etapa desse algoritmo, na Figura 15-e está indicado na cor azul as áreas reconhecidas como serigrafia, na cor verde as identificadas como teclas, na cor vermelha o *touchpad* e na cor rosa as etiquetas.

Para a realização da compressão da imagem a ser transmitida foi preciso desenvolver uma biblioteca que seria executada por um dispositivo *ANDROID*. O tempo de execução de operações do *OpenCV* em *JAVA* é maior que em C++. Por isso, para que a aplicação *JAVA* possa executar uma biblioteca de visão em C++ de forma optimizada foi preciso programar uma *JNI* (*JAVA Native Interface*). A *JNI* cria uma interface de comunicação que interliga algoritmos em *JAVA* a linguagens nativas como C e C++. A *JNI* é escrita em C e nela são declaradas funções que a partir de um comando do *JAVA* chama a aplicação em C/C++ e no sentido contrário também.

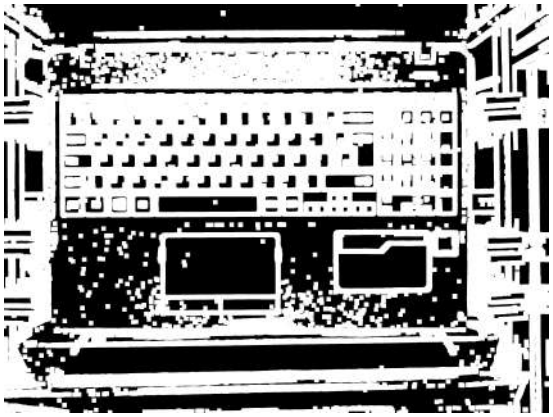
Outro desafio encontrado durante o projeto foi a compilação da biblioteca, principalmente para sistema operacional *Windows*. A compilação deveria gerar uma arquivo *.DLL* (*Dynamic Link Library*) que deveria conter não somente a biblioteca de visão desenvolvidos, mas também todas as dependências externas utilizadas, como o *OpenCV* e a biblioteca de Leitura de Código de Barras.

Para a compilação, foi utilizado a plataforma *CMAKE* associada a *toolchains* para *ANDROID* e *Windows*. Entretanto, ocorreram erros durante a compilação para *Windows*, quando esta era feita em um sistema operacional Linux utilizando o *MinGW* como compilador. Para solucionar, foi utilizado o *CMAKE* junto com o Visual Studio para realizar a compilação da *DLL* em um sistema operacional *Windows*. Para que a

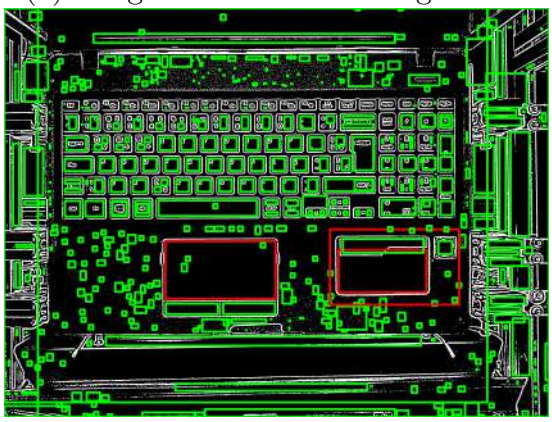
Figura 15 – Etapas do algoritmo de classificação dos tipos de *Bounding Box*.



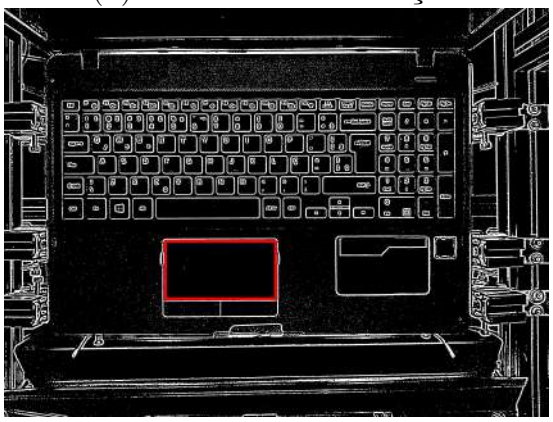
(a) Imagem de entrada do algoritmo.



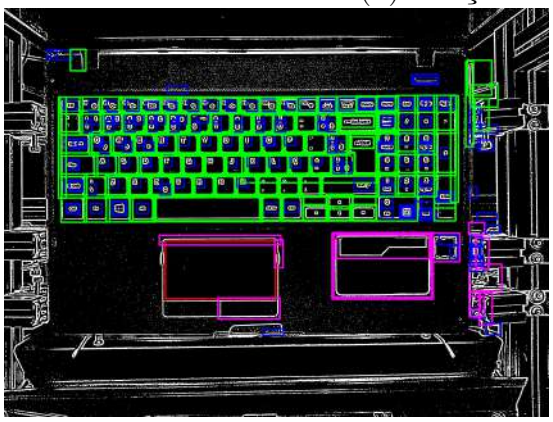
(b) Resultado da dilatação.



(c) Seleção das maiores áreas.



(d) Seleção da maior área central.



(e) Resultado final do algoritmo.

Fonte: Produzido pelo autor.

compilação ocorresse foi preciso modificar partes do código da *DLL* adequando-o para sistema *Windows* e uso do *VisualStudio* e alterar algumas configurações do *CMAKE*. Vale ressaltar que, para reduzir o tamanho da *DLL*, as bibliotecas do *OpenCV* foram também compiladas usando o *CMAKE* selecionando somente aquelas necessárias para o projeto.

Finalmente, a última etapa da integração consistiu em validar o uso da *DLL* no

ambiente do *LabView*, uma vez que esse é o ambiente que executará a *DLL* ao final do projeto. Ficou sob a responsabilidade do estagiário compilar a *DLL* em sistema *Windows* e testar seu funcionamento no *LabView*.

5.3 Participação em oficinas, cursos e reuniões de propostas

Durante o período do estágio foram desenvolvidas três oficinas sobre ferramentas necessárias para desempenho de atividades na plataforma de Computação Cognitiva. As oficinas oferecidas foram de programação em *SHELL* (*BASH*), oficina de *Docker* e curso de *JNI*.

Além disso, o estagiário teve a oportunidade de participar em reuniões com clientes, entre eles representantes de uma das maiores fabricantes de notebooks. Além disso, pôde participar de reuniões de elaboração de projetos, no qual o CPqD elabora um projeto baseado numa situação, ideia ou tema fornecidos por uma empresa. Além disso, participou também de reuniões de análises de propostas, em que, recebida uma demanda da empresa, cabe aos especialistas da área avaliar a viabilidade de execução dessa proposta e levantar questionamentos para se entender melhor o projeto. Também participou de reuniões entre os especialistas da plataforma de Computação Cognitiva discutindo temáticas sobre *Machine Learning*, Visão Computacional e biometria. Entre os assuntos discutidos estavam *reinforcement learning*, *online Learning*, aplicações com redes neurais e base de dados conhecidas, entre outros assuntos.

5.3.1 *Crash Course* de Produtividade *BASH*

O *BASH* é o *SHELL* dos sistemas operacionais *GNU*, ou seja é, uma interface que possibilita ao usuário acessar os serviços do sistema, por meio de uma linguagem baseada em comandos. Em sistemas *GNU*, como o Ubuntu, o uso de linhas de comando através do *BASH* é recorrente. Entretanto o uso de linhas de comando pode ser útil em outros sistemas como *Windows* ou *MAC*. Portanto, esse curso foi oferecido não apenas para mostrar o que é o *BASH*, mas principalmente para mostrar a potencialidade do uso de comandos em terminais de sistemas.

O curso buscou mostrar a capacidade do *BASH* com uso de comandos muitas vezes desconhecidos ou pouco explorados. Ter conhecimento do que se pode fazer foi o principal foco, buscando dessa maneira aumentar a produtividade da empresa. Tal meta se deve ao fato do uso do *BASH* possibilitar a realização de tarefas trabalhosas se feitas por interfaces gráficas de forma automatizada através de poucos comandos.

Os principais comandos abordados no curso foram comandos de busca como *find* e *grep*, o comando para realizar trabalhos em paralelo, o *parallel*, o uso do "*pipe*"para agregar comandos e o uso de expressões regulares (*regex*). O curso mostrou que ao unir

esses comandos e usá-los em conjunto abre um grande número de oportunidades, um dos exemplos citados foi o uso do *BASH* para mover um banco de imagens para um outro diretório, renomeando as imagens e alterando o formato e tamanho do arquivo com apenas uma única linha de comando.

5.3.2 *Crash Course* de *Docker*

Docker é uma plataforma para desenvolvedores e administradores de sistemas para desenvolver, implantar e executar aplicações utilizando contêineres (DOCKER INC, 2018). Na computação, contêineres são delimitadores abstratos, isto é, um objeto que contém outros objetos os quais podem ser incluídos ou removidos dinamicamente. Eles são úteis por serem:

- flexíveis, até as aplicações mais complexas, a exemplo de sistemas operacionais, podem se tornar contêineres;
- computacionalmente leves, uma vez que, compartilham e aproveitam o *Kernel* do sistema hospedeiro;
- intercambiáveis, podem receber atualizações e melhorias, inclusive durante execução;
- portáteis, podem ser criados localmente, implantados em nuvens e executados em computadores externos;
- escaláveis, podem ser aperfeiçoados e ter suas réplicas distribuídas automaticamente;
- empilháveis, seus serviços podem ser empilhados verticalmente e em processo.

De forma simplificada, o *Docker* oferece contêineres, que formam uma camada de abstração e isolamento em nível de sistema operacional, como uma máquina virtual que contém apenas as configurações necessárias para execução da atividade. Em outras palavras, essa ferramenta permite utilizar as funcionalidades do sistema operacional base, isolando as aplicações a serem instaladas e configuradas. Ou seja, no *Docker* os contêineres são instâncias de tempo de execução de uma imagem, ou seja, de pacote executável que inclui tudo o que é necessário para executar um aplicativo, como código, um tempo de execução, bibliotecas, variáveis de ambiente e arquivos de configuração (DOCKER INC, 2018).

A motivação para esse curso foi preparar a equipe para utilizar uma *GPU* compartilhada. O uso do *Docker* permite configurar a *GPU* para uma determinada atividade, sem alterar os dados gerais do sistema, isto é, cada usuário pode configurar seu contêiner e essas configurações serão isoladas e servirão apenas para essa tarefa. Isso faz com que as configurações feitas por um usuário não entrem em conflito com a de outro usuário,

isolando cada um em seu ambiente, apesar de utilizarem a mesma máquina. Além disso, por sua intercambialidade o usuário pode configurar seu contêiner em sua máquina local e apenas executá-lo na *GPU*, reduzindo o uso de tempo da *GPU*.

Durante o curso, foram apresentados as principais funções do *Docker*. O *Docker* pode ser executados por meio de linhas de comando em um terminal, como por exemplo o *BASH*. As principais rotinas mostradas foram como inserir, configurar e gerenciar uma imagem no *Docker*. Utilizou-se como exemplos base o "*hello-world*"do *Docker* presente na sua documentação e também a inserção de uma imagem do sistema operacional *Ubuntu*.

5.3.3 Treinamento em *JAVA NATIVE INTERFACE*

Uma *JNI* (*JAVA Native Interface*) é definido como método padrão para interoperabilidade entre programas *JAVA* (ou outras linguagens cujo alvo seja a *JAVA Virtual Machine* ou *JVM*) e um código nativo. Assim, a *JNI* permite com que uma aplicação *JAVA* acesse funções nativas, como o contrário também é válido. Os códigos nativos podem ser compreendidos como aqueles compatíveis com o *POSIX C*, como C/C++, *Python* e Lua. O *POSIX C* trata-se de padronização ou especificações da linguagem C padrão.

O uso de *JNI* possibilita o uso pelo *JAVA* de bibliotecas nativas, seja ela aberta ou fechada. Do lado do código nativo, permite que ele manipule objetos, faça injeção de novas classes e dispare exceções na *JVM*. Além disso, torna possível criar uma *JVM* dentro de um aplicação nativa.

As principais motivações para se fazer uma integração de *JVMs* com códigos nativos consistem no aproveitamento de um código já existente feito em linguagem nativa, no uso de recursos ocultos à plataforma *JAVA*, mas não a códigos nativos, como controlar pinos de *GPIO* de uma plataforma micro-controlável e na possibilidade de extensão das aplicações *JAVA* em outras linguagens, como a implementação de *plugins* em C++. Além disso, possibilita a optimização de porções de programa que se executados em linguagem *JAVA* tornam-se computacionalmente custosos, mas quando executados em outras linguagens são mais eficientes, a exemplo do uso de bibliotecas de processamento de imagens, como o *OpenCV*.

Devido ao grande número de projetos que envolvem o uso de smartphones com sistema operacional *ANDROID*, o uso de interfaces *JNI* para comunicar a *JVM* do dispositivo com bibliotecas nativas é bastante comum na empresa. Com o objetivo de capacitar, treinar e gerar novos desenvolvedores *JNI* foi oferecido esse curso, o qual foi formado por cinco aulas abordando as seguintes temáticas: conceitos , estrutura básica, principais funções, manipulação de objetos *JAVA*, exceções e dicas de boas práticas.

6 Conclusão

Esse relatório descreveu a atividade de estágio desenvolvida na Fundação CPqD, junto à equipe de Visão Computacional da Plataforma de Computação Cognitiva. O CPqD se mostrou um ambiente propício para o aprendizado e desenvolvimento do estágio, fornecendo ao estagiário ambiente ideal, apoio técnico, administrativo e experiências no mercado profissional.

O estagiário participou no desenvolvimento de uma solução para uma das maiores montadoras de notebooks do mundo. Tendo não somente participado de reuniões com seus representantes, mas também trabalhado no desenvolvimento da solução requisitada.

Nesse projeto de inspeção automática na linha de montagem dos notebooks por Visão Computacional, o estagiário teve a oportunidade de pôr em prática os conhecimentos adquiridos durante o curso. Foi fundamental para a execução desse projeto as disciplinas de Técnicas de Programação, Eletrônica e Informática Industrial e os conhecimentos em Processamento Digital de Imagens foram adquiridos durante as atividades de pesquisa, apesar de compor os conteúdos das disciplinas de Processamento Digital de Sinais e Sistemas de Processamento de Áudio e Vídeo.

Além desse projeto, o estagiário também teve a experiência de participar de discussões e oficinas sobre temas como Inteligência Artificial e ferramentas de programação. Ampliando os conhecimentos obtidos em disciplinas como Sistemas em Tempo Real, em que se aprende a programar por linhas de comando em *BASH*, e Automação Inteligente, na qual recebeu a base do conhecimento de IA, principalmente em temas como Redes Neurais e Algoritmos Evolutivos. Esses conhecimentos foram também necessários nas elaborações de propostas a novos clientes, nas quais o estagiário poderia fazer comentários e alterações na proposta.

O estagiário vivenciou um ambiente de mercado, onde aprendeu a trabalhar oferecendo serviços, produtos e soluções a clientes. Além disso, teve contato com um ambiente organizacional empresarial e pode vivenciar o uso de metodologias de desenvolvimento de projetos como o *SCRUM* e plataformas como o *JIRA*.

De forma adicional, pôde ampliar seus conhecimentos em Computação. Além das novas ferramentas aprendidas como *Docker* e *JNI*, aprendeu a trabalhar com uso de plataformas de controle de versão e repositórios como o *GIT* e a programar aplicações no formato *API* (*Application Programming Interface*) e compilar bibliotecas de vínculos dinâmicos e estáticos e que possam funcionar em múltiplos sistemas operacionais. Além disso, desenvolveu habilidades em linguagem de programação *Python* e ampliou seus conhecimentos sobre as bibliotecas *OpenCV* e sobre o universo do *Machine Learning* e da

Visão Computacional.

Teve a oportunidade de trabalhar com o software de engenharia *LabView*, no qual aprendeu a integrar bibliotecas em C/C++ externas. O uso desse software em disciplinas ao longo do curso facilitou a realização dessa atividade, reduzindo o tempo de adaptação do estagiário ao ambiente.

Além disso, o estagiário pôde explorar áreas do conhecimento que ainda não são foco no curso de graduação em Engenharia Elétrica da Universidade Federal da Paraíba, mas que poderiam futuramente fazer parte de disciplinas como *Machine Learning* ou Computação Cognitiva. Diante do exposto, o estágio no CPqD não apenas fomentou a formação profissional do aluno, como também ampliou o conhecimento técnico e teórico sobre as disciplinas vistas no curso e sobre os conhecimentos exigidos pelo mercado atual.

Referências

ARAUJO, A. F. de et al. Análise e caracterização de lesões de pele para auxílio ao diagnóstico médico. *Avanços em Visão Computacional*, 2012. Citado na página 20.

BALLARD, D. H.; BROWN, C. M. *Computer Vision*. Estados Unidos: Prentice Hall, 1982. Citado 2 vezes nas páginas 10 e 17.

BONATO, C. da S.; NETO, R. M. F. Etapas de pré-processamento de imagens nas técnicas de reconhecimento biométricas por digitais. *ENACOMP*, 2011. Citado na página 22.

CANNY, J. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, IEEE, p. 679–698, 1986. Citado na página 19.

COSTA, F. *Processamento de imagens como soluções práticas ao agronegócio*. 2017. Disponível em: <https://jornal.usp.br/artigos/processamento-de-imagens-como-solucoes-praticas-ao-agronegocio/>. Acesso em: 24 junho 2018. Citado na página 21.

CPQD. *Computação Cognitiva*. 2018. Disponível em: <https://www.cpqd.com.br/inovacao/computacao-cognitiva/>. Acesso em: 24 maio 2018. Citado 2 vezes nas páginas 14 e 15.

CPQD. *Inovação - CPqD*. 2018. Disponível em: <https://www.cpqd.com.br/inovacao/>. Acesso em: 24 maio 2018. Citado 2 vezes nas páginas 12 e 13.

CPQD. *Mercados - CPqD*. 2018. Disponível em: <https://www.cpqd.com.br/mercados/>. Acesso em: 24 maio 2018. Citado na página 13.

CPQD. *Pólis de Tecnologia - CPqD*. 2018. Disponível em: <https://www.cpqd.com.br/o-cpqd/polis-de-tecnologia/>. Acesso em: 24 maio 2018. Citado na página 14.

CPQD. *Sobre - CPqD*. 2018. Disponível em: <https://www.cpqd.com.br/sobre/>. Acesso em: 24 maio 2018. Citado 3 vezes nas páginas 12, 13 e 14.

DOCKER INC. *Get Started, Part 1: Orientation and setup*. Docker Docs, 2018. Disponível em: <https://docs.docker.com/get-started/#conclusion-of-part-one>. Acesso em: 24 maio 2018. Citado na página 37.

DUDA, R. O.; HART, P. E. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, ACM, v. 15, n. 1, p. 11–15, 1972. Citado na página 24.

GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006. ISBN 013168728X. Citado na página 19.

GOOGLE. *Project Sunroof*. 2018. Disponível em: <https://www.google.com/get/sunroof#p=0>. Acesso em: 25 maio 2018. Citado na página 21.

MARTINS, C. *Inovando com "Outros Olhos"*. 2018. Disponível em: <http://www.abecs.org.br/filesAbecs/cmep/12CMEP/apresentacoes/Sessao%2005%20Claudinei%20Martins.pdf>. Acesso em: 01 junho 2018. Citado na página 23.

NEVES, L. A. P.; NETO, H. V.; GONZAGA, A. (Ed.). *Avanços em Visão Computacional*. 1. ed. Curitiba, PR: Omnipax, 2012. 406 p. ISBN 978-85-64619-09-8. Citado 2 vezes nas páginas 20 e 22.

NIXON, M. S.; AGUADO, A. S. *Feature extraction & image processing for computer vision*. [S.l.]: Academic Press, 2012. Citado na página 24.

OPENALPR. *Cloud API - Programmatic plate & vehicle recognition as a service*. 2018. Disponível em: <https://www.openalpr.com/cloud-api.html>. Acesso em: 24 junho 2018. Citado na página 22.

OPENCV. *Image Thresholding*. 2018. Disponível em: <https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html>. Acesso em: 24 junho 2018. Citado na página 24.

OPENCV. *Morphological Transformations*. 2018. Disponível em: <https://docs.opencv.org/3.4/d9/d61/tutorial_py_morphological_ops.html>. Acesso em: 24 junho 2018. Citado na página 24.

OTSU, N. A threshold selection method from gray-level histograms. *Automatica*, v. 11, p. 285–296, 1975. Citado 2 vezes nas páginas 24 e 26.

PEDRINI, H.; SCHWARTZ, W. R. *Análise de Imagens Digitais: Princípios, Algoritmos e Aplicações*. São Paulo: THOMSON Learning, 2008. Citado 4 vezes nas páginas 18, 21, 23 e 25.

PERELMUTER, G. et al. Reconhecimento de imagens bidimensionais utilizando redes neurais artificiais. *Anais do VIII SIBGRAPI*, p. 197–203, 1995. Citado na página 25.

PÓLIS DE TECNOLOGIA. *Empresas instaladas*. 2018. Disponível em: <http://polisdetecnologia.com.br/o-polis/empresas-instaladas/index.html>. Acesso em: 24 maio 2018. Citado na página 14.

QUEIROZ, J. E. R. de; GOMES, H. M. *Introdução ao Processamento Digital de Imagens*. Universidade Federal da Campina Grande, 2014. Citado na página 19.

RIBEIRO, P.; PAIVA, M. S. V. de; JORGE, L. Application of texture analysis for differentiation of the greening from others pests. 10 2014. Citado na página 21.

RIOS, L. R. S. *Visão Computacional*. Universidade Federal da Bahia, 2010. Citado 4 vezes nas páginas 10, 17, 18 e 19.

RUSS, J. C. *The Image Processing Handbook*. 2. ed. [S.l.]: CRC Press, 1995. Citado na página 24.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence - A Modern Approach*. New Jersey: Pearson Education, 2010. Citado 3 vezes nas páginas 18, 19 e 24.