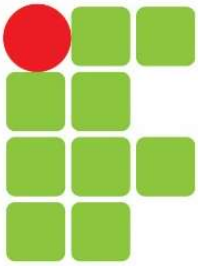


INSTITUTO FEDERAL
GOIÁS
Câmpus Inhumas

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS
CAMPUS DE INHUMAS

BACHARELADO EM INFORMÁTICA

DETECÇÃO DE BOVINOS EM UM AMBIENTE DE PASTAGEM POR INTERMÉDIO DA VISÃO COMPUTACIONAL

WILLIAN JÚNIO DE CAMPOS ALMEIDA, PAULO DOUGLAS FREITAS, ALEFF AUGUSTO SIQUEIRA

Inhumas-GO, 02 de outubro de 2019.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE GOIÁS
CAMPUS DE INHUMAS

BACHARELADO EM INFORMÁTICA

WILLIAN JÚNIO DE CAMPOS ALMEIDA, PAULO DOUGLAS FREITAS, ALEFF AUGUSTO SIQUEIRA

DETECÇÃO DE BOVINOS EM UM AMBIENTE DE PASTAGEM POR INTERMÉDIO DA VISÃO COMPUTACIONAL

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de Goiás – Campus de Inhumas, como parte dos requisitos para a obtenção do título de Bacharel em Informática.

Orientador: Prof. Me. Alexandre Bellezi José

Inhumas-GO, 02 de outubro de 2019.

Dados Internacionais de Catalogação na Publicação (CIP)

Siqueira, Aleff Augusto
S565 Detecção de bovinos em ambiente de pastagem através de visão computacional [Manuscrito]. / Aleff Augusto Siqueira; Paulo Douglas Freitas; Willian Júnio de Campos Almeida. – – Inhumas: IFG, 2019.
101f. : figs.
Bibliografia.

Orientador: Prof. Me. Alexandre Bellezi José

Trabalho de conclusão de curso de graduação em Bacharelado em Informática – IFG/Câmpus Inhumas, 2019.

1. Visão computacional. 2. OpenCV. 3. Detecção de bovinos. José, Alexandre Bellezi. (orientador). I. Título.

CDD 006.6

Ficha catalográfica elaborada pela bibliotecária Maria Aparecida Rodrigues de Souza
CRB/1-1497
Instituto Federal de Educação, Ciência e Tecnologia de Goiás
Câmpus Inhumas – Biblioteca Atena

WILLIAN JÚNIO DE CAMPOS ALMEIDA
PAULO DOUGLAS FREITAS
ALEFF AUGUSTO SIQUEIRA

DETECÇÃO DE BOVINOS EM UM AMBIENTE DE PASTAGEM POR INTERMÉDIO DA VISÃO COMPUTACIONAL

Monografia apresentada à banca examinadora da Coordenação do Curso de Bacharelado em Informática do Instituto Federal de Educação, Ciência e Tecnologia de Goiás – Unidade de Ensino Descentralizada de Inhumas, como parte dos requisitos para obtenção do título de Bacharel em Informática .

COMISSÃO EXAMINADORA

Prof. Ms. Alexandre Bellezi José
IFG-GO / Campus Anápolis

Prof. Dr. Carlos Roberto da Silveira Junior
IFG-GO / Campus Goiânia

Prof. Dr. Claudio Afonso Fleury
IFG-GO / Campus Goiânia

Inhumas, Dois de Outubro de Dois Mil e Dezenove

A todos aqueles que me apoiaram em minha jornada acadêmica e no meu desenvolvimento, em especial, a Minha Família.

AGRADECIMENTO

Primeiramente agradeço a Deus e aqueles que o acompanham por me permitir cumprir com mais este objetivo na minha vida.

A minha Família pelo apoio incondicional e incentivo mesmo diante das demasiadas dificuldades enfrentadas durante minha vida.

Ao meu Orientador e Professor Me. Alexandre Bellezi, pelos vários puxões de orelha e cobranças para o concluir desta etapa que nada mais é do que a ponta de um *iceberg* em uma longa jornada na vida acadêmica.

Ao meu Professor Dr. Carlos Roberto da Silveira Júnior pela oportunidade e a confiança de executar um projeto tão ambicioso, complexo e desafiador.

E a todos que direta ou indiretamente fizeram parte da minha formação acadêmica, meu singelo agradecimento.

Willian Júnio de Campos Almeida

A Deus por ter me dado coragem e sabedoria nas horas mais difíceis no percurso deste.

A minha família, em especial a minha mãe, Iraci Freitas, sem seu apoio nada seria possível.

Ao meu grande amigo Alef Rodrigues de Oliveira, por seu encorajamento em tempos difíceis.

Ao Orientador Professor Me. Alexandre Bellezi, por sua cobrança e incentivo nesta jornada.

Ao Professor Dr. Carlos Roberto, pela confiança de nos apresentar este projeto, que proporcionou uma enriquecedora experiência de aprendizado.

Paulo Douglas Freitas

Agradeço a todos pelo apoio nesta jornada, especialmente aos meus amigos e a todos os que me apoiaram nos momentos difíceis.

Aleff Augusto Siqueira

“Ninguém nasce sabendo, mas todos podemos aprender”

Plácido, Fernando e Terenzzo, Martha

RESUMO

Ao observar o cenário de crescimento do ramo pecuarista no final da segunda guerra mundial e a chamada revolução verde (ABIEC, 2018), é possível verificar o crescimento do setor tecnológico, o qual, proporcionou avanços principalmente relacionados com a Biotecnologia e a Tecnologia da Informação (TI). Desta forma, o presente trabalho parte da premissa de criar um sistema de detecção de bovinos em um ambiente não controlado de pastagem para o meio pecuarista a partir do uso da linguagem Python e da biblioteca OpenCV. Para tal, o trabalho a seguir é disposto em cinco partes onde na primeira apresenta o cenário que motivou o desenvolvimento do sistema e a motivação para tal. Posteriormente, serão apresentadas as ferramentas e as configurações utilizadas. Na terceira parte, tem-se os conceitos referentes ao funcionamento de um Sistema de Visão Computacional. A quarta parte é dedicada a apresentar o funcionamento da biblioteca OpenCV e do sistema de detecção proposto por Viola e Jones. Para finalizar são mostrados os resultados do trabalho, o qual, mostrou-se promissor quanto a detecção.

Palavras-Chave: Visão Computacional, OpenCV, detecção de bovinos.

ABSTRACT

Taking into account the scenario of growth of the cattle industry at the end of the Second World War and the so-called Green Revolution, this branch became attractive for the technological sector, from which it provided advances mainly related to Biotechnology and Information Technology. Therefore, the present work is based on the premise of creating a system for detecting cattle in an uncontrolled pasture environment for the cattle ranch using the Python language and the OpenCV library. For this, the present work is arranged in five parts where in the first one it presents the scenario that motivated the development of the system and the motivation for such. The tools and settings used will then be displayed. In the third part, we have the concepts regarding the operation of a Computer Vision System. The fourth part is dedicated to presenting the operation of the OpenCV library and the detection system proposed by Viola and Jones. Finally, the results of the work are shown, of which, it was promising as to the detection.

Key-Words: Computer Vision, OpenCV, cattle detection.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3D	Terceira Dimensão ou Tridimensional
ABIEC	Associação Brasileira de Importação e Exportação de Carne
BSD	*Berkeley Software Distribution License*
CUDA	*Compute Unified Device Architecture* – Arquitetura de Dispositivo de Computação Unificada
EPL	*Eclipse Public License*
FSF	*Free Software Foundation*
GPL	*General Public License*
GPU	*Graphics Processing Unit* – Unidade de Processamento Gráfico
GUI	*Graphical User Interface* – Interface Gráfica do Usuário
IA	Inteligência Artificial
IDE	*Integrated Development Environment* – Ambiente de Desenvolvimento Integrado
OSI	*Open Source Initiative*
PIL	*Python Image Library*
PyPI	*Python Package Index*
TI	Tecnologia da Informação
XML	*Extensible Markup Language*

SUMÁRIO

1. INTRODUÇÃO

Segundo a Associação Brasileira de Importação e Exportação de Carne (ABIEC), o Brasil é um dos principais criadores de gado bovino do mundo (ABIEC, 2018, p. 31). Com um rebanho de aproximadamente 221,8 milhões de animais, é responsável por aproximadamente 14,4% da produção mundial (em torno de 9,7 milhões de toneladas), atrás dos Estados Unidos, com a produção de 12,1 milhões de toneladas (ABIEC, 2018, p. 30).

Levando em conta este cenário, o ramo pecuarista atrai a atenção de vários setores, dentre eles, o setor tecnológico, que desde o final da segunda guerra mundial e a chamada revolução verde, tem passado por várias transformações, principalmente relacionadas com a biotecnologia, que tem proporcionado o aumento da produtividade do setor com o desenvolvimento de insumos (capim modificado, rações melhoradas) e criação de novas técnicas (inseminação artificial, criação intensiva) (NUNES, 2007).

Além da biotecnologia, a Tecnologia da Informação (TI) está presente no campo com sistemas contábeis, financeiros, uso de servidores, sensores, controladores, redes, sistemas de monitoramento e controle de maquinários, sistemas de irrigação, avaliação e monitoramento do solo, rastreabilidade, identificação de animais (brincos eletrônicos) dentre outras tecnologias, inclusive fazendas que utilizam de currais para extração automatizada de leite já foram implementadas no Brasil (EMBRAPA, 2018; RURAL, 2014).

Todavia, a implantação de sistemas nos ambientes rurais é influenciada por fatores socioeconômicos, culturais e contato com meios de comunicação do adotante, o que pode dificultar o desenvolvimento e implantação destes e de novos sistemas (NUNES, 2007).

Mesmo com as influências de tais fatores, as possibilidades de desenvolvimento de sistemas são amplas, o que permite suporte a várias demandas do setor (EMBRAPA, 2018). Desta forma, a proposta deste trabalho é desenvolver um sistema capaz de detectar bovinos em um ambiente de pastagem por intermédio da visão computacional.

1.1 MOTIVAÇÃO

Em um cenário onde a pecuária industrial surge como principal fonte de produção de alimentos e uma das principais fontes em termos econômicos (ABIEC, 2018), os produtores necessitam de ferramentas capazes de permitir a gerência de seus bens de produção, neste caso

em particular, os animais de produção de carne ou leite. Cada animal em si tem um valor considerável, o que já justifica um acompanhamento de cada animal do rebanho pela maior quantidade de tempo possível. Assim, o seguinte trabalho permitirá que o sistema de detecção de bovinos possa ser utilizado para monitorar os animais em seu ambiente de pastagem.

1.2 JUSTIFICATIVA

Ao verificar o cenário da agropecuária no Brasil, é possível notar que o setor ainda possui muitos pontos para informatizar (ABIEC, 2018). Além disso, é importante frisar o valor que cada animal possui, tendo demasiada importância o acompanhamento do rebanho. Sendo assim, um sistema capaz de detectar os bovinos abre espaço para novas tecnologias, principalmente com relação a análise comportamental do bovino, verificação da regularidade da alimentação e afins. Mas para a geração de tais análises, é necessário, antes de tudo, detectá-lo.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Desenvolver um sistema capaz de detectar bovinos em um ambiente não controlado de pastagem, seja a partir da captura de imagens com câmeras fixas no pasto ou através de drones sobrevoando a pastagem.

1.3.2 Objetivos Específicos

a) Aprender e utilizar a biblioteca OpenCV com seus recursos para detectar objetos em imagens.
b) Montar um banco de imagens de bovinos para realizar o treinamento utilizando a biblioteca OpenCV.
c) Tratar as imagens do treinamento para melhor desempenho do processo de detecção dos bovinos.
d) Desenvolver um protótipo de aplicativo para validar a detecção do bovino em uma imagem.

1.4 ESTRUTURA DO TRABALHO

Para a realização de tais objetivos, este trabalho encontra-se estruturado da seguinte maneira:

No Capítulo 2 serão informadas as ferramentas, configurações de ambiente e de *hardware* utilizados durante o desenvolvimento deste trabalho.

No Capítulo 3, será dada uma introdução do que vem a ser visão em termos humano e computacional.

Já o Capítulo 4 tratará sobre a biblioteca OpenCV e o processo de detecção de objetos.

E para finalizar, o Capítulo 5 demonstrará os resultados obtidos durante o desenvolvimento do presente trabalho.

2. METODOLOGIA

Existem muitas linguagens de programação disponíveis, bem como Ambientes de Desenvolvimento Integrados (*Integrated Development Environment* – IDE) gratuitas e pagas. Linguagens como C/C++ e Java além de estarem bem difundidas na comunidade de desenvolvedores, possuem uma alta complexidade no quesito de programação, deste modo, optou-se em desenvolver utilizando a linguagem de programação Python, pois a mesma possui uma estrutura e sintaxe simples (ROSSUM, 2009).

Assim, para programar em Python, é necessário configurar um ambiente que seja compatível com a linguagem de programação. Para isso, utilizou-se o Eclipse, uma plataforma de desenvolvimento que possui uma extensa comunidade de desenvolvedores além de compatibilidade com diversas linguagens de programação (ECLIPSE.ORG, 2017).

2.1 RECURSOS

As ferramentas que foram utilizadas para o desenvolvimento deste trabalho, salvo visto o Sistema Operacional, são gratuitas, sendo assim, aberta para a comunidade de desenvolvedores. As versões utilizadas para tal desenvolvimento foram as seguintes:

- Python 3.6.6
    - OpenCV 3.4.5
    - Numpy 1.14.5
    - Pillow 5.1.0
    - wxPython 4.0.3
    - PyInstaller 3.6.8 (opcional)
- Eclipse 4.8 (Oxygen) 64-bits
- Inno Setup 5 (opcional)

Vale ressaltar que, pelo Python ser uma linguagem multiplataforma, independente do Sistema utilizado (Windows ou Linux), o mesmo funcionará e, por todos os integrantes do grupo fazerem uso do Windows, foi decidido desenvolver utilizando tal plataforma. Além disso, o uso da biblioteca PyInstaller e do programa Inno Setup são para o desenvolvimento de um

arquivo executável para facilitar o uso do produto, e que este produto resultante, por ser desenvolvido em um ambiente Windows, tem compatibilidade apenas com tal sistema, em suas versões Windows 8.1 e Windows 10. E, mesmo com tal delimitação quanto ao executável, é possível executar o sistema por linha de comando em qualquer ambiente Linux ou Windows.

2.1.1 Python

O Python é uma linguagem de programação *Open Source*[1] de alto nível que enfatiza a interatividade e possui compatibilidade com diversos Sistemas Operacionais (Linux, IRIX, Compaq Tru64, OS X, Solaris e Windows). Foi idealizado por Guido van Rossum em 1991 e compartilha algumas características de linguagens *scripts* como o Shell, além de possuir características de linguagens de programação mais tradicionais como o C (PYTHON.ORG, 2019).

Segundo Russom, o Python veio para preencher uma lacuna entre as linguagens de programação C e Shell:

> O mais importante foi que, com um sistema de micro-kernel distribuído com um design radicalmente novo, as operações primitivas do [Projeto] Amoeba eram muito diferentes das operações primitivas tradicionais disponíveis no Shell. Portanto, havia a necessidade de uma linguagem que uniria a lacuna entre C e Shell (ROSSUM, 2009).

E como resultado, o Python tornou-se popular entre os desenvolvedores devido a sua sintaxe limpa e simples, além de proporcionar um aumento na produtividade (ROSSUM, 2009).

2.1.2 OpenCV

A *Open Source Computer Vision Library*, conhecida como OpenCV, foi desenvolvida pela Intel no começo de 2000 com a finalidade de oferecer melhorias para aplicações de intenso tráfego de processamento. O projeto inicial era composto por inúmeros outros projetos que trabalhavam principalmente com monitoramento em Terceira Dimensão (3D) (OPENCV.ORG, 2019).

Atualmente a biblioteca é multiplataforma (Windows, Linux, Android, iOS e Mac OS) e compatível com diversas linguagens de programação, como C/C++, Python, Java,

[1] Modelo de desenvolvimento que possui licenciamento livre, o que permite a consulta, exame ou modificação do produto.

Android e MATLAB. Além disso, possui diversas funções para programação em tempo real e conta com um otimizador para processadores, que o detecta e carrega automaticamente as configurações adequadas (OPENCV.ORG, 2019).

Vale ressaltar que, a biblioteca OpenCV é regida pela licença *BSD*[1] (*Berkeley Software Distribution license*) e seu código fonte é aberto para uso acadêmico e comercial (OPENCV.ORG, 2019).

2.1.3 Numpy

O Numpy é a biblioteca central para a computação científica em Python, pois oferece objetos de matrizes multidimensionais de alto desempenho e ferramentas que facilitam sua manipulação. Além disso, outras bibliotecas fazem uso desta, por exemplo a própria OpenCV, Matplotlib, Scikit-learn, SciPy dentre outras (NUMPY.ORG, 2019).

2.1.4 Pillow

Pillow é uma biblioteca Python originária da biblioteca PIL (*Python Image Library*) que permite abrir, manipular e salvar imagens em diferentes formatos, como PNG, TIFF, BMP, EPS e GIF (CLARK, 2019).

2.1.5 wxPython

O wxPython é uma plataforma GUI (*Graphical User Interface* – Interface Gráfica do Usuário) para Python que permite criar programas com componentes de interface gráfica. Além de ser *Open Source*, o wxPython é compatível com outras plataformas além do Windows, como Mac OS X e Linux (WXPYTHON.ORG, 2019).

[1] Licença de código fonte aberto que possui poucas restrições, colocando-a próxima do domínio público. Além disso, a licença BSD permite que os produtos distribuídos sobre ela, possa ser incorporado a outros sistemas proprietários.

2.1.6 PyInstaller

O PyInstaller é uma biblioteca destinada a criar executáveis autônomos, no Windows, Linux, Mac OS X, FreeBSD, Solaris e AIX. Além de multiplataforma, o PyInstaller cria executáveis menores devido suas ferramentas de compactação e possuir compatibilidade com pacotes de terceiros, como PyQt, Django ou Matplotlib. Vale ressaltar que, a biblioteca está registrada sobre a licença GPL[1] (PYINSTALLER.ORG, 2019).

2.1.7 Eclipse

O Eclipse é uma IDE usado para o desenvolvimento de programas. Foi desenvolvido originalmente pela IBM em 2001 e hoje está regida sob licença EPL[2] (*Eclipse Public License*). Além disso, sua licença é aprovada pela OSI[3] (*Open Source Initiative* – Iniciativa pelo código aberto) e está listado como uma licença de *software* livre pela FSF[4] (*Free Software Foundation*) (ECLIPSE.ORG, 2017).

Trabalha originalmente com Linguagens C e Java, porém, com o uso de *plugins*, é possível trabalhar com outras linguagens de programação, como Ada, ABAP, COBOL, D, Erlang, Fortran, Haskell, JavaScript, Julia, Lasso, Lua, NATURAL, Perl, PHP, Prolog, Python, R, Ruby, Rust, Scala, Clojure, Groovy e Scheme (ECLIPSE.ORG, 2017).

2.1.7.1 PyDev

O PyDev é um *plugin* que deve ser instalado no Eclipse a partir do *Eclipse Marketplace*. Este *plugin* é responsável por realizar a interpretação e execução dos códigos Python dentro do Eclipse.

---

[1] A licença GPL (*Gnu General Public License*) é uma licença de software livre que, diferente da BSD, exige que qualquer trabalho oriundo desta, seja licenciado sob a mesma licença.
[2] A licença EPL (*Eclipse Public License*) permite ao desenvolvedor usar, modificar, copiar e distribuir o trabalho e versões modificadas, em alguns casos sendo obrigados a liberar suas próprias alterações
[3] A OSI (*Open Source Initiative* – Iniciativa pelo código aberto) é uma organização sem fins lucrativos fundada em 1998 que tem por objetivo incentivar e promover o *software* de código aberto ou *software* livre.
[4] A FSF (*Free Software Foundation*) é uma organização sem fins lucrativos fundada em 1985 que tem por objetivo promover a liberdade dos usuários, defendendo os direitos destes de utilizar *software*.

2.1.8 Inno Setup

O Inno Setup é um programa que gera um arquivo de instalação para Sistemas Operacionais Windows desenvolvido e mantido desde 1997 por Jordan Russel e Martijn Laan (RUSSELL e LAAN, 2018). A partir dele, e em conjunto com o PyInstaller, é possível gerar um arquivo executável para a instalação em Sistemas Operacionais Windows.

2.2 DESCRIÇÃO E CONFIGURAÇÃO DO AMBIENTE DE DESENVOLVIMENTO

Para o desenvolvimento deste trabalho, foi necessário analisar o poder computacional disponível pois, o trabalho com Sistema de Visão Computacional se beneficia de um *hardware* potente, em principal, relacionados ao processo de treinamento, o qual, será abordado no Capítulo 5, página 28. Deste modo, foram utilizados os seguintes hardwares:

- Processador AMD PRO A10-7800B, com 12 núcleos, sendo 4 de Processamento e 8 Gráfico.
- Memória RAM com capacidade de 8 GB
- HD de 500 GB
- Monitor 21”
- Mouse e Teclado

Estas configurações foram disponibilizadas no Laboratório 4 do Instituto Federal de Ciência e Tecnologia de Goiás, Campus Inhumas.

Além do *hardware*, houve a necessidade de configurar o ambiente de desenvolvimento. Para isso, iniciou-se com o processo de configuração com a instalação do Python 3 (64-bit) e suas bibliotecas (Numpy, Pillow, wxPython, PyInstaller e OpenCV), seguidas da instalação e configuração do Eclipse 4.8 (*Oxygen*) e a instalação do Inno Setup 5, dos quais podem ser vistos na Figura 2.1. Vale ressaltar que os detalhes sobre o processo de instalação e configuração do ambiente de desenvolvimento encontra-se no APÊNDICE A: Instalação das Ferramentas, cito na página 43.

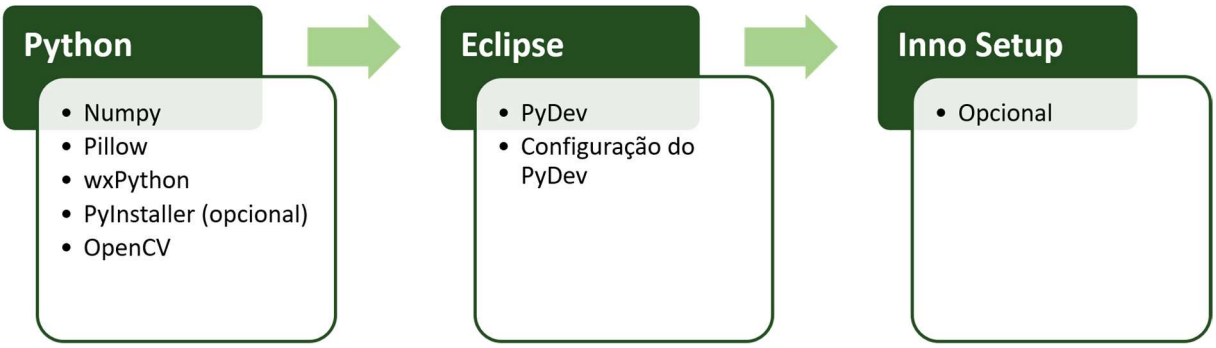


Figura 2.1 - Esquema do processo de Configuração do Ambiente de Desenvolvimento

3. SISTEMA DE VISÃO COMPUTACIONAL

Dentre os cinco sentidos do sistema sensorial, a visão, ou capacidade de detectar e interpretar a luz do ambiente (CONCEITO.DE, 2018), é um dos sentidos que o ser humano passou a depender acima dos outros. Tal dependência dá-se pelo fato da visão ser o sentido que mais fornece dados do meio em que nos encontramos (DAVIES, 2018).

Vale ressaltar que, o sistema visual não está atrelado apenas à capacidade humana. Outros animais também são providos de um sistema visual e cada um deles, com suas particularidades, como por exemplo, a capacidade dos felinos de enxergar em ambiente com baixa iluminação, ou a capacidade de um coelho de enxergar panoramicamente (DAVIES, 2018; SILVA, 2005).

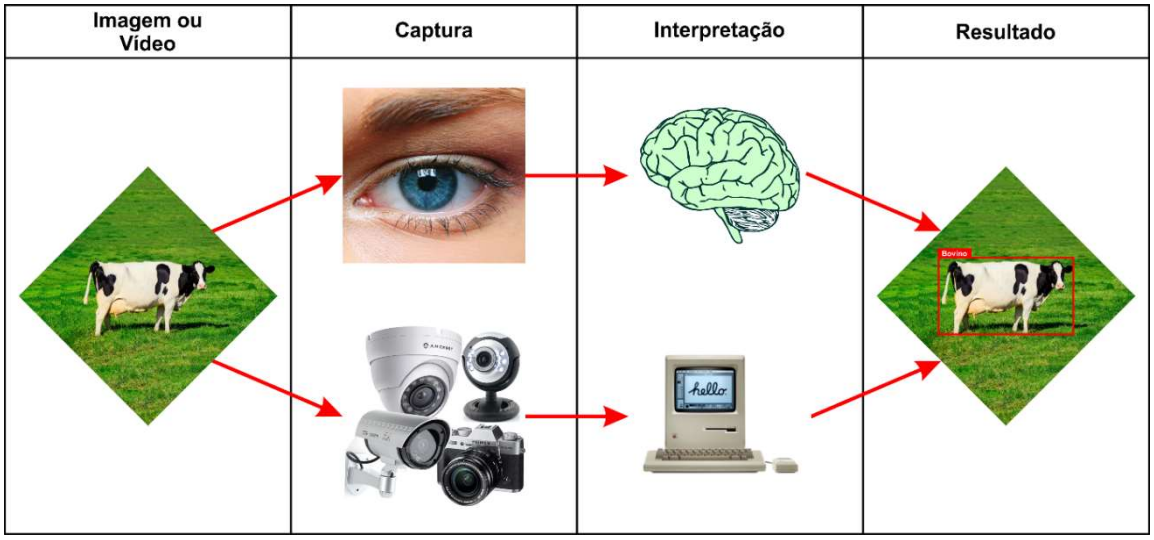


Figura 3.1 - Sistema de Visão Humano e Sistema de Visão Computacional
Fonte: (FEI-FEI, 2015), adaptado graficamente pelos autores.

Os Sistemas de Visão Computacionais são sistemas que utilizam como base, o Sistema de Visão Humana com um diferencial de possuírem uma capacidade abrangente, principalmente ao possibilitar a relação de recursos que não estão disponíveis no Sistema de Visão Humana, a recursos encontrados nos sistemas visuais de outros animais (como as capacidades visuais dos felinos e dos coelhos citados anteriormente).

3.1 SISTEMA DE VISÃO HUMANO

A visão humana é um sistema sofisticado que tem evoluído por cerca de 540 milhões de anos, e grande parte deste período, foi utilizado para aprimorar o aparelho de processamento visual no cérebro (FEI-FEI, 2015). Neste sistema, o olho, órgão formado por um conjunto de várias membranas e estruturas, é o órgão sensitivo responsável por captar as informações luminosas do ambiente e transmiti-las, em forma de impulsos elétricos, através dos nervos ópticos, para o cérebro (especificamente o córtex visual), responsável por processar e interpretar estas informações (FEI-FEI, 2015; GONZALEZ e WOODS, 2013).

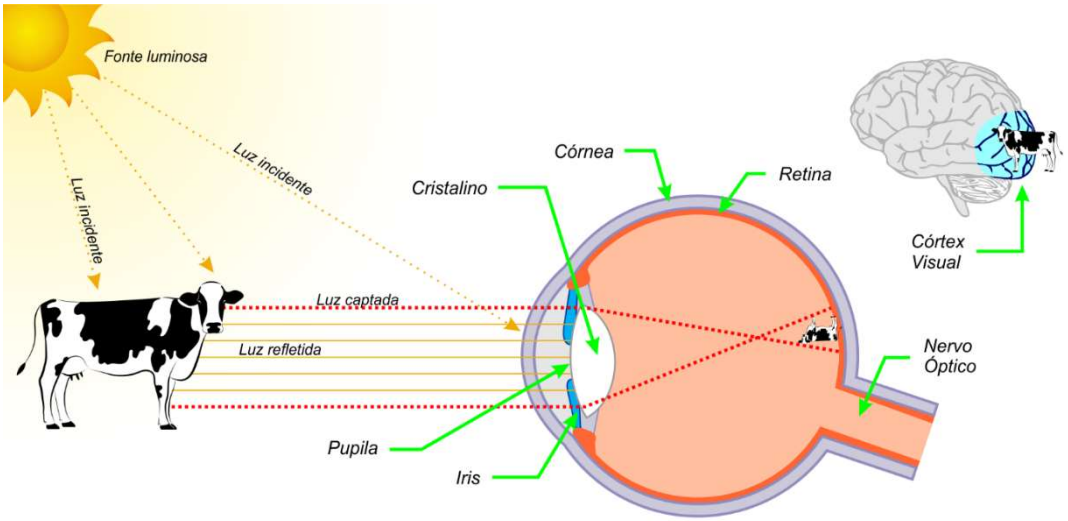


Figura 3.2 - Esquema do Funcionamento da Visão Humana
Fonte: (DAVIES, 2018; GONZALEZ e WOODS, 2013), adaptado graficamente pelos autores

As informações que o cérebro recebe através dos nervos ópticos são todas processadas em áreas diferentes do Córtex Visual. Estas áreas são recobertas por inúmeros neurônios encarregados de interpretar formas, movimentos e diversas outras funções, como diferenciação entre um bovino de um equino e a transmissão de sinais químicos e elétricos (SUPER INTERESSANTE, 2016).

3.2 SISTEMA DE VISÃO COMPUTACIONAL

O Sistema de Visão Computacional, também conhecida por Sistema de Visão Artificial ou Sistema de Visão por Computador, está incisa na grande área da Inteligência Artificial (IA) e pode ser definido, segundo Ogê Marques Filho e Hugo Vieira Neto (1999, p. 9),

"como um sistema computadorizado capaz de adquirir, processar e interpretar imagens correspondentes a cenas reais". Seu funcionamento foi idealizado a partir de estudos da Neurobiologia relacionados, principalmente à Visão Humana. Por ter origem na Visão Humana, o Sistema de Visão Computacional possui similaridades quanto sua estrutura e funcionamento (GONZALEZ e WOODS, 2013; MARQUES FILHO e VIEIRA NETO, 1999).

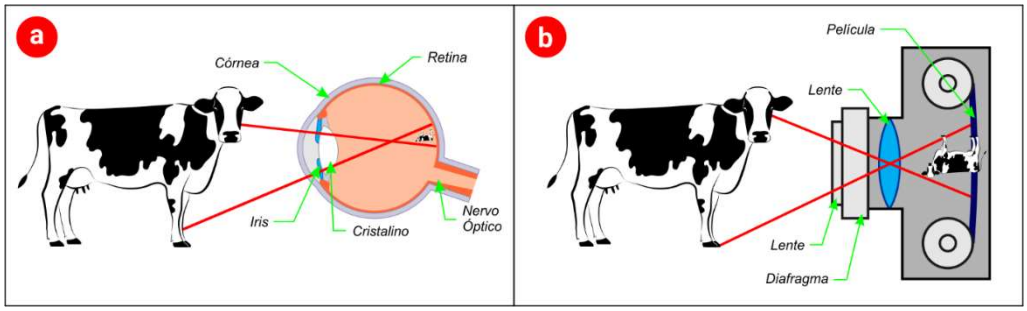


Figura 3.3 - Sistema de Visão Humano e Sistema de Visão Computacional
Em (a) tem-se um esquema ilustrativo da visão Humana; e em (b), um esquema ilustrativo utilizado por câmeras analógicas
Fonte: (GONZALEZ e WOODS, 2013), adaptado pelos autores

É possível desmembrar um Sistema de Visão Computacional em etapas, das quais, são explicadas a seguir.

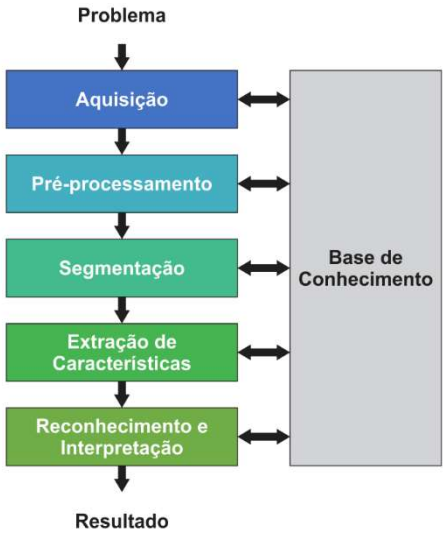


Figura 3.4 - Etapas de um Sistema de Visão Computacional
Fonte: (MARQUES FILHO e VIEIRA NETO, 1999) adaptado pelos autores

3.2.1 Problema

Um problema pode ser definido, neste contexto, como uma questão ou um determinado assunto que necessite de uma solução (CONCEITO.DE, 2018). Nos casos relacionados a

Sistemas de Visão Artificial, estes problemas estão vinculados com a necessidade de detectar, identificar e ou classificar determinados objetos em uma imagem ou cena. Sendo assim, estes problemas podem abranger qualquer temática, sobre qualquer aspecto e em qualquer área do conhecimento (GONZALEZ e WOODS, 2013).

3.2.2 Aquisição da Imagem

O processo de aquisição está relacionado à obtenção de uma imagem digital[1], que é a forma de representação digitalizada de um documento analógico, podendo ser adquirida por meio de câmeras, scanners, microscópio eletrônico, máquina de Raio X e outros. Além disso, é possível utilizar bancos de imagens disponíveis na internet. Pode-se citar como exemplo o ImageNet[2] que possui um banco de imagens categorizados e classificados, dos quais, possibilitam trabalhos relacionados à detecção de um determinado objeto ou animal, sendo o mesmo utilizado para os propósitos relacionados a este trabalho (FEI-FEI, 2015; GONZALEZ e WOODS, 2013; MARQUES FILHO e VIEIRA NETO, 1999).

3.2.3 Pré-processamento

Após a aquisição da imagem, a mesma pode apresentar pixels ruidosos, contraste e/ou brilho inadequado dentre outras falhas. Sendo assim, fica a cargo desta etapa o aprimoramento da qualidade da imagem. Tal aprimoramento pode ser realizado a partir de técnicas como o ajuste da Saturação (alto contraste), redução de ruídos (baixo contraste), a Normalização do Histograma e afins, conforme podemos observar na Figura 3.5. Como resultado, tem-se uma imagem digital de maior qualidade (GONZALEZ e WOODS, 2013; MARQUES FILHO e VIEIRA NETO, 1999).

[1] Uma imagem digital monocromática pode ser definida matematicamente como uma matriz, dos quais, cada elemento é representado pela função bidimensional $f(x, y)$, onde x e y são as coordenadas espaciais, e f define a amplitude ou intensidade de cinza neste ponto (GONZALEZ e WOODS, 2013).
[2] www.image-net.org

Figura 3.5 - Exemplos de tratamento de imagens
Em (a), tem-se uma imagem original; em (b) uma imagem em tons de cinza; e em (c) uma imagem com alto contraste a partir da normalização do histograma

3.2.4 Segmentação de Imagem

A segmentação é um processo que subdivide a imagem em regiões ou unidades significativas (MARQUES FILHO e VIEIRA NETO, 1999). O nível de detalhamento da subdivisão varia de acordo com o problema a ser resolvido, tornando a ambientação do problema um fator determinante para a segmentação, pois em muitos casos não é controlado, o que influencia diretamente a subdivisão dos objetos na imagem processada (GONZALEZ e WOODS, 2013).

O processo de subdivisão de uma imagem pode ser desenvolvido a partir de uma das categorias básicas relacionadas aos valores de intensidade, que pode ser por descontinuidade ou por similaridade (GONZALEZ e WOODS, 2013). A descontinuidade é baseada na ideia de que as fronteiras das regiões são diferentes entre si e entre o fundo da imagem o suficiente para a detecção de limites baseada na diferença local de intensidade, como por exemplo, a imagem possuir um fundo com uma única cor e que se difere da região desejada. Já a categoria de similaridade tem como principal abordagem a segmentação baseada em região que divide a imagem em regiões que tenham semelhança com critérios predefinidos ou pontos de interesses em comum (GONZALEZ e WOODS, 2013). A Figura 3.6 demonstra um exemplo de imagem que passou pelo processo de segmentação. Observe que o fundo da Figura 3.6 (b) foi removido completamente.

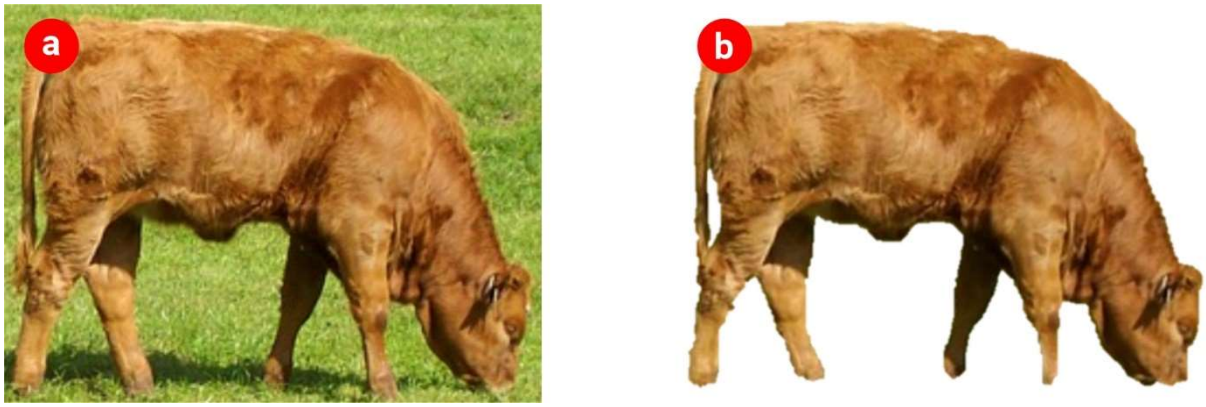


Figura 3.6 - Exemplo de Segmentação de Imagem
Em (a), tem-se a imagem original; e em (b), o resultado do processo de segmentação.

3.2.5 Extração de Características

Nesta etapa, é possível extrair as características das imagens segmentadas, utilizando-se de descritores, que por sua vez, são representados por estruturas de dados adequada ao algoritmo de reconhecimento (MARQUES FILHO e VIEIRA NETO, 1999).

Nota-se que, a entrada neste processo é dada por uma imagem e, como resultado, tem-se um conjunto de dados correspondentes a tal imagem (MARQUES FILHO e VIEIRA NETO, 1999).

3.2.6 Reconhecimento e Interpretação

Nesta etapa, é realizado reconhecimento e interpretação. O reconhecimento é definido como um processo de atribuição de um rótulo, do qual tem-se como base as características, padrões ou descritores obtidos das imagens pela etapa anterior (MARQUES FILHO e VIEIRA NETO, 1999). Após a atribuição de um rótulo, os mesmos são agrupados em classes de acordo com o grau de similaridade (GONZALEZ e WOODS, 2013).

Com o agrupamento dos rótulos, é gerado um significado ao conjunto de objetos reconhecidos, do qual, tem-se a interpretação da cena (MARQUES FILHO e VIEIRA NETO, 1999). O objetivo consiste em determinar uma classe para cada padrão com a menor interferência humana possível (GONZALEZ e WOODS, 2013).

Vale ressaltar que o presente trabalho tem por objetivo detectar ou reconhecer o bovino em um ambiente de pastagem, sendo assim, não cabe a este a realização da interpretação da cena ou o estado em que o bovino se encontra.

3.2.7 Base de Conhecimento

Para cada uma das etapas, é necessário um guia contendo as informações referentes a realização de um dos processos. Este guia é chamado de Base de Conhecimento, que contém a sequência de passos a seguir em cada uma das etapas (MARQUES FILHO e VIEIRA NETO, 1999).

3.3 SISTEMA DE VISÃO COMPUTACIONAL E A DETECÇÃO DE BOVINOS

Para este trabalho, relacionando as partes que compõem um Sistema de Visão Computacional, é possível organizar da seguinte forma:

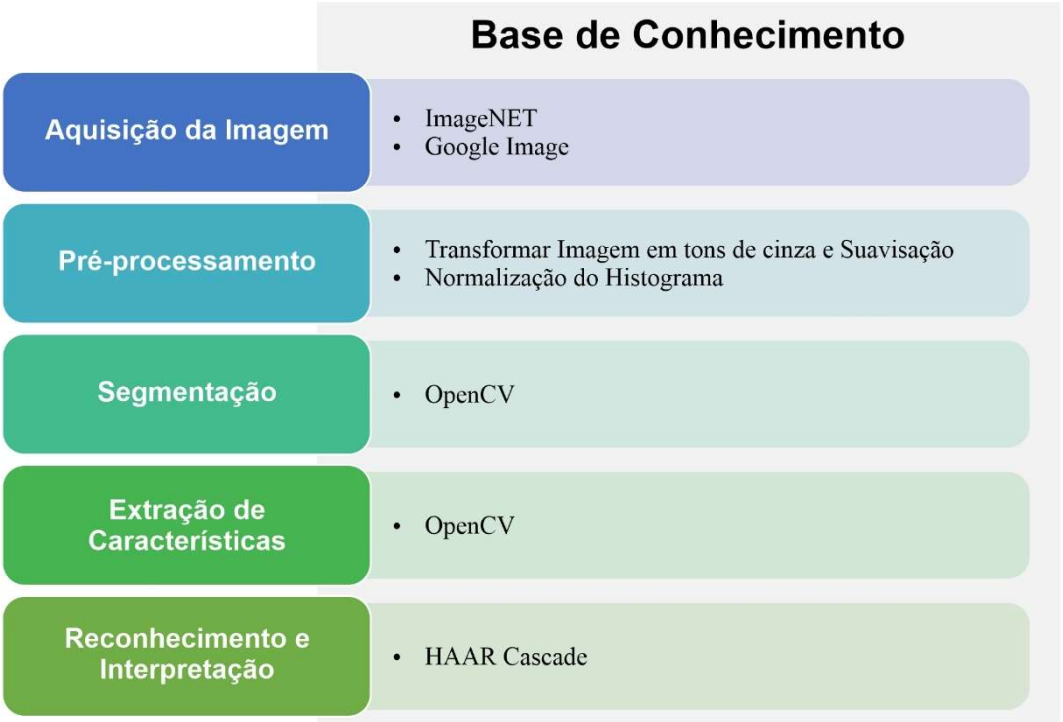


Figura 3.7 - Esquema do Sistema de Visão Computacional utilizado

Neste esquema, na área de Aquisição da Imagem foram utilizadas as imagens disponíveis na base de dados do ImageNet e completados com imagens do Google Image. No Pré-processamento, executou-se a transformação das imagens em tons de cinza com a aplicação da suavização da imagem e em outro conjunto de imagens, além de transformar em tons de cinza e suavizar, fez-se a normalização ou equalização do histograma.

As partes de Segmentação, Extração de Características e HAAR Cascade ficou a cargo da própria biblioteca OpenCV, os quais, estão realizados com os processos descritos no próximo capítulo.

4. A BIBLIOTECA OPENCV

A OpenCV (*Open Source Computer Vision Library*) é uma biblioteca de código fonte aberto desenvolvida inicialmente pela Intel, cujo projeto teve início em 1999 por Gary Bradskey com o apoio de Vadim Pisarevsky e lançada oficialmente no ano seguinte. Hoje, a biblioteca conta com uma comunidade com mais de 47.000 usuários e mais de 2.500 algoritmos otimizados que podem ser utilizados tanto para reconhecimento e detecção de objetos, quanto para classificação de ações, detecção de movimento e análise de similaridade (OPENCV.ORG, 2019).

Além disso, a biblioteca possui interfaces para o desenvolvimento em linguagem como C++, Java, Python e MATLAB, e suporta múltiplos ambientes de trabalho, como Windows, Linux, Android e Mac OS, além de suporte para operações de GPU (*Graphics Processing Unit* – Unidade de Processamento Gráfico) com alta performance baseada em CUDA (*Compute Unified Device Architecture* – Arquitetura de Dispositivo de Computação Unificada) e OpenCL[1] (OPENCV.ORG, 2019).

O processo de detecção de objetos ocorre a partir do uso de uma técnica conhecida como *Haar feature-based Cascade Classifiers* (Classificadores em Cascata Baseados em recursos Haar), proposto em 2001 por Paul Viola e Michael Jones. Esta técnica, faz uso de conjunto de imagens positivas (possui o objeto desejado para a detecção) e de outro conjunto de imagens negativas (não possui o objeto desejado para a detecção) que são utilizadas para definir padrões e possibilitar a detecção do objeto em outras imagens[2] (OPENCV.ORG, 2019).

As informações referentes aos padrões são armazenadas em um arquivo no formato XML (*Extensible Markup Language*). A própria biblioteca oferece alguns arquivos, que detectam determinados objetos[3], como cães, gatos, veículos e pessoas, porém, nem a biblioteca e nem a comunidade oferecem recursos para realizar a detecção de bovino, sendo necessário a criação de tal arquivo, o que torna este trabalho uma contribuição pois cria um novo arquivo de padrões para bovinos, como descrito no tópico 5.2 Treinamento, página 28.

[1] Arquitetura desenvolvida para que os programas funcionem em plataforma heterogêneas, constituído em CPUs e GPUs.
[2] O sentido do termo imagens, engloba também a ideia de vídeos, pois o último nada mais é do que um conjunto de imagens sequenciais.
[3] Objetos engloba quaisquer coisas, animais e afins pertencentes ao mundo real.

4.1 SISTEMA DE DETECÇÃO DA BIBLIOTECA OPENCV

Para a criação do arquivo XML contendo os padrões do objeto que será detectado, assim como ocorre em modelos relacionados à Inteligência Artificial, é necessário a realização de um treinamento. Segundo a técnica desenvolvido por Viola e Jones, o processo de treinamento funciona a partir do cálculo do nível de cinza de uma determinada sessão da imagem (retângulos horizontais ou verticais) subdividida igualmente em retângulos brancos e cinzas, nomeada de *Feature* e, a partir dos resultados obtidos no processo de treinamento, é formado um classificador (VIOLA e JONES, 2001).

4.1.1 Features

As *Features* são representações baseadas nas funções de *Haar* descritas por Constantine P. Papageorgiu em 1998 e selecionadas por Viola e Jones a partir de seus estudos para compor as *features* utilizadas em seu trabalho (VIOLA e JONES, 2001). São formas geométricas que, em determinado momento, sobreporão a imagem para realizar o cálculo do nível de cinza desta região conforme mostrado na Figura 4.1.

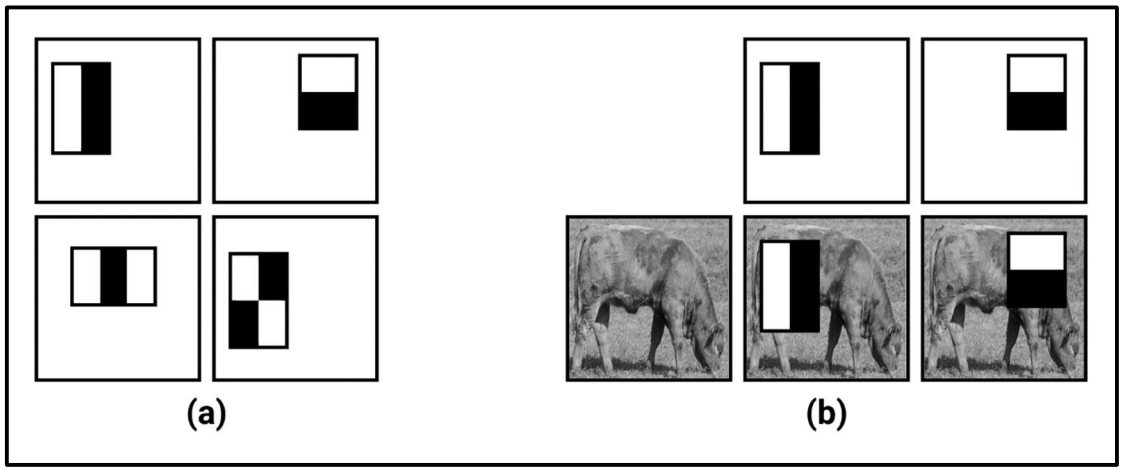


Figura 4.1 - Exemplo de *Features* selecionadas por Viola e Jones
(a) Exemplos de *Features* selecionadas por Viola e Jones; (b) Aplicação das *Features* em uma imagem.
Fonte: (VIOLA e JONES, 2001), adaptada graficamente pelos autores.

Além dos padrões selecionados por Viola e Jones, foram adicionados outros tipos de *features* que influenciaram diretamente na qualidade dos classificadores, conforme explica Rainer Lienhart e Jochen Maydt, em seu artigo intitulado "*An Extended Set of Haar-like Features for Rapid Object Detection*", publicado em 2002 (LIENHART e MAYDT, 2002). As *features* de Rainer Lienhart e Jachen Maydt podem ser visualizadas na Figura 4.2.

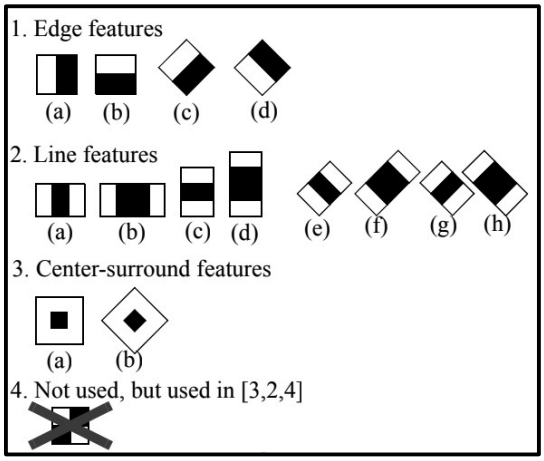
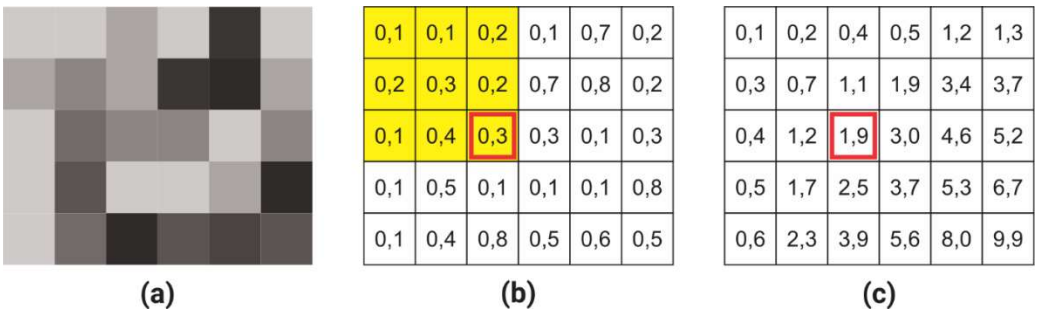


Figura 4.2 - *Features* de Lienhart e Maydt com Viola e Jones
Fonte: (LIENHART e MAYDT, 2002)

Tanto os padrões selecionados por Viola e Jones quanto os de Lienhart e Maydt são utilizados pela OpenCV, sendo estes padrões utilizados para realizar o cálculo do nível de cinza da região, a qual é feita a partir da soma dos pixels que estão no retângulo branco subtraída da soma dos pixels dos retângulos cinza (LIENHART e MAYDT, 2002; OPENCV.ORG, 2019; VIOLA e JONES, 2001). É possível observar este cálculo na Figura 4.4 na página 23.

4.1.1.1 Imagem Integral

Para agilizar e otimizar o processo de cálculo do nível de cinza de uma região da imagem através das *features*, no processamento interno, utiliza-se uma derivação da imagem original, chamada de Imagem Integral ou *Integral Image* que, segundo Derzu Omaia (2009, p. 42) pode ser definida como "uma matriz do tamanho da imagem a ser analisada, onde cada elemento da matriz possui a soma de todos os níveis de cinza dos pixels à esquerda e acima do pixel atual, incluindo o próprio valor". A definição de Omaia pode ser ilustrado pela Figura 4.3.



0,1	0,1	0,2	0,1	0,7	0,2
0,2	0,3	0,2	0,7	0,8	0,2
0,1	0,4	0,3	0,3	0,1	0,3
0,1	0,5	0,1	0,1	0,1	0,8
0,1	0,4	0,8	0,5	0,6	0,5

0,1	0,2	0,4	0,5	1,2	1,3
0,3	0,7	1,1	1,9	3,4	3,7
0,4	1,2	1,9	3,0	4,6	5,2
0,5	1,7	2,5	3,7	5,3	6,7
0,6	2,3	3,9	5,6	8,0	9,9

(a) (b) (c)

Figura 4.3 - Exemplo de uma Imagem Integral
Em (a) tem-se a imagem em escala de cinza; (b) representa os níveis de cinza de cada pixel de (a); (c) representa a Imagem integral de (b)
Fonte: (OMAIA, 2009; VIOLA e JONES, 2001), adaptado graficamente pelos autores

Na figura acima, é possível observar que o valor demarcado na Figura 4.3 (c) (1,9) representa o resultado da soma dos valores superiores e à esquerda do valor demarcado na Figura 4.3 (b) (OMAIA, 2009; VIOLA e JONES, 2001). Vale ressaltar que estas *features* percorrem toda a imagem.

4.1.1.2 Summed-Area Table

A partir do momento em que se tem uma imagem integral, para reduzir a quantidade dos cálculos, é aplicada uma técnica chamada de Tabela de Área Somada ou *Summed-Area Table* (Figura 4.4). Durante o processo, ao selecionar uma *feature*, representada na Figura 4.4 (a), é realizado um cálculo da área da seleção utilizando como base os valores obtidos da imagem integral. Dessa forma, o cálculo é feito conforme mostra a Figura 4.4 (b) e a Figura 4.4 (c). (OMAIA, 2009).

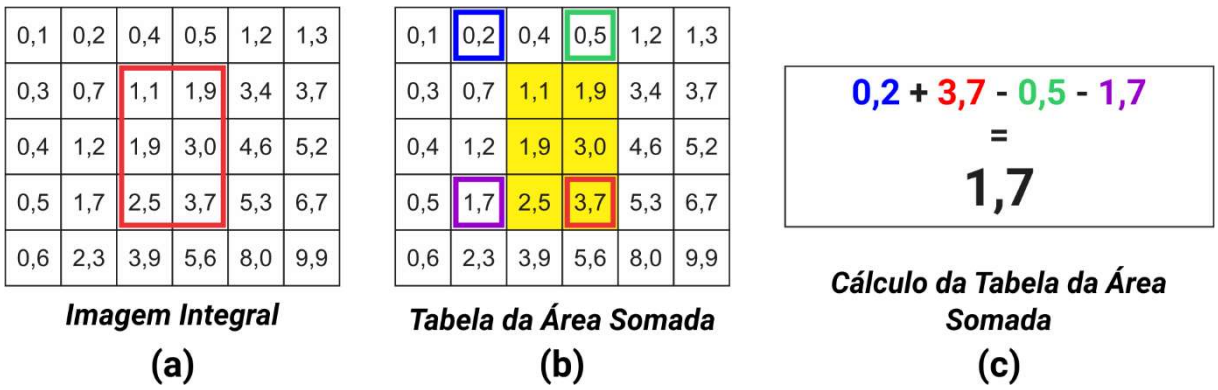


Figura 4.4 - Representação esquemática do cálculo de *Summed-Area Table*
Em (a) tem-se a representação esquemática de uma Imagem Integral com uma *feature* selecionada; em (b) os valores selecionados para a realização do cálculo da Tabela da Área Somada; e em (c), o cálculo da Tabela da Área Somada.
Fonte: (OMAIA, 2009; VIOLA e JONES, 2001), adaptado pelos autores

4.1.2 Classificadores em Cascata

Mesmo sendo um processo simples a utilização de Imagem Integral e a Técnica de *Summed-Area Table* para realizar o somatório dos níveis de cinza de uma região da imagem, existe um conjunto com muitas possibilidades para realizar a análise (cerca de 20.000 combinações possíveis de *features*) (VIOLA e JONES, 2001). Sendo assim, houve a necessidade de utilizar um algoritmo capaz de agilizar o processo de classificação, reunindo um conjunto de características (*features*) mais relevantes (OMAIA, 2009; VIOLA e JONES, 2001).

Os Classificadores em Cascata (*Cascade Classifiers*) são técnicas derivadas do algoritmo *Adaptative boosting* ou *AdaBoost* (SANTANAS, SANTOS e GOMES, 2015; OPENCV.ORG, 2019), método de aprendizado de máquina que utiliza a combinação de classificadores fracos para criar um classificador forte. Durante o treinamento o *boosting* localiza e analisa as *features*, determinando se são úteis ou não para a classificação. Assim os atributos aprovados são utilizados no classificador em cascata (OPENCV.ORG, 2019; SANTANAS, SANTOS e GOMES, 2015).

Segundo Viola e Jones, o processo em cascata ajusta os classificadores para terem um alto índice de aprovação, e assim um segundo classificador só vai ser chamado se o resultado do primeiro for positivo e assim sucessivamente. Caso algum resultado for negativo todo o processo é interrompido, e a sub-região[1] rejeitada (VIOLA e JONES, 2001). Todo o processo de detecção pode ser visto na Figura 4.5, o qual, se resume em carregar a imagem; dividir a imagem em sub-regiões; carregar individualmente cada sub-região; submeter a sub-região às cascatas para a realização do somatório do nível de cinza; rejeitar ou aprovar determinada sub-região; selecionar uma próxima sub-região até percorrer toda a imagem.

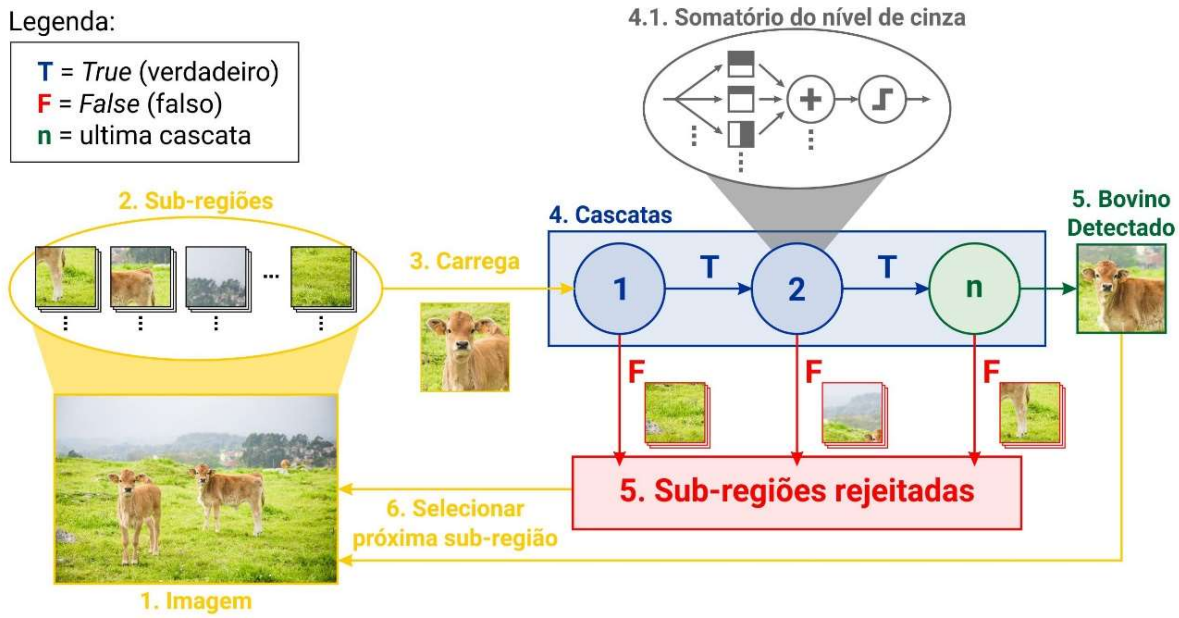


Figura 4.5 - Esquema da aprovação de uma sub-região em um Classificador em Cascata
Fonte: (SANTANAS, SANTOS e GOMES, 2015; VIOLA e JONES, 2001), adaptada pelos autores

[1] Sub-regiões: regiões da qual, uma imagem é subdividida igualmente.

5. RESULTADOS

Como resultado dos estudos e desenvolvimento, obteve-se 4 (quatro) arquivos XML relacionados à detecção de bovinos, chamados aqui de detectores. Os processos e procedimentos utilizados para a criação de tais arquivos são descritos neste capítulo. Além disso, é importante frisar que, os processos de criação de um projeto por via do sistema desenvolvido no decorrer deste trabalho, com as partes relacionadas ao treinamento e teste estão disponíveis no APÊNDICE B: Manual do Usuário, descrito a partir da página 62. Desta forma, neste capítulo será descrito apenas os resultados obtidos, sendo os processos e forma de utilização do sistema descritos no APÊNDICE B.

Com relação às configurações e valores utilizados para o desenvolvimento dos arquivos XML, teve-se como base os estudos de Abhishek Kumar Annamraju e Akash Deep Singh, em seu artigo intitulado "*Analysis and Optimization of Parameters used in Training a Cascade Classifier*", publicado em 2015 (ANNAMRAJU e SINGH, 2015) e alguns valores utilizados em tutoriais disponíveis na própria documentação da biblioteca OpenCV (OPENCV.ORG, 2019). Vale ressaltar que, devido à quantidade de recursos de *hardware* e principalmente a quantidade de imagens utilizadas no artigo de Annamraju e Singh, alguns valores sofreram adequações, as quais, serão destacadas em seus devidos momentos.

5.1 AQUISIÇÃO E PROCESSAMENTO DAS IMAGENS

Para a criação dos arquivos relacionados à detecção dos bovinos, foram utilizados 710 (setecentas e dez) imagens positivas e 2.130 (duas mil, cento e trinta) imagens negativas, as quais foram adquiridas através do site *ImageNet*[1] e do próprio *Google Image*. Em contrapartida, segundo os estudos de Annamraju e Singh (2015), seria recomendado o uso de 10.000 (dez mil) imagens positivas e 20.000 (vinte mil) imagens negativas, o que diferencia da quantidade de imagens encontradas, podendo acarretar problemas de desempenho e assertividade.

As imagens positivas foram tratadas individualmente de maneira que apenas o objeto de interesse, no caso o bovino, fosse exposto, com o objetivo de minimizar o número de

[1] http://www.image-net.org/

falsas detecções ou detecções incorretas conforme mostra a Figura 5.1 (GONZALEZ e WOODS, 2013; MARQUES FILHO e VIEIRA NETO, 1999).

Figura 5.1 - Exemplo de recorte de imagem positiva
Em (a) encontra-se a imagem adquirida na internet, em (b) e (c) são as imagens originadas da imagem (a)
Fonte: http://www.image-net.org, adaptado graficamente pelos autores

Com relação às imagens negativas, foram selecionadas imagens das quais, o objeto de interesse (bovino) normalmente se encontra, como imagens de pastagens, arvoredos, margens de rios dentre outras, com o intuito de facilitar o processo de segmentação da imagem por via da própria biblioteca (GONZALEZ e WOODS, 2013; MARQUES FILHO e VIEIRA NETO, 1999; OPENCV.ORG, 2019). É possível observar alguns exemplos de imagens negativas na Figura 5.2, onde tem-se pastagens e rio com diferentes aspectos e características.

Figura 5.2 - Exemplo de imagens negativas
Fonte: http://www.image-net.org, adaptado graficamente pelos autores

Com o banco de imagens adquirido e organizado, realizou-se o processamento das imagens. Esta etapa, segundo estudos, pode apresentar melhores resultados quanto à detecção (DAVIES, 2018; GONZALEZ e WOODS, 2013; MARQUES FILHO e VIEIRA NETO, 1999; SANTANAS, SANTOS e GOMES, 2015). Dentre os processos disponíveis, aplicou-se duas técnicas: a conversão da imagem para escala de cinza e a Equalização ou Normalização do Histograma.

A primeira técnica está relacionada a questões de otimização (GONZALEZ e WOODS, 2013) e recomendação da própria biblioteca (OPENCV.ORG, 2019), a qual, simplifica os canais RGB para Cinza (GONZALEZ e WOODS, 2013).

Já a segunda técnica está relacionada com o aumento do contraste[1] de uma imagem. Com isso, tem-se uma imagem mais demarcada em regiões que apresentam diferenças nos tons escuros e claros, conforme pode ser visto na Figura 5.3 (GONZALEZ e WOODS, 2013; MARQUES FILHO e VIEIRA NETO, 1999).

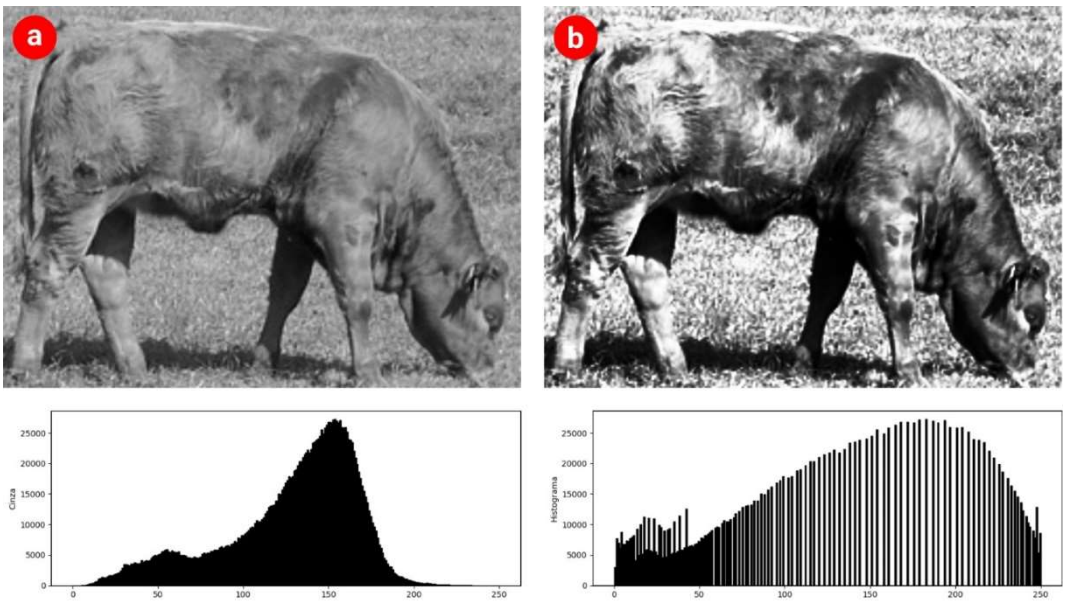


Figura 5.3 - Imagem cinza e imagem cinza com histograma normalizado
Em (a), tem-se uma imagem cinza com o gráfico do histograma da imagem abaixo; na imagem (b), a mesma imagem, porém, com a aplicação na Equalização ou Normalização do Histograma
Fonte: (GONZALEZ e WOODS, 2013; MARQUES FILHO e VIEIRA NETO, 1999), adaptado graficamente pelos autores.

[1] Contraste: está relacionado com a diferença entre as regiões claras e escuras de uma imagem (GONZALEZ e WOODS, 2013).

5.2 TREINAMENTO

Com o Banco de Imagens preparado, executou-se o processo de treinamento. Este processo é composto de configurações ou parâmetros que resultam em um arquivo XML contendo o conjunto agrupado dos classificadores fracos, tornando-o um classificador forte (OPENCV.ORG, 2019).

Assim, em linhas gerais, foram obtidos os detectores ou arquivos de detecção com as configurações expostas na Tabela 5.1. A diferença entre um detector e outro está relacionado ao tipo de imagem utilizada para o treinamento (cinza ou histograma) e na quantidade de eras, estágios ou épocas de treinamento (*numStages*).

Tabela 5.1 - Parâmetros utilizados no treinamento

Parâmetro	Valor utilizado	Valor Recomendado
Imagens Positivas	710	10.000
Imagens Negativas	2130	20.000
(*w*, *h*)[1]	[24x24]	Entre [20x20] e [35x35]
sample[2]	710	10.000
numPos[3]	85% (604)	[68% e 85%]
numNeg[4]	95% (2024)	[90% e 98%]
numStages[5]	5 ou 10	[40 e 60]
bt[6]	RAB	RAB
minHitRate[7]	0,995	[0,980 e 0,999]
maxFalseAlarmRate[8]	0,5	[0,3 e 0,5]
weightTrimRate[9]	0,95	[0,90 e 0,99]

[1] (*w*, *h*): refere-se ao tamanho das imagens utilizadas no treinamento, onde w representa a largura (*width*) e h a altura (*height*) (OPENCV.ORG, 2019).
[2] *sample*: vetor que representa o conjunto de coordenadas relacionadas ao posicionamento do objeto de interesse dentro de uma imagem positiva utilizada no processo de treinamento (OPENCV.ORG, 2019).
[3] *numPos*: número (quantidade) de imagens positivas utilizadas para o treinamento (OPENCV.ORG, 2019).
[4] *numNeg*: número (quantidade) de imagens negativas utilizadas para a realização do treinamento (OPENCV.ORG, 2019).
[5] *numStages*: quantidade de eras ou etapas que serão executadas durante o treinamento (OPENCV.ORG, 2019).
[6] *bt*: parâmetro relacionado ao *boosted type*, podendo assumir os valores DAB, RAB (utilizado neste trabalho), LB ou GAB, sendo este último o padrão (OPENCV.ORG, 2019).
[7] *minHitRate*: taxa de acerto mínima aceita para cada estágio do classificador (OPENCV.ORG, 2019).
[8] *maxFalseAlarmRate*: taxa mínima de falsos alarmes para cada estágio do classificador (OPENCV.ORG, 2019).
[9] *weightTrimRate*: representa a profundidade máxima de cada um dos classificadores fracos (OPENCV.ORG, 2019).

Parâmetro	Valor utilizado	Valor Recomendado
maxWeakCount[1]	100	[80 e 120]
featureType[2]	HAAR	LBP, HOG, HAAR

Fonte: (ANNAMRAJU e SINGH, 2015), adaptado pelos autores.
Nota: Valores obtidos pela biblioteca OpenCV (OPENCV.ORG, 2019) em união com os estudos de Annamraju e Singh (2015, p. 46). Os valores recomendados são do artigo de Annamraju e Singh (2015, p. 46)

Dessa forma, obteve-se os arquivos listados na Tabela 5.2 com suas respectivas diferenças quanto a configuração e ao tipo de imagem utilizada no processo de treinamento.

Tabela 5.2 - Nomenclatura dos arquivos XML

Nome do Arquivo	Tipo de Imagem	*numStage*
cinza-5	Cinza	5
cinza-10	Cinza	10
histograma-5	Normalização do Histograma	5
histograma-10	Normalização do Histograma	10

Fonte: Elaborado pelos pesquisadores.

Vale ressaltar que todo os passos para a realização deste procedimento por via do Gerenciador de Projetos de Detecção por Visão Computacional (GPDVC) encontra-se disponível do APÊNDICE B: Manual do Usuário, tópico 4. Treinamento, cito na página 77.

5.3 TESTES

Além das imagens positivas e negativas, foi utilizada uma coletânea de 30 (trinta) imagens organizadas aleatoriamente e que não participaram do processo de treinamento, com a finalidade testar os 4 (quatro) detectores desenvolvidos. Para tal, cada um destes detectores realizou uma sequência de detecção em cada uma das 30 (trinta) imagens pertencentes à coletânea de teste.

Por questões de análise, os objetos representados nas 30 (trinta) imagens de teste foram distribuídos de acordo com a Tabela 5.3.

[1] *maxWeakCount*: número máximo de árvores fracas para cada classificador (OPENCV.ORG, 2019).
[2] *featureType*: representa o tipo de característica que será utilizado para o treinamento, sendo estes HAAR (utilizado neste trabalho) ou LBP (padrão binário) (OPENCV.ORG, 2019).

Tabela 5.3 - Distribuição do catálogo de imagens para teste

Descrição	Quantidade
Bovinos e Representação de Bovinos	35
Outros Animais	7

Fonte: Elaborado pelos pesquisadores.

Além da coletânea de 30 (trinta) imagens para testes, é necessário a configuração de alguns parâmetros relacionados principalmente à sensibilidade da detecção e a delimitação (tamanho) mínima e máxima para detectar, sendo estes parâmetros descritos na Tabela 5.4. Cada um destes parâmetros influencia diretamente na capacidade de detecção do sistema (OPENCV.ORG, 2019).

Tabela 5.4 - Parâmetros de sensibilidade do detector

Parâmetro	Valor utilizado
scaleFactor[1]	1.3
minNeighbors[2]	25
minSize[3]	(100, 100)
maxSize[4]	(A, L)

Fonte: Elaborado pelos pesquisadores.
Nota: A: Altura total da imagem; L: Largura total da imagem

Além disso, durante o processo de teste, foram coletados alguns dados referentes às detecções nas imagens de teste com o objetivo de mensurar a qualidade do detector, conforme mostrado na Figura 5.4, explicados no APÊNDICE B: Manual do Usuário, tópico 5.1 na página 85.

[1] *scaleFactor*: define a escala de redução da imagem (OPENCV.ORG, 2019).
[2] *minNeighbors*: indica a quantidade de detecções vizinhas a imagem deve possuir para assumir como uma detecção (OPENCV.ORG, 2019).
[3] *minSize*: tamanho mínimo do retângulo que fará a marcação da detecção. Objetos que apresentarem um tamanho inferior serão ignorados (OPENCV.ORG, 2019).
[4] *maxSize*: tamanho máximo do retângulo que fará a marcação da detecção. Objetos que apresentarem um tamanho superior serão ignorados (OPENCV.ORG, 2019).

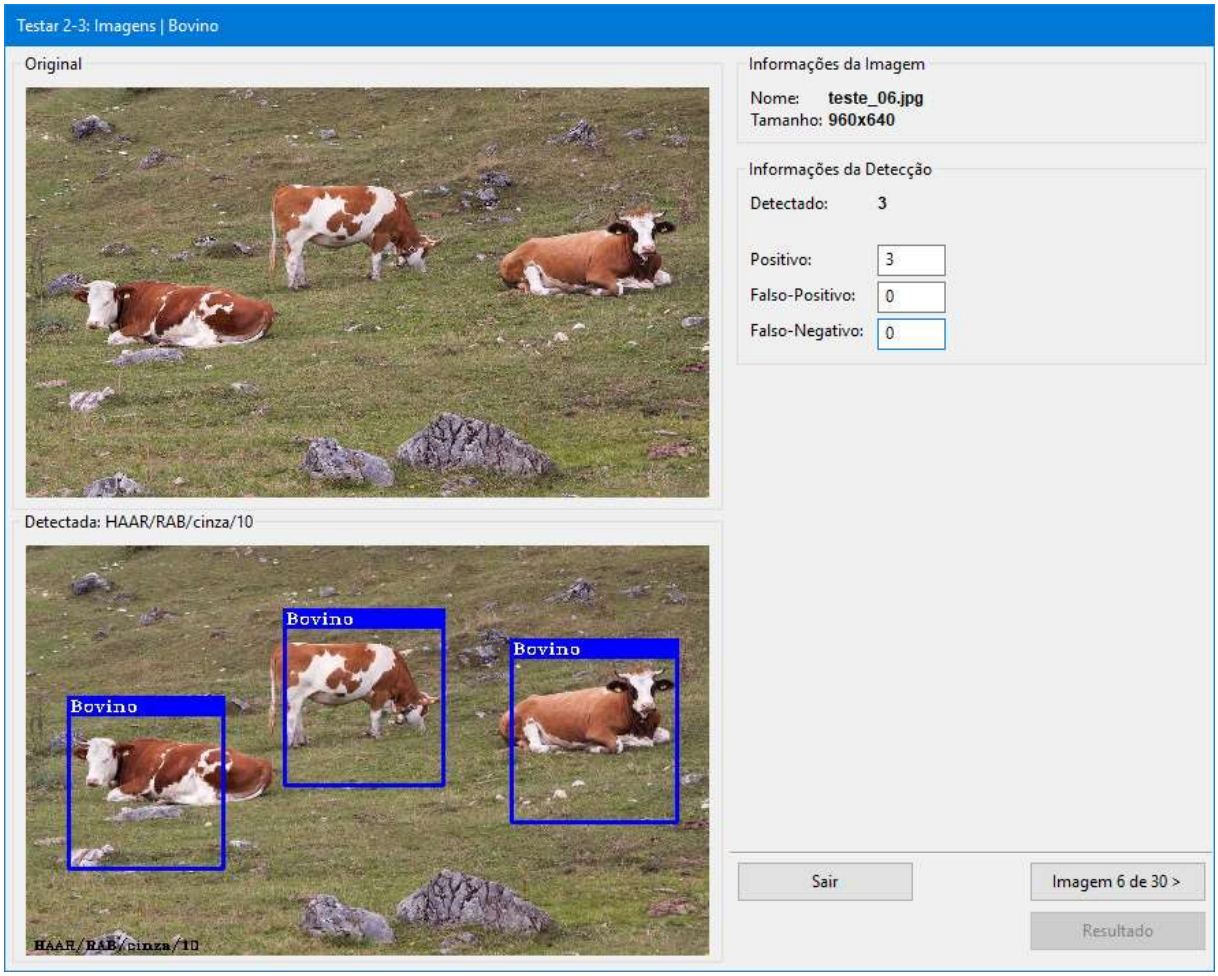


Figura 5.4 - Interface do sistema, Teste 2-3: Imagens

Observe que, na Figura 5.4 é possível informar a quantidade de detecções tidas como Positivo, Falso-Positivo e Falso-Negativo. Ao final do processo de teste nas imagens, é exibida uma tela contendo o somatório destas informações, sendo estas, utilizadas para analisar a qualidade do detector. Tal processo é descrito em detalhes no APÊNDICE B: Manual do Usuário, tópico 5.1 na página 85.

5.4 RESULTADOS

Para a contabilidade dos resultados oriundos dos detectores, realizou-se a contagem de cada uma das informações detectadas e não detectadas para cada um dos arquivos XML gerados no processo de treinamento. Tais dados foram adquiridos a partir da etapa de Teste, o qual, pode ser verificado todo o processo no APÊNDICE B: Manual do Usuário tópico 5.1 na página 85. Vale ressaltar que todas as 30 (trinta) imagens utilizadas durante a etapa dos testes

estão no tamanho padrão de 500 pixels de largura por 300 pixels de altura com um padrão de 96 DPI.

Ao submeter os detectores à detecção nas imagens de teste, foi possível gerar uma comparação entre eles, conforme pode ser visto na Figura 5.5. Todos os dados coletados das 30 (trinta) imagens de teste podem ser verificados no ANEXO 1: Dados comparativos entre os detectores, na página 42. Além disso, cada detecção demarcada na imagem foi categorizada, de acordo com o artigo de PORTO, *et al*, 2013, em:

- **Verdadeiro Positivo (VP)**: Caso o bovino fora corretamente detectado;
- **Falso Negativo (FN)**: Embora o bovino estivesse presente na imagem, o detector não conseguiu realizar a detecção;
- **Falso Positivo (FP)**: Áreas de fundo que foram erroneamente selecionadas como Verdadeiro Positivo (bovino).

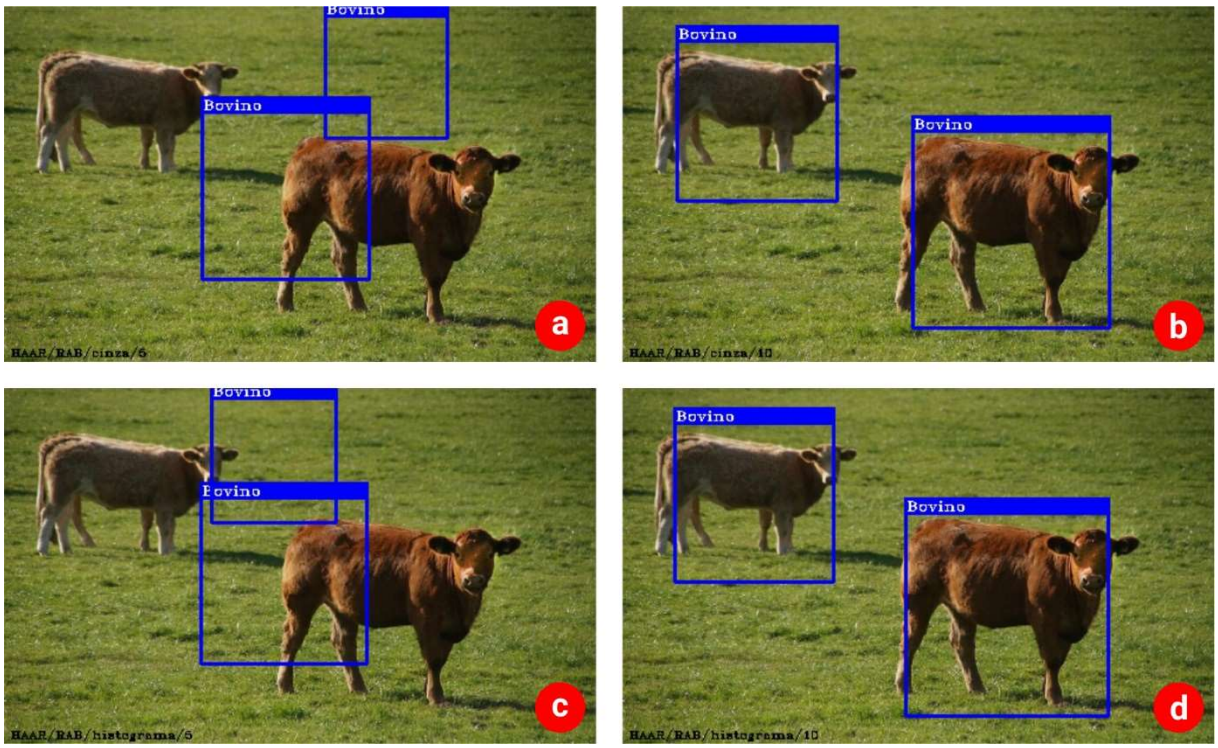


Figura 5.5 - Comparativo entre os detectores
Em (a), tem-se o detector cinza-5; em (b) o detector cinza-10; em (c), o detector histograma-5; e em (d), o detector histograma-10

Com os dados coletados, realizou-se uma relação comparativa entre os detectores, relacionadas a acurácia atingida durante a fase de teste. Tais dados estão dispostos na Tabela 5.5 (PORTO, ARCIDIACONO, *et al.*, 2013).

Tabela 5.5 - Índice Acurácia dos detectores

Índices	Ideal	Tons de Cinza 5 Eras	Tons de Cinza 10 Eras	Histograma 5 Eras	Histograma 10 Eras
Hit Rate (HR)[1]	1,20	0,43	1,03	0,43	0,93
Miss Rate (MR)[2]	0%	67%	17%	77%	27%
False Positive Rate (FPR)[3]	0%	230%	133%	227%	143%

Fonte: Elaborado pelos pesquisadores (PORTO, ARCIDIACONO, *et al.*, 2013).

Ao verificar os índices de acurácia dos detectores, nota-se que o índice de Falsos Positivos está muito acima do Ideal, o que demonstra uma dificuldade ao segmentar o bovino do fundo da imagem (PORTO, ARCIDIACONO, *et al.*, 2013). Tal problema poderia ser resolvido com uma quantidade superior de imagens (tanto positivas quanto negativas), ou com a realização de treinamento usando uma era maior. Foi possível verificar que, quanto maior a quantidade de eras, menor é o valor relacionado à detecção de Falsos Positivos, assim como ocorre nos demais índices.

Além dos fatores relacionados a acurácia dos testes, fez-se a validação das detecções, que pode ser visualizada na Tabela 5.6.

Na coluna Parâmetros estão listados os parâmetros de acordo com o artigo proposto de PORTO, et al, 2013. Cada um destes parâmetros corresponde a um valor relacionado a qualidade das detecções realizadas. Na coluna Ideal, tem-se os valores ideais de acordo com a análise das imagens que foram utilizadas no teste. A partir da coluna Ideal, tem-se cada um dos detectores obtidos com os dados correspondentes aos parâmetros listados.

Tabela 5.6 - Relação entre detectores e detecções realizadas

Parâmetros	Ideal	Tons de Cinza 5 Eras	Tons de Cinza 10 Eras	Histograma 5 Eras	Histograma 10 Eras
Branching Factor (BF)[4]	0,00	5,31	1,29	5,23	1,54
Miss Factor (MF)[5]	0,00	1,54	0,16	1,77	0,16

[1] *Hit Rate (HR)*: Razão entre o número total de Verdadeiros Positivos (VP) e o número total de imagens utilizadas durante a fase de teste (PORTO, ARCIDIACONO, *et al.*, 2013).
[2] *Miss Rate (MR)*: Razão entre o número total de Falsos Negativos (FN) e o número total de imagens utilizadas durante a fase de teste (PORTO, ARCIDIACONO, *et al.*, 2013).
[3] *False Positive Rate (FPR)*: Razão entre o número total de Falsos Positivos (FP) e o número total de imagens utilizadas durante a fase de teste (PORTO, ARCIDIACONO, *et al.*, 2013).
[4] *Branching Factor (BF)*: Relação entre o total de Falsos Positivos (FP) e o número total de Verdadeiros Positivos (VP) obtido das imagens de teste. Valores próximos a 0 (zero) indicam uma boa capacidade do detector para diferenciar bovinos do fundo (PORTO, ARCIDIACONO, *et al.*, 2013).
[5] *Miss Factor (MF)*: Relação entre o número total de Falsos Negativos (FN) e o número total de Verdadeiros Positivos (VP) obtido das imagens de teste. Valores próximos a 0 (zero) indicam uma boa capacidade do detector em reconhecer bovinos (PORTO, ARCIDIACONO, *et al.*, 2013).

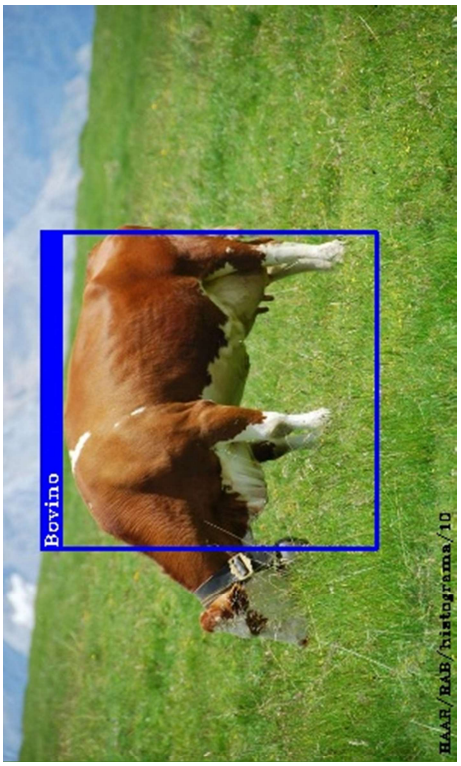
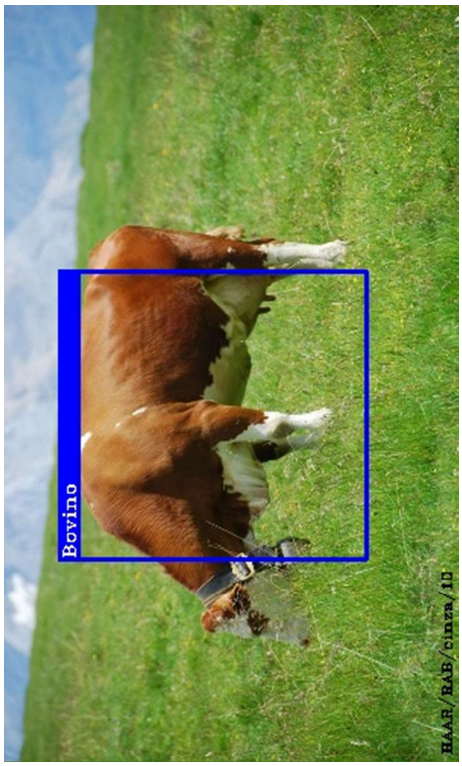
Parâmetros	Ideal	Tons de Cinza 5 Eras	Tons de Cinza 10 Eras	Histograma 5 Eras	Histograma 10 Eras
Sensitivity[1]	100%	39%	86%	36%	78%
Quality Porcentage (QP)[2]	100%	13%	41%	13%	35%

Fonte: Elaborado pelos pesquisadores (PORTO, ARCIDIACONO, *et al.*, 2013).

Ao comparar os dados, é possível verificar que, os valores mais próximos ao Ideal foram obtidos do detector Tons de Cinza com 10 Eras.

Além disso, segue abaixo algumas das imagens que foram utilizadas durante os testes. As marcações presentes nas imagens foram realizadas pelo Gerenciador de Projetos de Detecção por Visão Computacional (GPDVC) com algumas informações referentes à detecção.

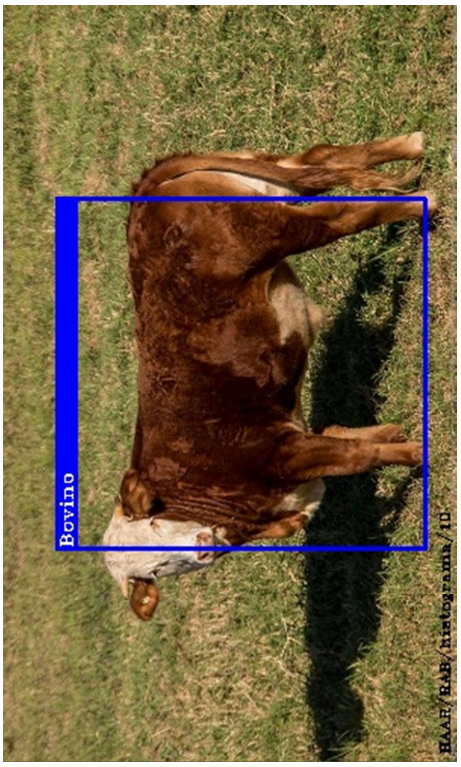
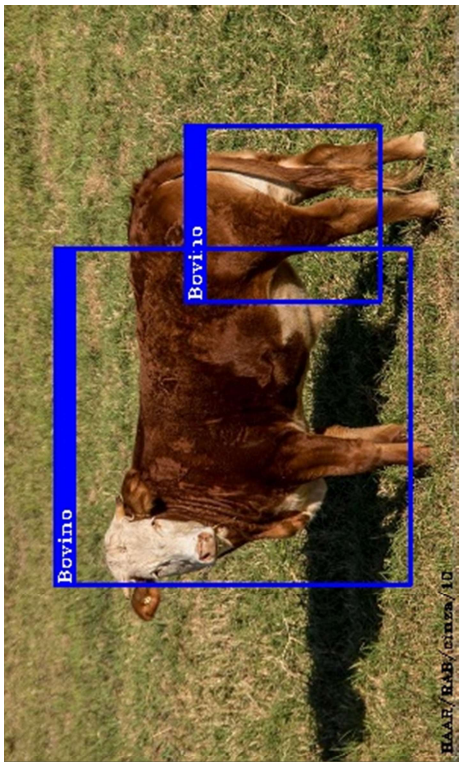
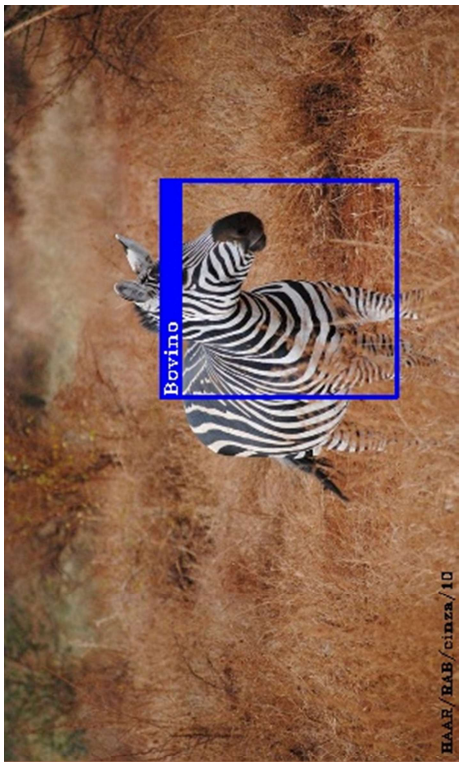
Tabela 5.7 - Comparativo entre as imagens de teste: histograma-10 e cinza-10

CÓDIGO	HISTOGRAMA-10	CINZA-10
1 HISTOGRAMA-10: Verdadeiro Positivo: 01 Falso Negativo: 00 Falso Positivo: 00 CINZA-10: Verdadeiro Positivo: 01 Falso Negativo: 00 Falso Positivo: 00		

[1] *Sensitivity*: Relação entre os bovinos detectados corretamente, ou seja, os Verdadeiros Positivos (VP) e o número total de bovinos que realmente estão presentes nas imagens, obtido a partir da soma entre os Verdadeiros Positivos (VP) com os Falsos Negativos (FN) (PORTO, ARCIDIACONO, *et al.*, 2013).
[2] *Quality Porcentage (QP)*: Em comparação com o índice de sensibilidade, essa proporção leva em consideração a presença dos Falsos Positivos nas imagens, ou seja, áreas do fundo classificadas incorretamente como bovinos (PORTO, ARCIDIACONO, *et al.*, 2013).

CÓDIGO	HISTOGRAMA-10	CINZA-10
2		
HISTOGRAMA-10	Verdadeiro Positivo: 03 Falso Negativo: 00 Falso Positivo: 02	
CINZA-10	Verdadeiro Positivo: 03 Falso Negativo: 00 Falso Positivo: 00	
3		
HISTOGRAMA-10	Verdadeiro Positivo: 03 Falso Negativo: 00 Falso Positivo: 00	
CINZA-10	Verdadeiro Positivo: 03 Falso Negativo: 01 Falso Positivo: 00	

CÓDIGO	HISTOGRAMA-10	CINZA-10
4		
HISTOGRAMA-10	Verdadeiro Positivo: 01 Falso Negativo: 00 Falso Positivo: 01	
CINZA-10	Verdadeiro Positivo: 01 Falso Negativo: 00 Falso Positivo: 02	
5		
HISTOGRAMA-10	Verdadeiro Positivo: 02 Falso Negativo: 00 Falso Positivo: 01	
CINZA-10	Verdadeiro Positivo: 02 Falso Negativo: 00 Falso Positivo: 00	

CÓDIGO	HISTOGRAMA-10	CINZA-10
6 HISTOGRAMA-10: Verdadeiro Positivo: 01 Falso Negativo: 00 Falso Positivo: 00 CINZA-10: Verdadeiro Positivo: 01 Falso Negativo: 00 Falso Positivo: 01		
7 HISTOGRAMA-10: Verdadeiro Positivo: 00 Falso Negativo: 00 Falso Positivo: 00 CINZA-10: Verdadeiro Positivo: 00 Falso Negativo: 00 Falso Positivo: 01		

É possível observar que, em algumas detecções, o detector histograma-10 se sobressaiu em relação ao cinza-10. Mesmo assim, de forma geral, houve menos detecções excessivas de um único objeto por parte do detector cinza-10, conforme visto nas imagens 2, 3 e 5.

6. CONCLUSÃO

De forma geral, é possível verificar que, mesmo com a quantidade de imagens e de eras utilizadas para o treinamento, o qual foge do escopo proposto por Annamraju e Singh (ANNAMRAJU e SINGH, 2015), atingiu-se resultados promissores com uma capacidade de detectar ou reconhecer um bovino de 86% e um índice qualitativo de 41% (PORTO, ARCIDIACONO, *et al.*, 2013).

Ao analisar os dados coletados durante o procedimento de teste, é possível verificar que grande parte relacionada ao baixo índice qualitativo está atrelado à grande quantidade de Falsos Positivos detectados, atingindo 133% como o melhor resultado obtido entre os 4 (quatro) detectores, podendo tal índice ser melhorado, ou com o aumento no número de imagens positivas e negativas, ou com o aumento no número de eras de treinamento, o que não pode ser realizado devido principalmente ao tempo de uso disponível do Laboratório 4.

7. TRABALHOS FUTURO

Desta forma, com a finalização da proposta inicial do trabalho e, analisando as possibilidades em que este trabalho possa ser melhorado, abaixo, apresenta-se uma lista com possíveis temas ou melhorias:

- Aumentar a quantidade de imagens positivas e negativas e a quantidade de eras de treinamento para melhorar os índices de acurácia dos detectores;
- Analisar o comportamento do bovino detectado e relacionar a possíveis problemas de saúde, por exemplo;
- Verificar se o bovino está se alimentando regularmente;
- Realizar a contagem regular do rebanho;
- Verificar a presença de pessoas no ambiente de pastagem.
- Melhoramento na usabilidade e legibilidade do processo no Gerenciador de Projetos de Detecção por Visão Computacional (GPDVC) (tempo de carregamento, cópia, visual);
- Aplicação de outros filtros e técnicas de aprimoramento das imagens antes do processo de treinamento.

REFERÊNCIAS

ABIEC. **Perfil da Pecuária no Brasil: Relatório Anual**. São Paulo. 2018.

ANNAMRAJU, A. K.; SINGH, A. D. Analysis and Optimization of Parameters used in Training a Cascade Classifier. **Advances in Image and Video Processing**, Goa, India, v. 3, n. 2, p. 25-48, abr. 2015. ISSN 2054-7412.

CLARK, A. Pillow. **Pillow**, Janeiro 2019. Disponivel em: <https://pillow.readthedocs.io/en/stable/>. Acesso em: mar. 2019.

CONCEITO.DE, 2018. Disponivel em: <https://conceito.de/>. Acesso em: jan. 2019.

DAVIES, E. R. **Computer and Machine Vision:** Theory, Algorithms, Practicalities. 5. ed. Egham: Elsevier, 2018. ISBN 978-0-12-809284-2.

ECLIPSE.ORG. Eclipse Oxygen. **Eclipse Foundation**, 2017. Disponivel em: <http://www.eclipse.org/oxygen/>. Acesso em: fev. 2018.

EMBRAPA. **Embrapa**, 2018. Disponivel em: <https://www.embrapa.br/>. Acesso em: maio 2018.

FEI-FEI, L. How we're teaching computers to understand pictures, 2015. Disponivel em: <https://www.ted.com/talks/fei_fei_li_how_we_re_teaching_computers_to_understand_pictures>. Acesso em: jan. 2019.

GONZALEZ, R. C.; WOODS, R. E. **Processamento Digital de Imagens**. 3. ed. São Paulo: Pearson, 2013. ISBN 978-85-8143-586-2.

LIENHART, R.; MAYDT, J. An Extended Set of Haar-like Features for Rapid Object Detection. **Proceedings. International Conference on Image Processing**, Rochester, NY, USA, set. 2002.

MARQUES FILHO, O.; VIEIRA NETO, H. **Processamento Digital de Imagens**. Rio de Janeiro: Brasport, 1999. ISBN 8574520098.

NUMPY.ORG. About NumPy: NumPy. **NumPy**, 2019. Disponivel em: <http://www.numpy.org>. Acesso em: mar. 2019.

NUNES, S. P. O desenvolvimento da agricultura brasileira e mundial e a idéia de Desenvolvimento Rural. **Boletim Eletrônico do DESER - Departamento de Estudos Sócio-Econômicos Rurais**, n. 157, p. 5-17, mar. 2007.

OMAIA, D. **Um sistema para detecção e reconhecimento de face em vídeo utilizando a transformada cosseno discreta**. João Pessoa: Universiade Federal da Paraíba, 2009. Dissertação (Mestrado em Informática).

OPENCV.ORG. OpenCV (Open Source Computer Vision). **OpenCV**, 2019. Disponivel em: <http://www.opencv.org/>. Acesso em: mar. 2019.

PORTO, S. M. C. et al. An automatic system for the detection of dairy cows lying behaviour in free-stall barns. **Journal of Agricultural Engineering** , Catania, Italia, v. XLIV, 2013.

PYINSTALLER.ORG. PyInstaller: Welcome to PyInstaller official website. **PyInstaller**, 2019. Disponivel em: <https://www.pyinstaller.org>. Acesso em: mar. 2019.

PYTHON.ORG. Python 3.6.8 documentation. **Python Documentation**, 2019. Disponivel em: <https://docs.python.org/3.6/>. Acesso em: fev. 2019.

ROSSUM, V. Personal History. **The History of Python**, 2009. Disponivel em: <http://python-history.blogspot.com.br/2009/01/personal-history-part-1-cwi.html>. Acesso em: nov. 2016.

RURAL, G. Globo Rural. **globoplay**, 21 dez 2014. Disponivel em: <https://globoplay.globo.com/v/3844834/>. Acesso em: 24 out 2019.

RUSSELL, J.; LAAN, M. About: Inno Setup. **Jordan Russell's Software**, 2018. Disponivel em: <http://www.jrsoftware.org/>. Acesso em: jan. 2019.

SANTANAS, L. M. Q.; SANTOS, T. S. R.; GOMES, F. R. Processo de Detecção Facial, utilizando Viola-Jones. **Interfaces Científicas**, Aracaju, v. 1, n. 1, p. 35-40, fev 2015. ISSN 2359-4948.

SILVA, R. R. **Reconhecimento de imagens digitais utilizando Redes Neurais Artificiais**. Lavras-MG: Departamento de Ciência da Computação da Universidade de Lavras, 2005.

SUPER INTERESSANTE. Ciência: A ótica do cérebro: neurônios e eletricidade. **Super Interessante**, 2016. Disponivel em: <https://super.abril.com.br/ciencia/a-otica-do-cerebro-neuronios-e-eletricidade/>. Acesso em: jan. 2017.

VIOLA, P.; JONES, M. Rapid Object Detection using a Boosted Cascade of Simple Features. **Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition**, Kauai, HI, USA, dez. 2001.

WXPYTHON.ORG. wxPython: About. **wxPython - The GUI Toolkit for Python**, 2019. Disponivel em: <https://www.wxpython.org>. Acesso em: jan. 2019.

ANEXO 1: DADOS COMPARATIVOS ENTRE OS DETECTORES

Imagem	Ideal			Tons Cinza 5 Eras			Tons de Cinza 10 Eras			Histograma 5 Eras			Histograma 10 Eras		
	VP	FP	FN	VP	FP	FN	VP	FP	FN	VP	FP	FN	VP	FP	FN
teste_1	0	0	0	0	4	0	0	2	0	0	6	0	0	2	0
teste_2	1	0	0	1	1	0	1	0	0	0	3	1	1	0	0
teste_3	1	0	0	1	5	0	0	1	1	0	2	1	0	2	1
teste_4	1	0	0	0	2	0	1	1	0	0	3	1	1	0	0
teste_5	1	0	0	0	1	1	0	1	1	0	1	1	0	1	1
teste_6	3	0	0	1	1	2	3	0	0	1	0	2	3	2	0
teste_7	0	0	0	0	4	0	0	2	0	0	4	0	0	2	0
teste_8	1	0	0	0	1	1	1	2	0	0	2	1	1	1	0
teste_9	3	0	0	1	0	2	2	0	1	1	0	2	3	0	0
teste_10	1	0	0	1	4	0	1	1	0	0	3	1	1	1	0
teste_11	1	0	0	0	1	0	1	1	0	0	3	1	0	1	1
teste_12	1	0	0	0	4	0	1	3	0	0	2	1	1	4	0
teste_13	2	0	0	1	1	1	2	0	0	1	1	1	2	1	0
teste_14	2	0	0	1	0	1	2	2	0	1	0	1	1	4	1
teste_15	1	0	0	0	2	1	1	1	0	0	5	1	1	0	0
teste_16	2	0	0	1	2	1	2	0	0	1	1	1	1	2	1
teste_17	1	0	0	1	4	0	1	2	0	1	3	0	1	2	0
teste_18	0	0	0	0	2	0	0	1	0	0	2	0	0	1	0
teste_19	0	0	0	0	3	0	0	1	0	0	4	0	0	0	0
teste_20	1	0	0	0	2	1	1	4	0	1	2	0	1	3	0
teste_21	1	0	0	1	3	0	1	0	0	1	3	0	1	0	0
teste_22	0	0	0	0	3	0	0	2	0	0	3	0	0	2	0
teste_23	3	0	0	1	5	2	2	0	1	1	4	2	2	0	1
teste_24	1	0	0	0	4	1	1	0	0	1	0	0	1	1	0
teste_25	2	0	0	2	1	0	2	1	0	2	2	0	2	2	0
teste_26	1	0	0	0	1	1	1	3	0	0	1	1	0	2	1
teste_27	1	0	0	0	3	1	1	2	0	1	2	0	0	3	1
teste_28	1	0	0	0	1	1	1	5	0	0	1	1	1	3	0
teste_29	1	0	0	0	2	1	0	2	1	0	3	1	1	1	0
teste_30	2	0	0	0	2	2	2	0	0	0	2	2	2	0	0
TOTAL	**36**	**0**	**0**	**13**	**69**	**20**	**31**	**40**	**5**	**13**	**68**	**23**	**28**	**43**	**8**

Nota: (VP) Verdadeiro Positivo; (FP) Falso Positivo; (FN) Falso Negativo

APÊNDICE A: INSTALAÇÃO DAS FERRAMENTAS

Neste apêndice será demonstrado todo o processo de instalação e configuração dos produtos utilizados para o desenvolvimento deste trabalho.

1. INSTALAÇÃO DO PYTHON 3 (64-BIT)

Após baixar o instalador do Python 3.6.6 disponível gratuitamente no próprio site (`https://www.python.org`), será exibida sua tela inicial de instalação conforme mostrado na Figura 1.1. Nesta tela, marque a opção *`Add Python 3.6 to PATH`*, para que o mesmo adicione uma variável de ambiente no usuário do sistema durante o processo de instalação, o que permitirá a utilização de códigos `Python` no Prompt de Comando do Windows.

Após marcado a opção `Add Python 3.6 to PATH`, basta clicar em *`Install Now`*, conforme mostra a Figura 1.1.

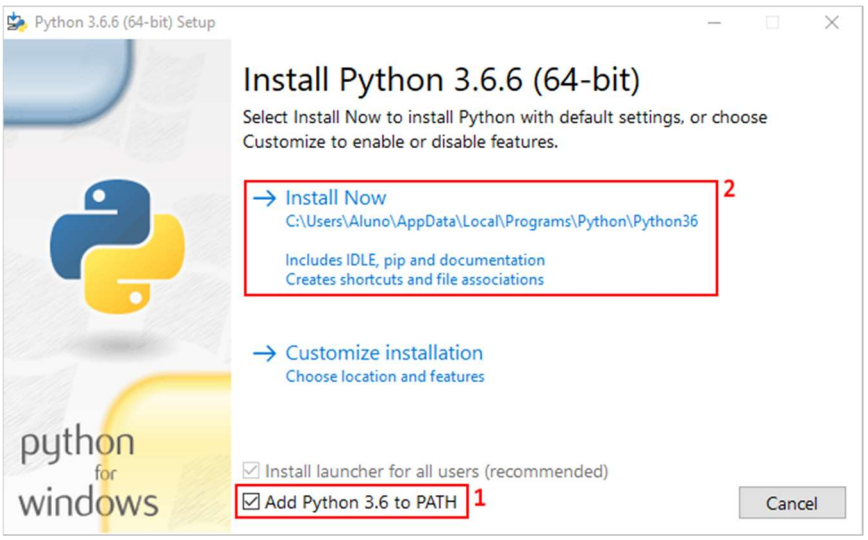


Figura 1.1 - Instalação do Python: Tela inicial de Instalação

Após clicar em `Install Now`, será dado início ao processo de instalação do Python 3.6.6 (64-bit) conforme mostra a Figura 1.2.

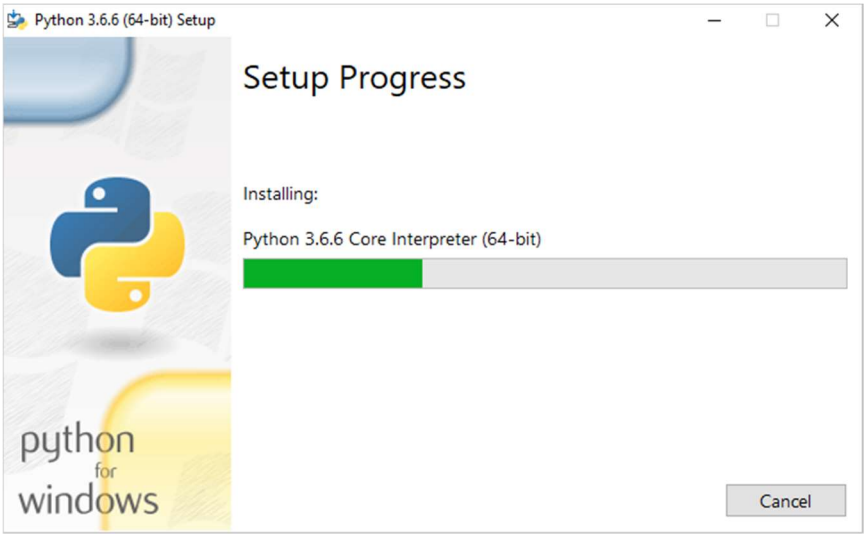


Figura 1.2 - Instalação do Python: *Setup Progress*

Ao concluir o processo de instalação, será mostrado uma tela informando que a instalação foi realizada com sucesso, conforme mostra a. Para finalizar, basta clicar no botão `Close`.

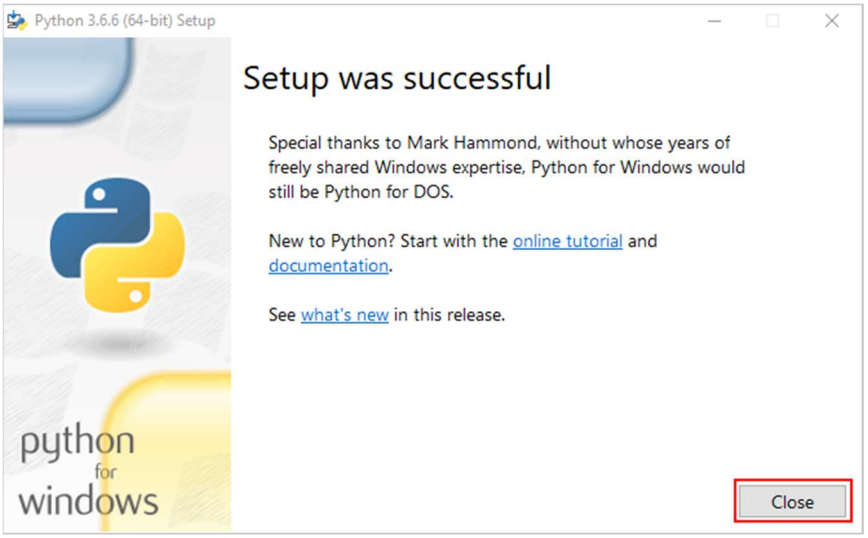


Figura 1.3 - Instalação do Python: Finalização da Instalação

Vale ressaltar que, com o Python, serão adicionados o IDLE (*Integrated Development Environment* ou *Integrated Development and Learning Environment*), o `pip` (comando que acessa o gerenciador de pacotes do Python) e a documentação da linguagem, além de realizar a associação dos arquivos relacionados ao Python (com a extensão .py ou .pyc)

1.1 ATUALIZAÇÃO DO PIP

Após concluir a instalação do Python, o próximo passo será a atualização do *pip*. O *pip* é um gerenciador de pacotes criado pela comunidade de desenvolvedores Python e disponibilizado a partir do Python 3.4 (PYTHON.ORG, 2019). Com ele, é possível gerenciar módulos e pacotes específicos do Python encontrados no PyPI (*Python Package Index*), um repositório que contém módulos, pacotes e bibliotecas do Python.

Para atualizar o *pip*, abra o Prompt de Comando do Windows como administrador, digite o comando `python -m pip install --upgrade pip` e pressione a tecla `Enter`. Com isso, o Python verificará se há alguma atualização para o *pip* e, caso haja, fará o *download* e instalação do mesmo, conforme mostrado na Figura 1.4. Vale ressaltar que, para fazer uso do comando *pip*, é necessário que o computador possua conexão com a internet.

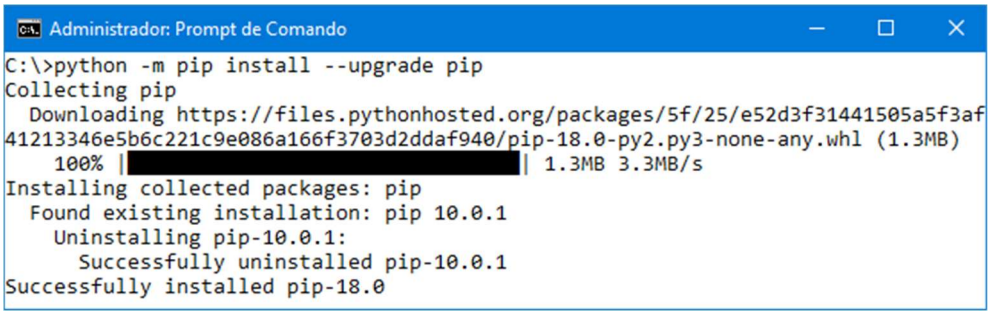


Figura 1.4 - Atualização do pip

Caso o *pip* já esteja em sua última versão (versão mais recente), será exibida a mensagem conforme mostrado na Figura 1.5.

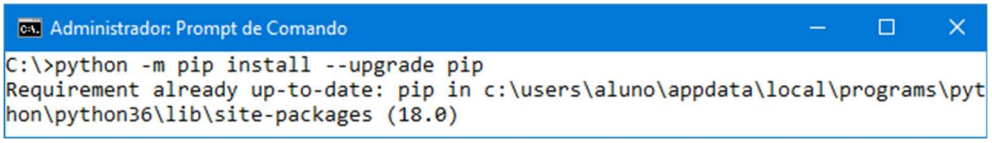


Figura 1.5 - Pip atualizado

2. NUMPY, PILLOW, WXPYTHON E PYINSTALLER

Com o Prompt de Comando do Windows aberto, digite o comando `pip install numpy==1.14.5`. Com este comando, será feito uma busca no repositório do PyPI e em seguida, realizará o processo de *download* e instalação do NumPy na versão 1.14.5 conforme mostra a Figura 2.1. Vale ressaltar que, para realizar esta instalação a partir do *pip*, é necessário possuir uma conexão com a internet.

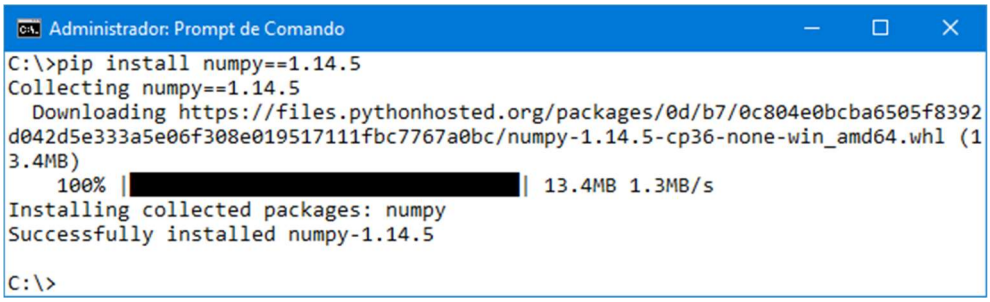


Figura 2.1 - Instalação do Numpy

Concluída a instalação do NumPy, a próxima biblioteca a instalar será a Pillow. Para isso, com o Prompt de Comando do Windows aberto, digite o comando `pip install Pillow==5.1.0` (Figura 2.2).

Figura 2.2 - Instalação do Pillow

Com a biblioteca Pillow instalada, a próxima será a biblioteca wxPython e para isso, digite o comando `pip install wxPython==4.0.3` no Prompt de Comando do Windows e pressione Enter, conforme mostra a Figura 2.3.

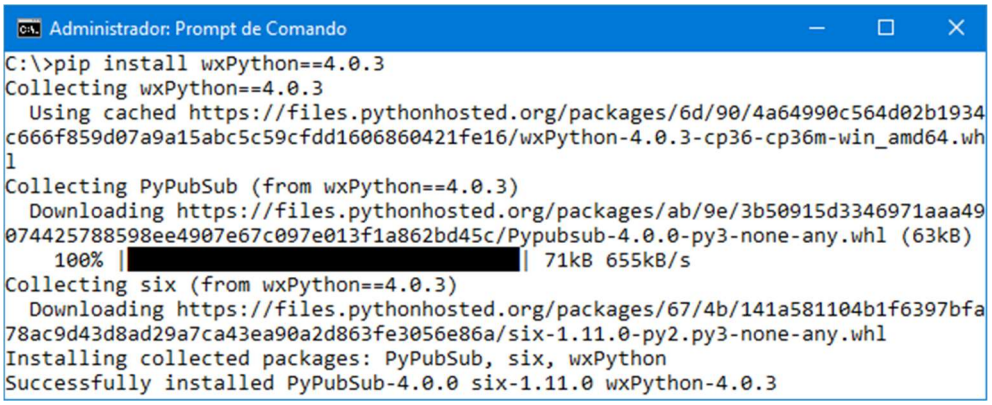


Figura 2.3 - Instalação do wxPython

Já para a instalação do PyInstaller, digite o comando `pip install PyInstaller==3.3.1` no Prompt de Comando do Windows (Figura 2.4).

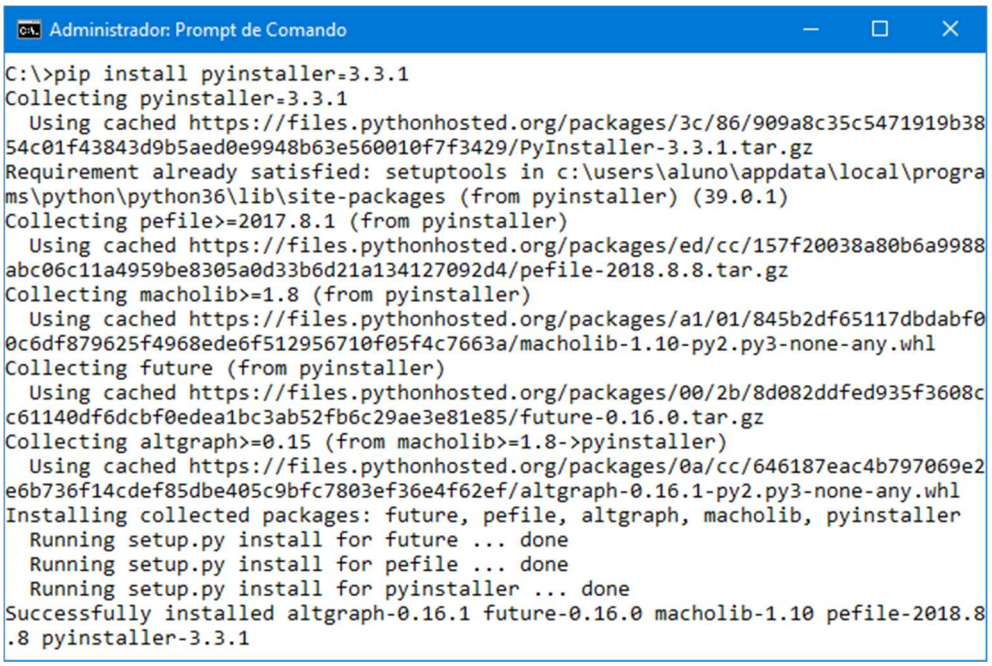


Figura 2.4 - Instalação do PyInstaller

3. INSTALAÇÃO DA OPENCV

O processo de instalação da biblioteca OpenCV diferencia-se parcialmente das demais bibliotecas pelo fato de adicionar o *download* da mesma no site da biblioteca para fazer uso de alguns arquivos já configurados. Para isso, realize o download da biblioteca versão 3.4.0 no site oficial[1] e, em seguida, realize o processo de extração dos arquivos (Figura 3.1).

Dentro do diretório extraído, encontram-se os arquivos `opencv_createsamples.exe`, `opencv_traincascade.exe` e as DLLs `opencv_ffmpeg310_64.dll`, `opencv_world310.dll` e `opencv_world310d.dll`. Estes arquivos serão utilizados para a realização do treinamento e os mesmos foram copiados para `sample/bin` dentro do diretório do projeto.

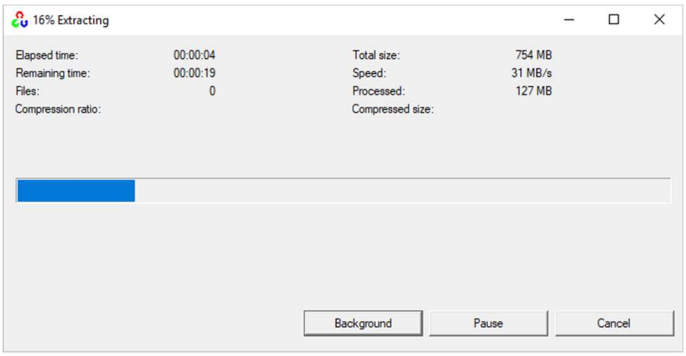


Figura 3.1 - Extração dos arquivos da OpenCV

Após realizar a extração dos arquivos (Figura 3.1), abra o Prompt de Comando do Windows no modo administrativo e realize a instalação da biblioteca a partir do comando `pip install opencv-python==3.4.0.14`, conforme mostra a Figura 3.2.

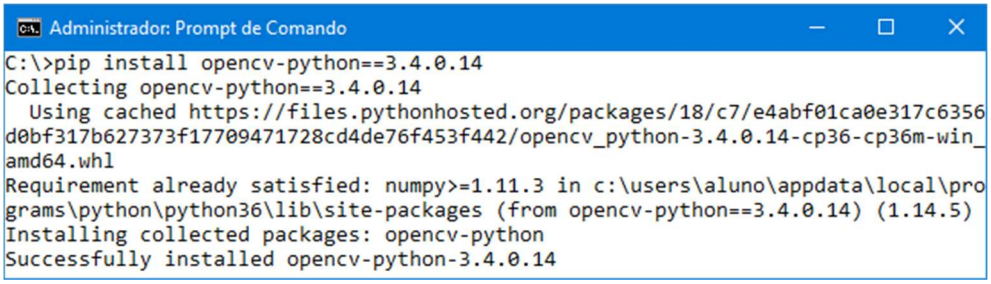


Figura 3.2 - Instalação da OpenCV via Prompt de Comando

[1] https://www.opencv.org

4. VERIFICAÇÃO DAS INSTALAÇÕES

Após realizadas as instalações e atualizações, é interessante realizar a verificação da integridade das instalações. Para isso, com o Prompt de Comando do Windows aberto, digite o comando `pip list` e pressionar a tecla `Enter`. Dessa forma, será demonstrado uma lista contendo todas as bibliotecas e dependências instaladas no Python.

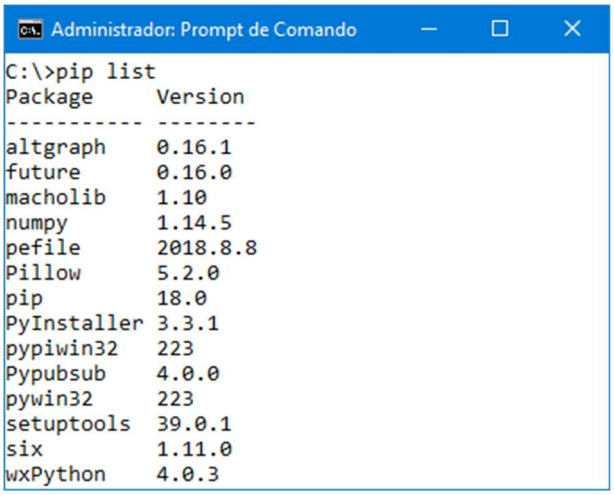


Figura 4.1 - Bibliotecas instaladas no Python

5. INSTALAÇÃO DO ECLIPSE 4.8 (OXYGEN)

Após realizar o *download* do Eclipse 4.8 (*Oxygen*) disponível no próprio site do Eclipse (`http:\\www.eclipse.org`) e iniciado o processo de instalação, deve-se selecionar uma das opções disponíveis pelo Eclipse. Estas opções estão relacionadas com as configurações do ambiente de desenvolvimento. Como o Eclipse não possui um ambiente específico para desenvolvimento em Python, selecionou-se a opção `Eclipse IDE for Java EE Developers`.

Figura 5.1 - Seleção do ambiente de desenvolvimento para o Eclipse

Neste momento pode mudar ou manter o caminho padrão para instalação do Eclipse (Figura 5.2). Após a escolha do caminho de instalação, clique no botão `INSTALL` (Figura 5.2) que o processo de instalação será inicializado e, quando finalizado, clique no botão `LAUNCH` para abri-lo (Figura 5.3).

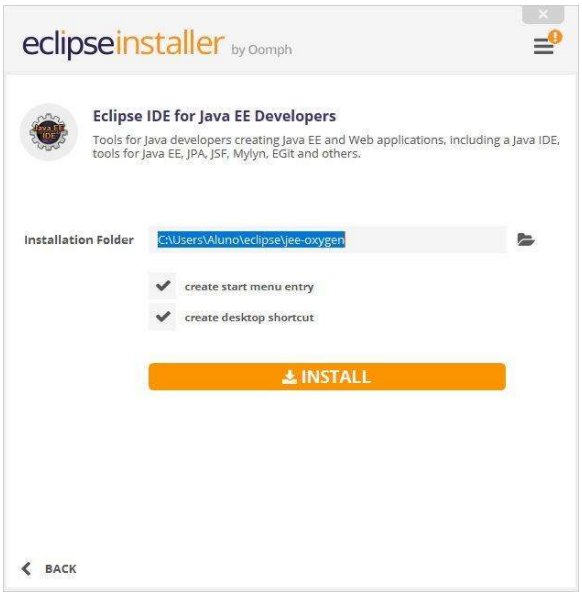


Figura 5.2 - Tela de instalação do *Eclipse IDE for Java EE Developers*

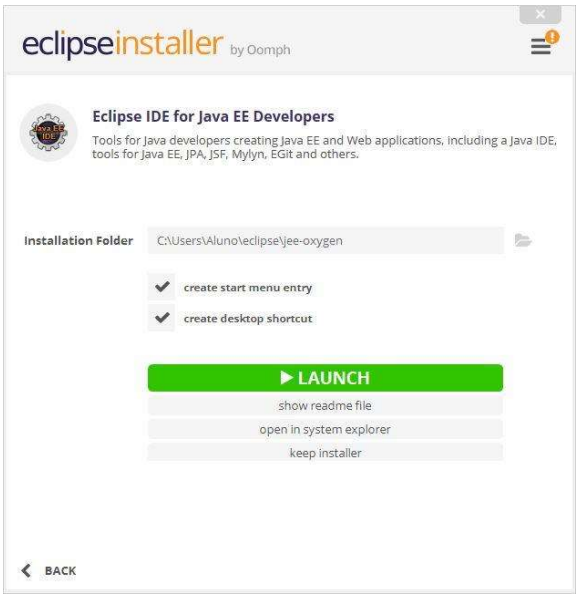


Figura 5.3 - Tela finalização da instalação do *Eclipse IDE for Java EE Developers*

5.1 INSTALAÇÃO DO PYDEV

Para fazer a instalação e configuração do Eclipse, com a IDE aberta, acesse o menu `Help > Eclipse Marketplace`.

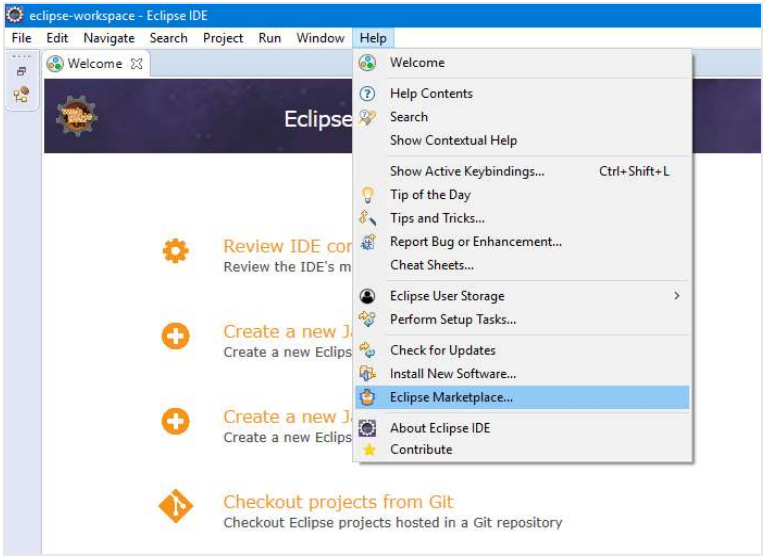


Figura 5.4 - Eclipse *Marketplace*

Com o Eclipse Marketplace aberto, pesquise por `PyDev` e, assim que localizado, pressione o botão `Install`.

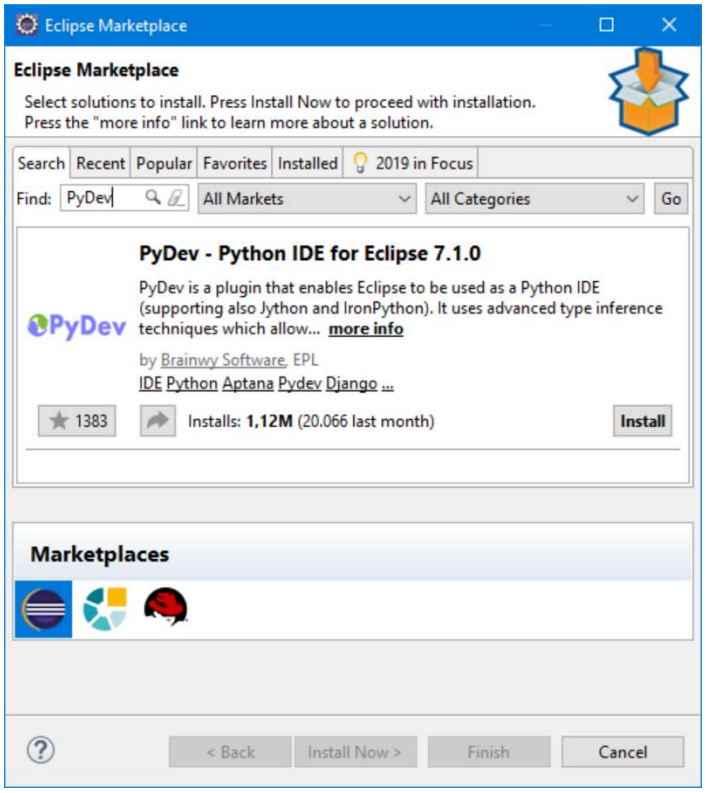


Figura 5.5 - Eclipse *Marketplace*: PyDev

Após pressionar o botão `Install`, será mostrado os arquivos de instalação. Sendo assim, basta selecioná-los e pressionar o botão `Confirm`.

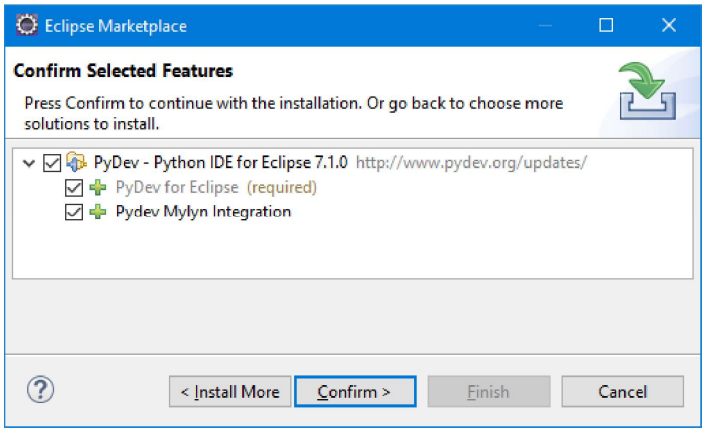


Figura 5.6 - Eclipse *Marketplace*: Confirm Features

Após pressionar o botão `Confirm`, será exibido os termos de licença de uso. Selecione a opção `I accept`... e pressione o botão `Finish` que o PyDev será instalado.

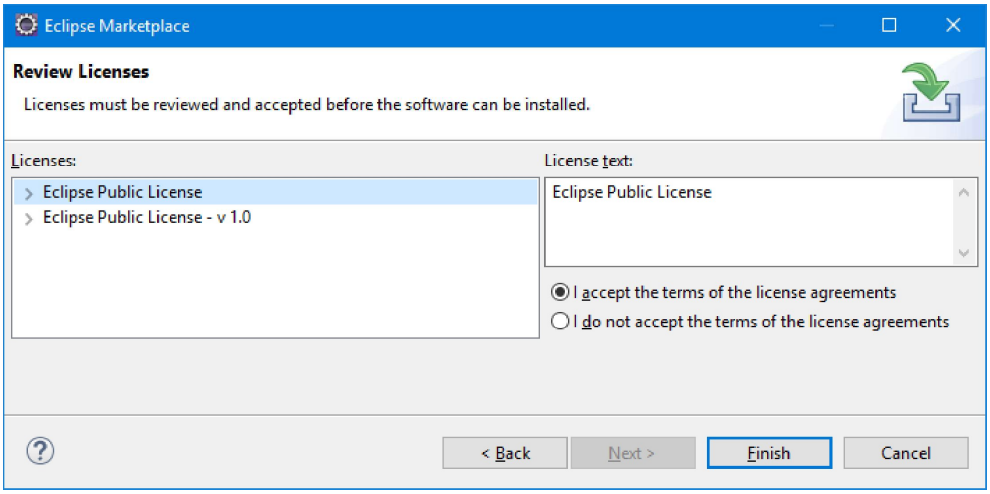


Figura 5.7 - Eclipse *Marketplace*: Licença e instalação do PyDev

Quando o processo de instalação for concluído, será solicitado que seja reiniciado o Eclipse.

5.2 CONFIGURAÇÃO DO INTERPRETADOR PYTHON

Mesmo com o *Pydev* instalado, existe a necessidade de informar a localização Python dentro do sistema. Sendo assim, ao iniciar o Eclipse, clique no menu `Window > Preferences` (Figura 5.8).

Figura 5.8 - Tela *Workspace* do Eclipse

Ao acessar a tela de `Preferences`, na extremidade a direita, selecione `Pydev > Interpreters > Python Interpreters` (Figura 5.9).

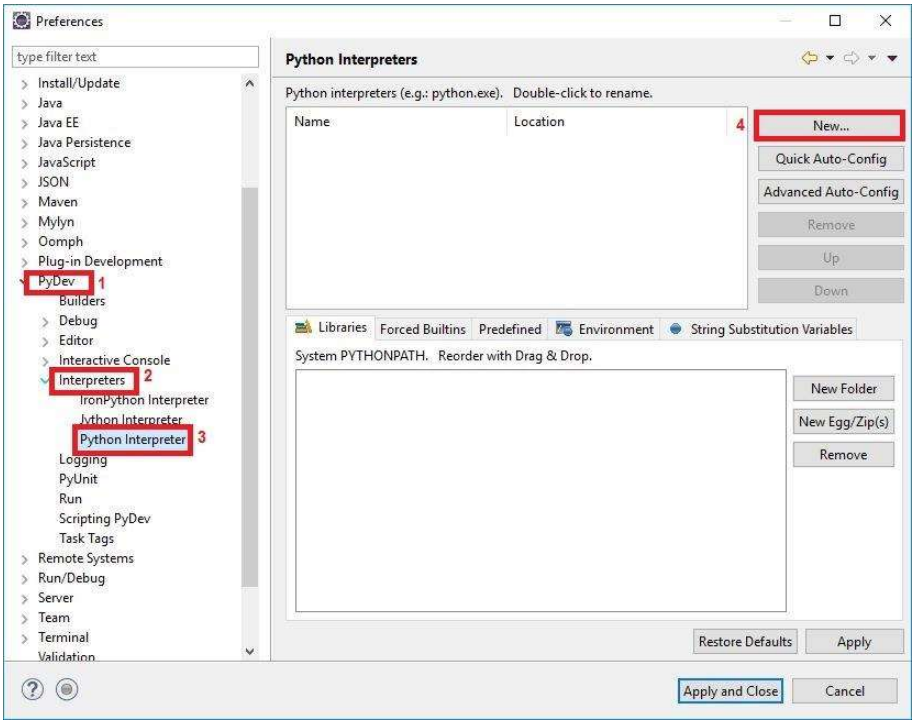


Figura 5.9 - Tela *Python Interpreters*

Na tela Python `Interpreters`, clique no botão `New` (Figura 5.9). Ao clicar, será solicitado, na tela `Select Interpreters`, informar o caminho de onde o Python foi instalado. Para isso, clique no botão `Browse` (Figura 5.10).

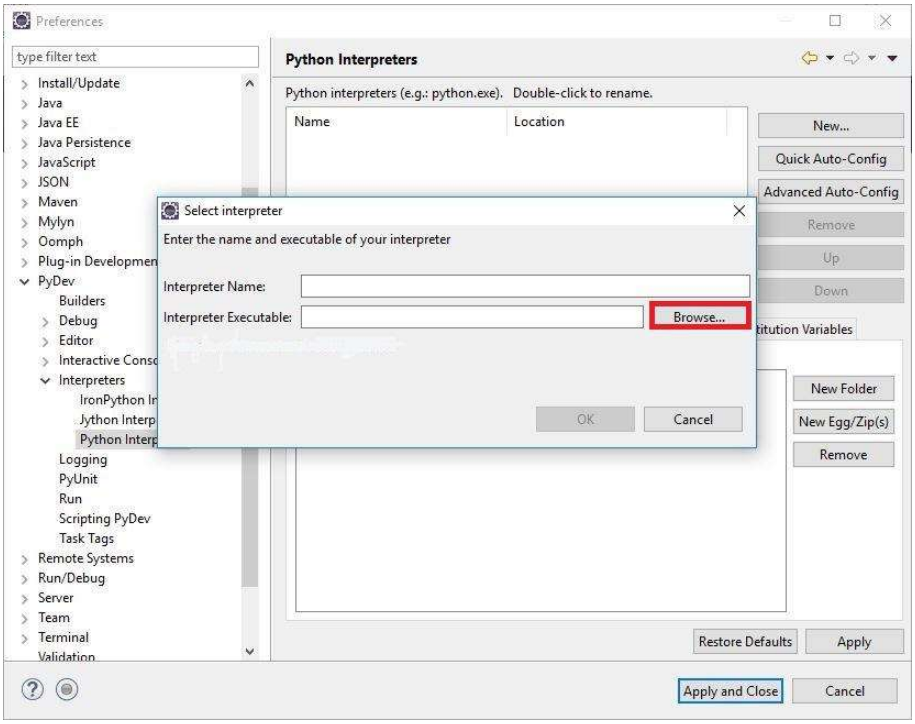


Figura 5.10 - Tela "*Select Interpreter*"

Dando continuidade ao processo, a localização do Python.exe deve ser passada corretamente como no exemplo da Figura 5.11. Vale ressaltar que, por padrão, o Python está instalado em `C:\User\Aluno\AppData\Local\Programs\Python`[1], e que a pasta `AppData` está oculta.

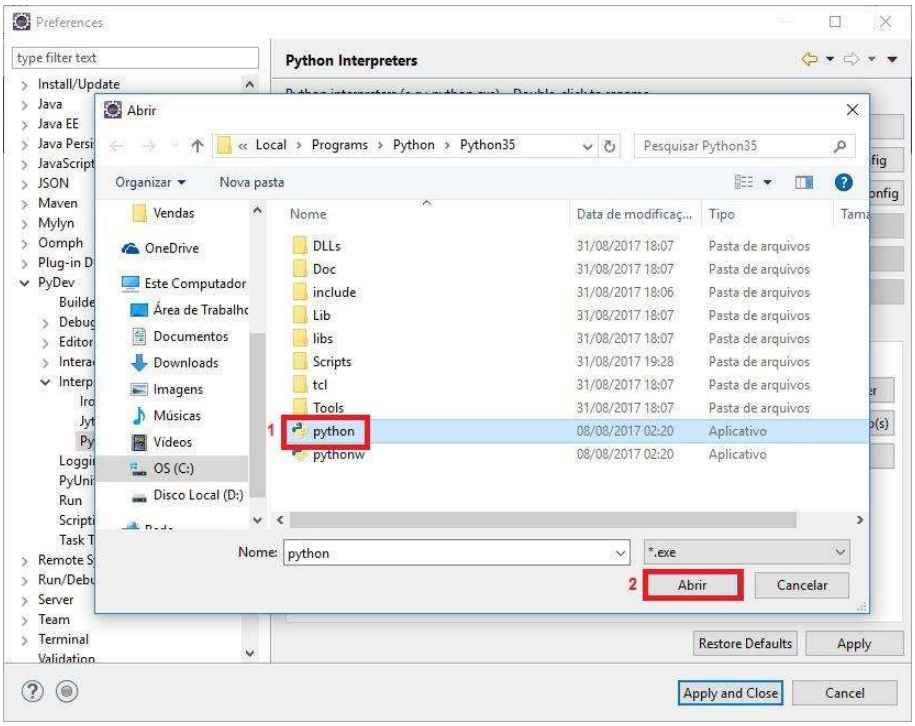


Figura 5.11 - Localização do Python.exe

Após localizar o arquivo `Python.exe`, selecione-o e clique no botão `Abrir`. Com isso, retornando para a tela `Select Interpreter`, tanto o `Interpreter Name` e o `Interpreter Executable` estarão com as informações necessárias. Sendo assim, para dar prosseguimento na configuração, clique no botão `Ok` (Figura 5.12).

[1] Neste endereço, a palavra Aluno refere-se ao usuário Aluno. Isso modifica de acordo com o usuário registrado para uso no sistema operacional.

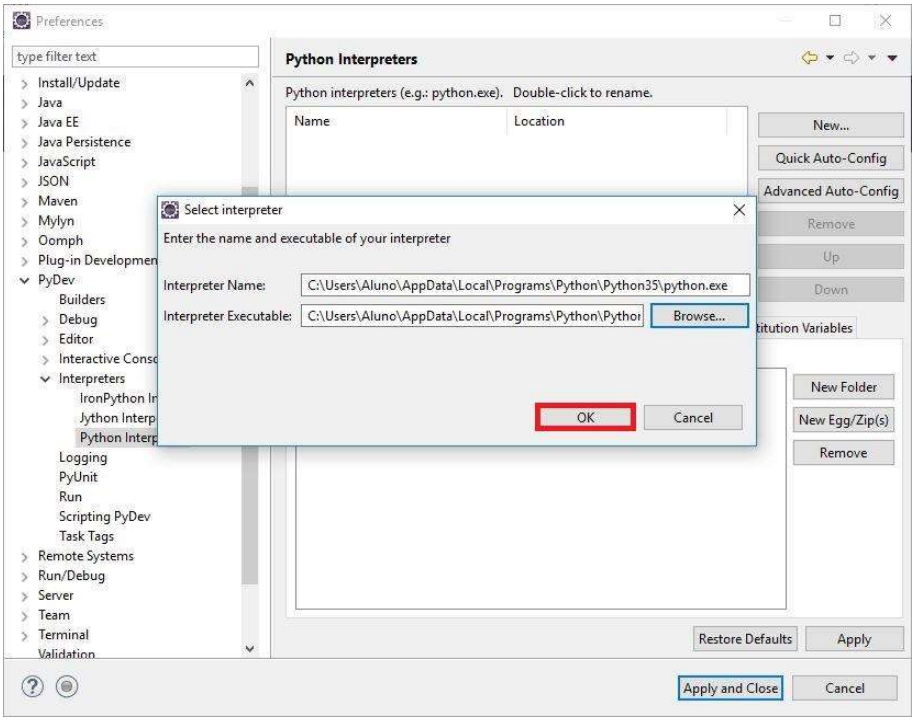


Figura 5.12 - Tela "*Select interpreter*"

Dando continuidade, na tela `Selection Needed` (Figura 5.13), será exibido os diretórios que deverão compor o *pythonpath*. Após selecioná-los clique no botão `Ok` (Figura 5.13).

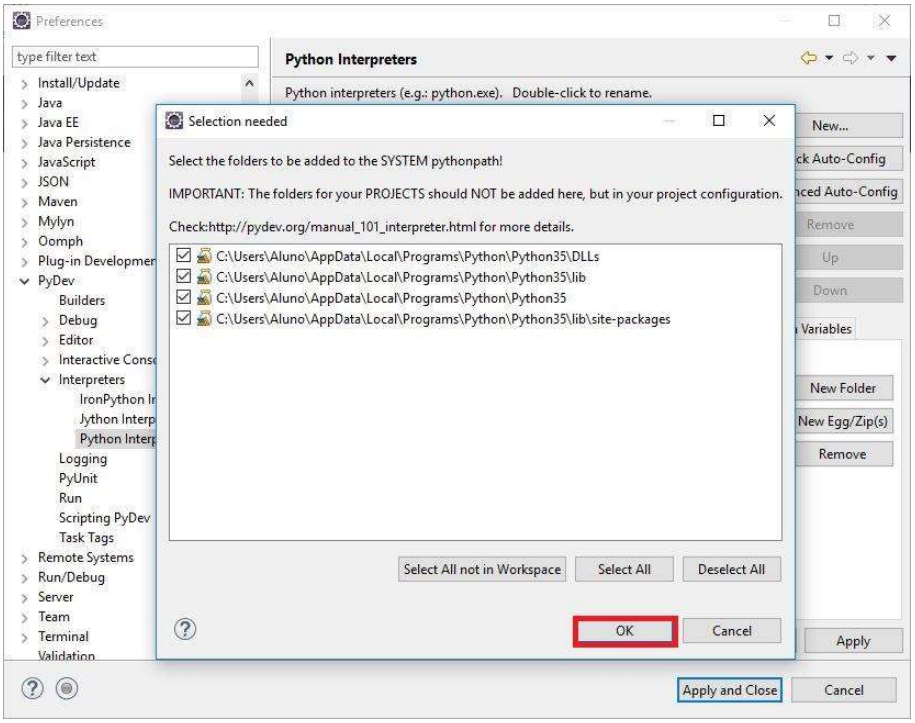


Figura 5.13 - Tela "*Selection Needed*"

Para finalizar o processo de configuração do interpretador Python, clique no botão `Apply and Close` (Figura 5.14).

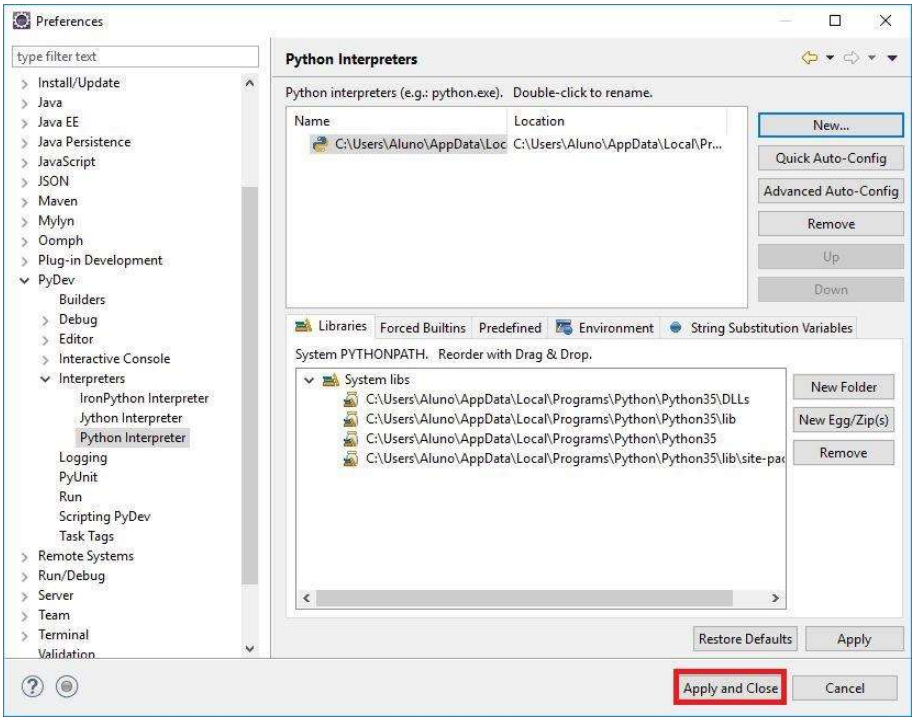


Figura 5.14 - *Apply and Close*: Tela "*Python Interpreters*"

Desta forma, todo o ambiente de desenvolvimento está configurado para dar início ao trabalho com Python e OpenCV através do Eclipse IDE.

6. INSTALAÇÃO DO INNO SETUP 5 (OPCIONAL)

Para realizar a instalação do Inno Setup 5, após baixar e executar o instalador, será pedido que selecione o idioma que será usado durante o processo de instalação. Dessa forma, na caixa de opções, selecione a opção Português Brasileiro conforme mostra a Figura 6.1, e pressione o botão OK.

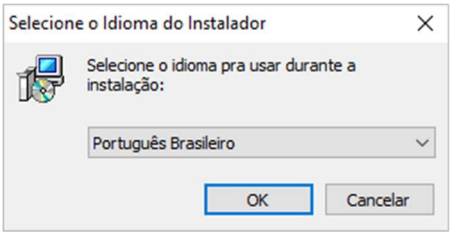


Figura 6.1 - Instalação do Inno Setup: Idioma de Instalação

Após selecionar o idioma de instalação e pressionar o botão OK, será mostrado o Acordo de Licença de uso do software, leia-o, selecione a opção "*Eu aceito o acordo*" e pressione o botão Próximo (Figura 6.2).

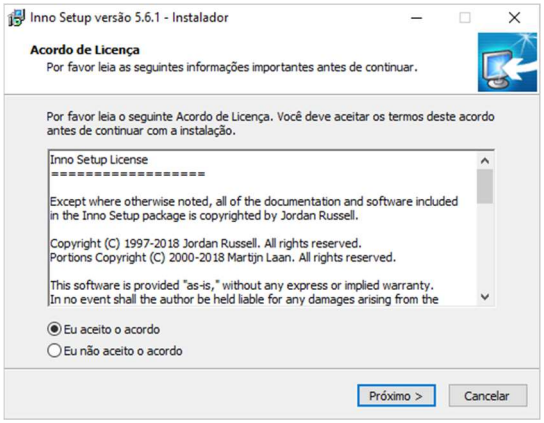


Figura 6.2 - Instalação do Inno Setup: Termo de Licença

Na próxima janela, selecione a opção "*Install Inno Setup Preprocessor*" e pressione o botão Próximo (Figura 6.3).

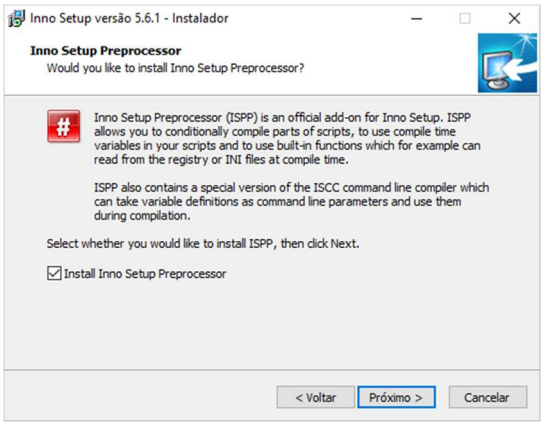


Figura 6.3 - Instalação do Inno Setup: Inno Setup *Preprocessor*

Caso deseje, é possível criar um ícone na área de trabalho e/ou associar o programa com os arquivos de extensão `iss`, sendo esta última opção recomendada. Dessa forma, selecione a ou as opções e pressione o botão Próximo (Figura 6.4).

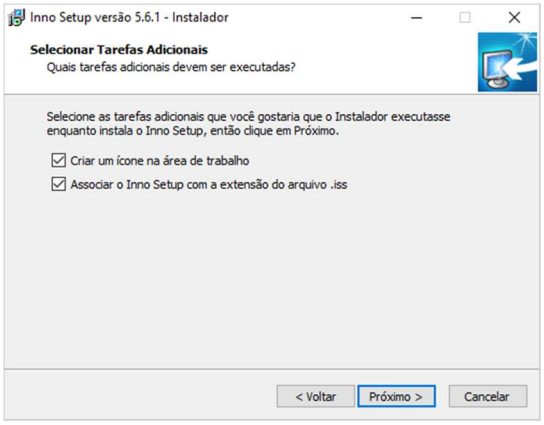


Figura 6.4 - Instalação do Inno Setup: Ícone e Associação de Arquivos

Ao clicar no botão Próximo, será exibida a tela com as informações relacionadas às opções selecionadas anteriormente, como a criação do ícone e a associação de arquivos conforme mostra a Figura 6.5.

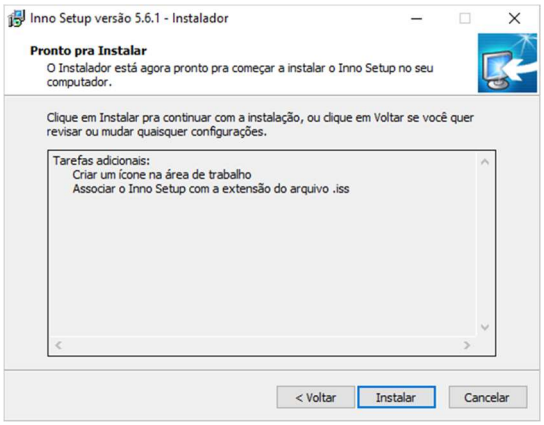


Figura 6.5 - Instalação do Inno Setup: Pronto para Instalar

Assim que clicar no botão Instalar, será dado início ao processo de extração e instalação do Inno Setup 5 (Figura 6.6).

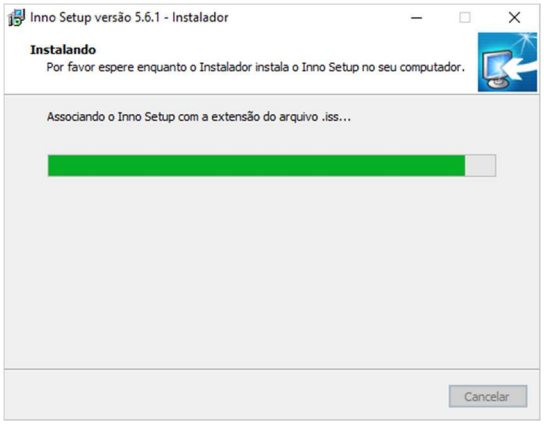


Figura 6.6 - Instalação do Inno Setup: Instalação

Quando concluído, basta clicar no botão Concluir (Figura 6.7). Caso deseje iniciar o Inno Setup 5, marque a opção "*Iniciar o Inno Setup*".

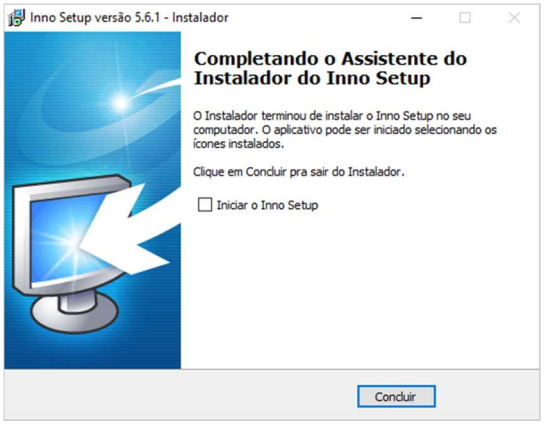


Figura 6.7 - Instalação do Inno Setup: Finalizada

APÊNDICE B: MANUAL DO USUÁRIO

O Gerenciador de Projetos de Detecção por Visão Computacional (GPDVC) é um sistema capaz de, a partir do momento em que o usuário possui um banco de imagens referentes a um objeto específico, criar um arquivo XML com as descrições de tal objeto, o que possibilita ao usuário, detectar tal objeto em imagens ou vídeos a partir do uso do arquivo XML.

Este sistema foi desenvolvido em Python, juntamente com a biblioteca OpenCV. Vale ressaltar que o processo de instalação de tal sistema pode ser realizado a partir do arquivo executável compatível apenas com Windows 8.1 ou Windows 10.

O GPDVC encontra-se em sua primeira versão beta (1.0.0.0), o qual pode apresentar instabilidades, principalmente relacionadas aos testes com vídeos.

1. INSTALAÇÃO DO GERENCIADOR DE PROJETOS DE DETECÇÃO POR VISÃO COMPUTACIONAL (GPDVC)

Para realizar a instalação do Gerenciador de Projetos de Detecção por Visão Computacional (GPDVC), siga os passos abaixo:

1. Baixe o instalador disponível em http://bit.ly/2L4j1Al;

2. Assim que concluído o *download*, dê um clique duplo sobre o arquivo;

3. Após o clique duplo será mostrado a janela inicial de GPDVC;

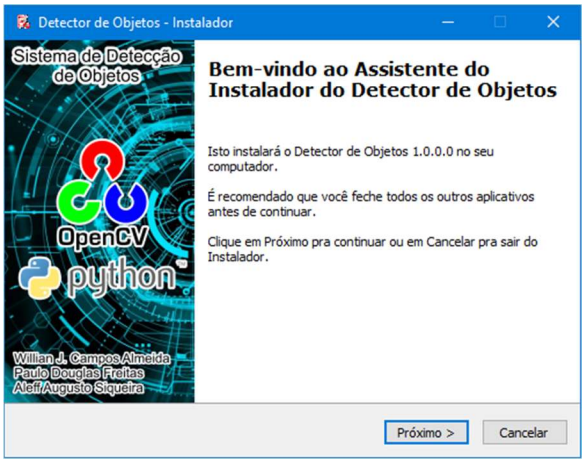


Figura 1.1 - Janela inicial de Instalação do GPDVC

4. Após pressionar o botão “Próximo” na janela inicial, será exibida a janela relacionada ao termo de licença de uso do sistema. Leia-o atentamente, marque a opção “Eu aceito o acordo” e clique no botão “Próximo”;

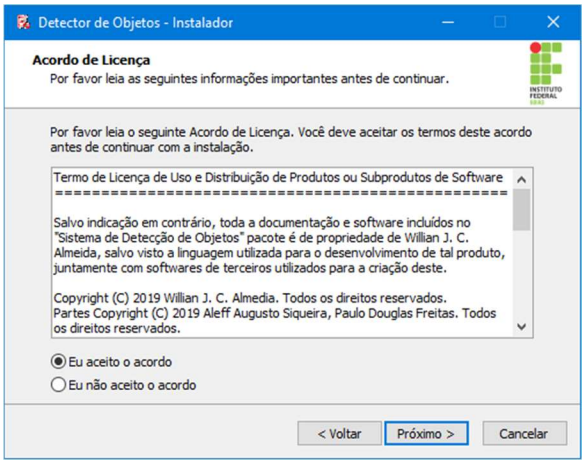


Figura 1.2 - Instalação do GPDVC: Acordo de Licença

5. Após pressionar o botão "Próximo", será exibida algumas informações acerca do sistema desenvolvido. Leia-as e clique no botão "Próximo";

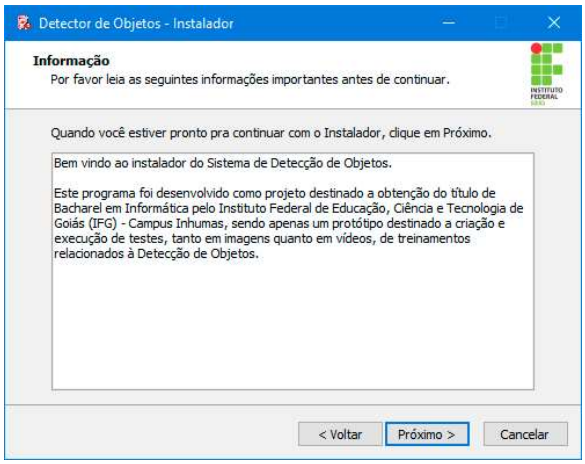


Figura 1.3 - Instalação do GPDVC: Informações

6. Na próxima janela, possibilita que, após o processo de instalação, seja criado um ícone na área de trabalho. Marcar ou não, não influenciará no funcionamento do GPDVC. Assim que concluído, clique no botão "Próximo";

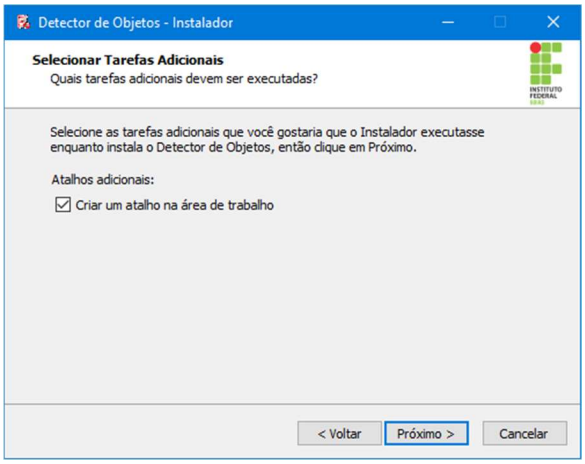


Figura 1.4 - Instalação do GPDVC: Ícone na área de trabalho

7. Em seguida, o sistema exibirá as informações resumidas quanto a instalação. Nesta janela, basta clicar no botão “Instalar”;

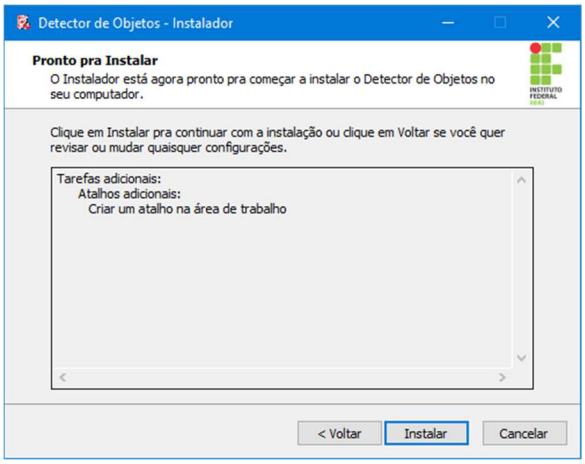


Figura 1.5 - Instalação do GPDVC: Revisão

8. Após clicar no botão “Instalar” será dado início ao processo de instalação do GPDVC;

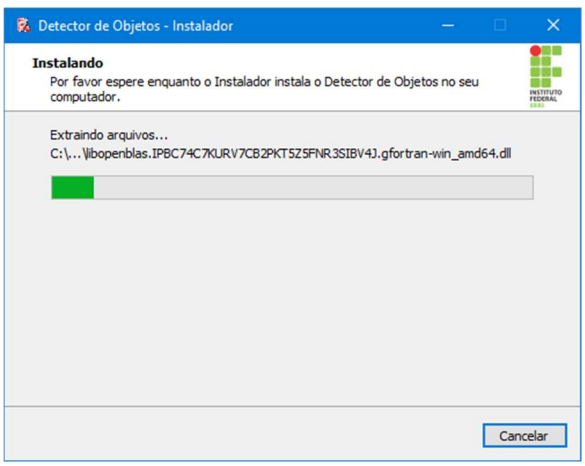


Figura 1.6 - Instalação do GPDVC: Extraindo arquivos

9. Assim que concluído o processo de instalação, será exibida algumas informações acerca do produto;

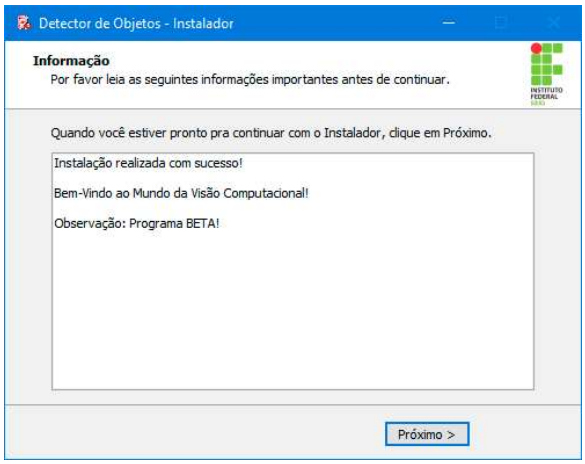


Figura 1.7 - Janela de Informação

10. Ao concluir o processo de instalação, será exibida a janela de finalização. Clique no botão "Concluir". Caso tenha marcado a opção "Iniciar o Detector de Objetos", a tela inicial do sistema será iniciada (Figura 2.1).

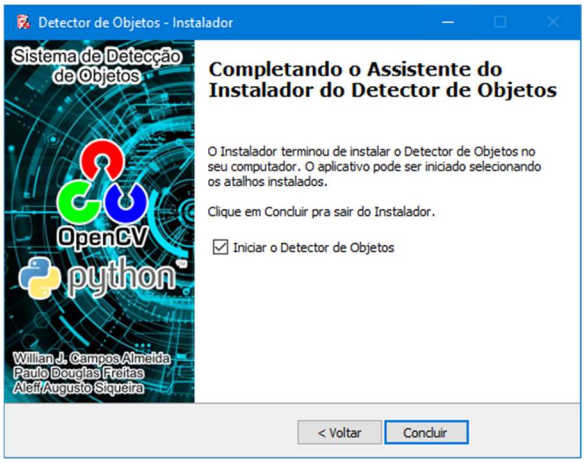


Figura 1.8 - Instalação do GPDVC: Concluído

Vale ressaltar que, todos os arquivos serão instalados/descompactados por padrão no diretório `C:/Detector-Objetos`.

2. MENU PRINCIPAL

Assim que iniciado o sistema, será exibida a seguinte janela:

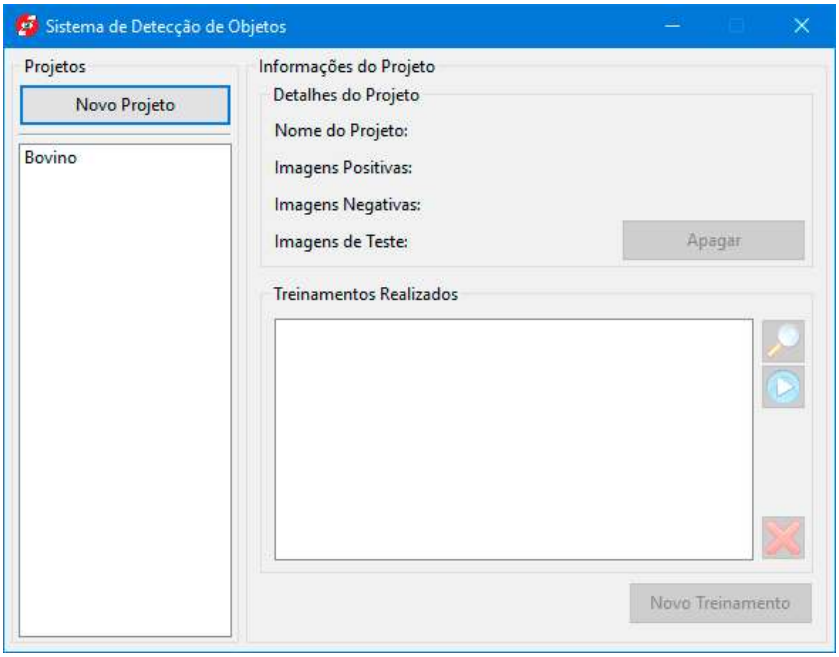


Figura 2.1 - Janela Principal do GPDVC

Nesta janela, estão esquematizadas algumas áreas que possuem e que listarão algumas informações. Na área “Projetos”, estão disponíveis o botão “Novo Projeto”, destinado para a criação de um projeto (vide tópico 3. Novo Projeto, cito a partir da página 70) e a lista dos projetos criados, conforme pode ser visto na Figura 2.2.

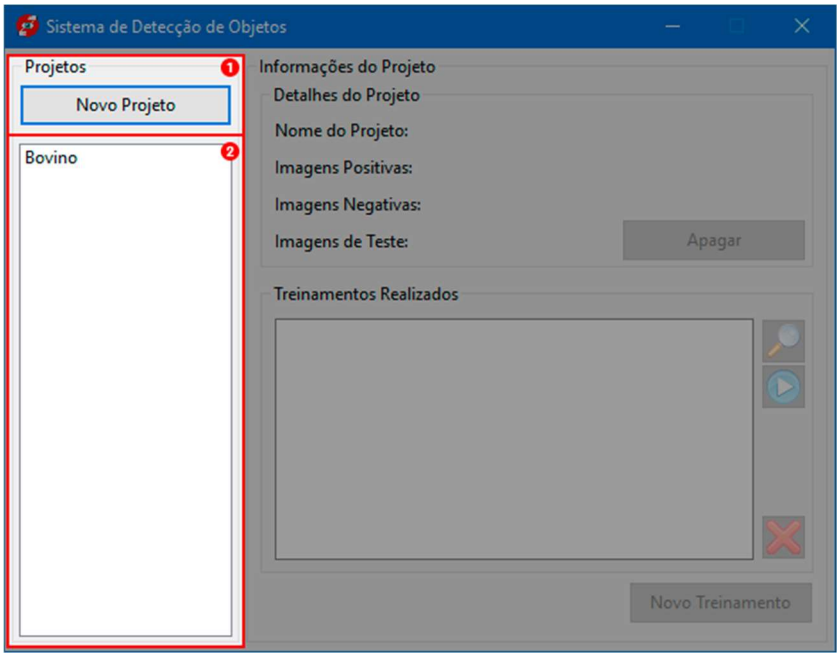


Figura 2.2 - Janela principal do GPDVC: Área de Projetos
Na área 1 exibe o botão "Novo Projeto", destinado para a criação de um novo projeto; na área 2, é exibido o(s) projeto(s) criado. Caso não haja projeto, esta área será exibida em branco.

Assim que selecionado um projeto, serão exibidas outras informações, como a quantidade de imagens positivas e negativas o projeto possui (área "Informações do Projeto") e os arquivos referente a treinamentos (área "Treinamentos Realizados"), conforme mostrado na Figura 2.4.

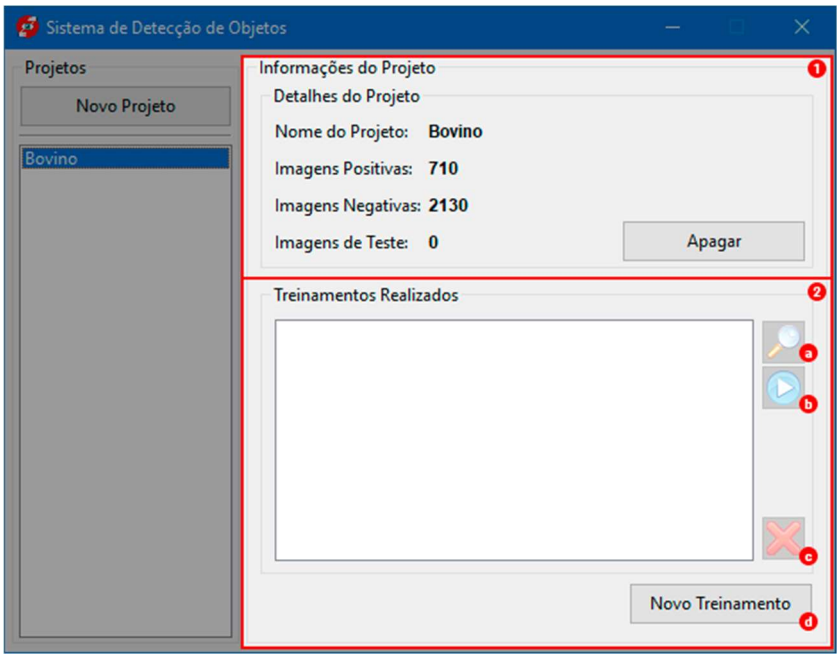


Figura 2.3 - Janela principal do Sistema: Área de Informações
Na área 1 exibe as informações relacionadas aos detalhes do projeto, como o nome do projeto, quantidade de imagens positivas, negativas e de teste; na área 2 são listados os treinamentos realizados e as ferramentas relacionadas a detalhes do treinamento (a), teste (b), exclusão (c) e a criação de um novo treinamento (d)

Vale ressaltar que, caso não haja treinamentos realizados no projeto, a área “Treinamentos Realizados” não exibirá quaisquer informações. Além disso, só serão habilitadas as ferramentas relacionadas a área “Treinamentos Realizados” caso seja selecionado algum treinamento.

A funcionalidade relacionada a exclusão (Figura 2.3 C) de um treinamento realizado na área “Treinamentos Realizados” não está disponível e, para que seja realizada a exclusão, deve-se acessar o diretório do projeto e apaga-lo manualmente, tanto o diretório com os arquivos, quanto o cabeçalho no arquivo “`listaCascade.txt`”. Ambos os arquivos (de treinamento e o “listaCascade.txt” podem ser localizados no diretório padrão, de instalação, em:
`C:\Detector de Objetos\sample\projetos\PROJETO\cascade\`

Um detalhe importante é que, a lista relacionada aos arquivos de treinamento (*cascade*) do projeto estará disposta da seguinte forma:

```
featureType\bt\TipoImagem\numStages
```

Tal forma é proporcionada para identificar a configuração de maneira básica de cada um dos arquivos de treinamento (*cascade*), além de servir como uma referência de localização dentro do diretório do sistema.

3. NOVO PROJETO

Para criar um novo projeto usando o sistema construído, clique no botão “Novo Projeto” na janela principal do GPDVC.

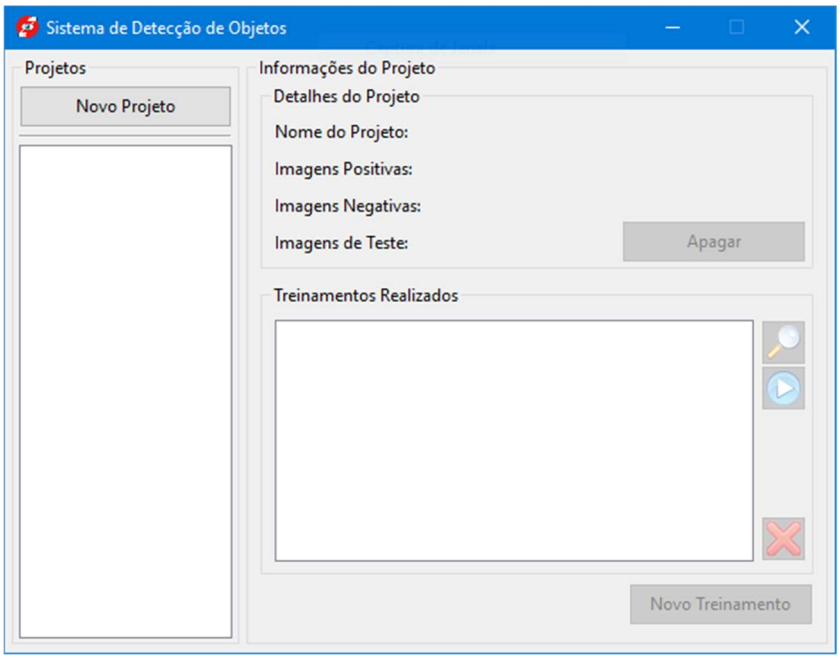


Figura 3.1 - Janela principal do GPDVC

Após clicar no botão “Novo Projeto”, será exibida a janela “Novo Projeto 1-6: Nome do Projeto”, do qual, solicita-se o nome do novo projeto. Assim, no campo “Nome do Projeto” informe o nome desejado para o projeto. Não utilize acentuação, espaço ou qualquer tipo de caractere especial, pois o sistema não realiza o tratamento e em processos futuros serão gerados erros. Como dica, utilize nomes no singular, pois este nome será utilizado nas etiquetas de identificação quando for realizar os testes em imagens.

Após definir um nome, clique no botão “Avançar”.

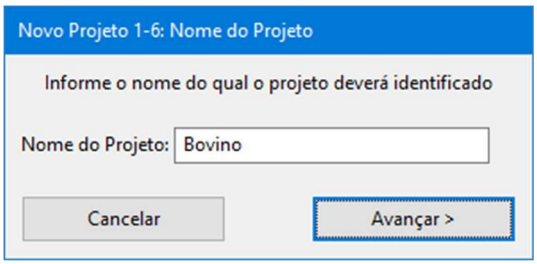


Figura 3.2 - Janela "Novo Projeto 1-6: Nome do Projeto"

Caso tente criar um projeto com um nome de um projeto existente, será exibido um alerta informando que o projeto com este nome já existe, e retornará para a janela de "Novo Projeto 1-6: Nome do Projeto" para informar um outro nome para o projeto.

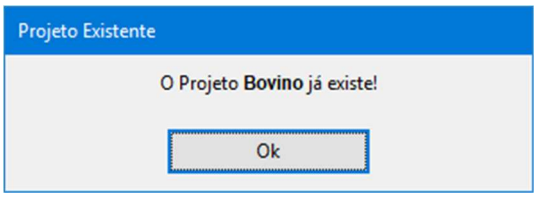


Figura 3.3 - Janela "Projeto Existe"

Assim que clicar no botão "Avançar", será exibida a janela "Novo Projeto 2-6: Selecionar Imagens". Nesta etapa, deve-se realizar a seleção do diretório do qual, encontram-se as imagens Positivas[1] e Negativas[2]. Assim que selecionado os diretórios clicando nos botões com os três pontos (...) e em seguida, clique no botão "Avançar".

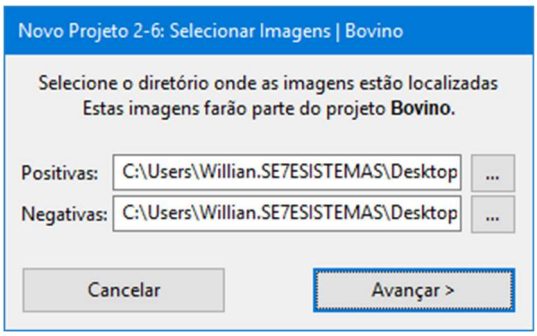


Figura 3.4 - Janela "Novo Projeto 2-6: Selecionar Imagens"

O sistema consegue identificar se os diretórios informados para as imagens positivas e negativas forem iguais e, neste caso, se tentar prosseguir com o processo, será emitido o alerta abaixo:

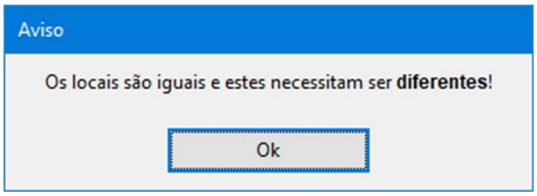


Figura 3.5 - Locais iguais

[1] Imagens Positivas: Refere-se a imagens que contém o objeto do qual, deseja-se realizar a detecção.
[2] Imagens Negativas: Refere-se a imagens aleatórias do qual, não se encontra o objeto que se deseja realizar a detecção. É aconselhável utilizar de imagens que remetem ao local onde tal objeto pode ser encontrado. Exemplo: Para o projeto "Bovino" utilizou-se principalmente imagens relacionadas a pastagem e vegetações em geral.

Caso haja outros arquivos que não estão no formato JPG ou BMP o sistema não os copiará. Além disso, O botão “Avançar” só será liberado caso os diretórios para as imagens positivas e negativas sejam informados.

Assim que clicar no botão “Avançar” será exibida a janela “Novo Projeto 3-6: Selecionar Filtros”. Nesta janela, é possível realizar a seleção de filtro de otimização de imagens. Atualmente, o sistema conta apenas com a Normalização ou Equalização do Histograma nas imagens. Caso deseje ter disponível imagens com este processamento, basta marcar a opção “Equalização do Histograma” e clicar no botão “Avançar”.

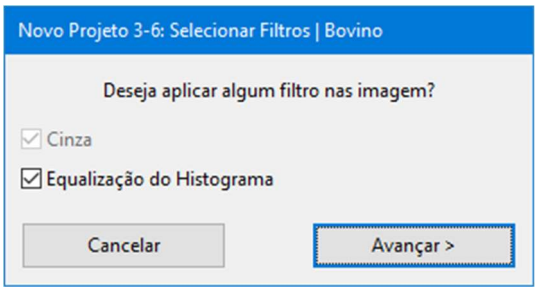


Figura 3.6 - Janela "Novo Projeto 3-6: Selecionar Filtros

Após clicar no botão “Avançar” será exibida a janela “Novo Projeto 4-6: Tamanho Máximo”. Nesta janela é possível definir o tamanho, em pixels, que as imagens positivas assumirão. Ao informar o valor, basta clicar no botão “Avançar”. Vale ressaltar que as imagens negativas, por padrão do sistema, assumirão o dobro do tamanho das imagens positivas, sendo assim, caso as imagens positivas sejam definidas para 50px, as imagens negativas possuirão o tamanho de 100px. Além disso, as imagens são redimensionadas de maneira escalar para não interferir no formato da imagem. É importante frisar para inserir valores inteiros pois o sistema não realiza o tratamento destes valores.

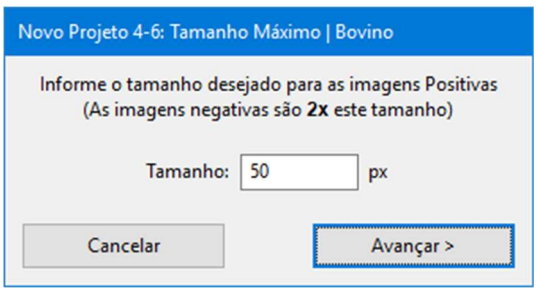


Figura 3.7 - Janela "Novo Projeto 4-6: Tamanho Máximo"

Após definir o tamanho das imagens positivas e clicar no botão “Avançar”, será exibida a janela “Novo Projeto 5-6: Realizar Treinamento”. Nesta janela, é possível selecionar

se deseja ou não iniciar um treinamento para o projeto. Nesta janela ocorre o processamento de todo o percurso para a criação do projeto, ou seja, é a partir desta janela que serão copiadas as imagens, aplicados os filtros e aplicado o redimensionamento das imagens. Desta forma, dependendo da quantidade de imagens e dos recursos computacionais disponíveis, tal processo pode levar demasiado tempo.

Este processo não foi otimizado a ponto de não gerar travamentos no sistema. Assim, pode parecer que a janela possa não estar respondendo, mas o processo está ocorrendo em background. Vale ressaltar que não há feedback de processamento, ou seja, não é possível verificar quais imagens foram copiadas ou em qual parte do processo se encontra a criação do projeto.

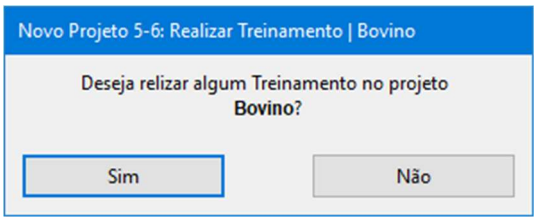


Figura 3.8 - Janela "Novo Projeto 5-6: Realizar Treinamento"

Assim que o projeto for criado, será exibida uma das janelas abaixo dependendo da opção selecionada na janela anterior.

Caso deseja realizar um treinamento após a criação do projeto, será exibida a janela abaixo contendo a mensagem “Iniciando o Processo de Treinamento”, do qual abrirá a janela “Treinamento 1-3: Filtro”, que pode ser vista no tópico 4. Treinamento, cito a partir da página 77.

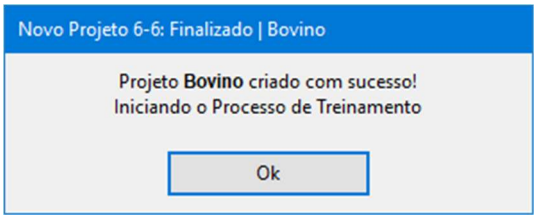


Figura 3.9 - Janela "Novo Projeto 6-6: Finalizado" COM Treinamento

Caso não deseje realizar um treinamento após a criação do projeto, ou seja, na janela “Novo Projeto 5-6: Realizar Treinamento” for pressionado o botão “Não”, será exibida a janela abaixo, do qual, informa que o projeto foi criado. Quando pressionado o botão “Ok”, será exibida a janela principal do sistema.

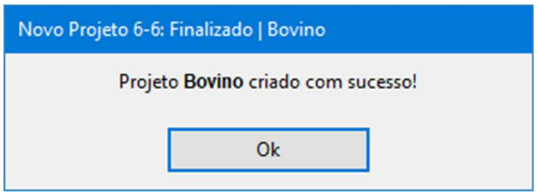


Figura 3.10 - Janela "Novo Projeto 6-6: Finalizado" SEM Treinamento

Assim que finalizado a criação do projeto e retornado para a janela principal do sistema, será exibido no lado esquerdo da janela, conforme mostra a marcação na Figura 3.11, uma lista com o nome do(s) projeto(s).

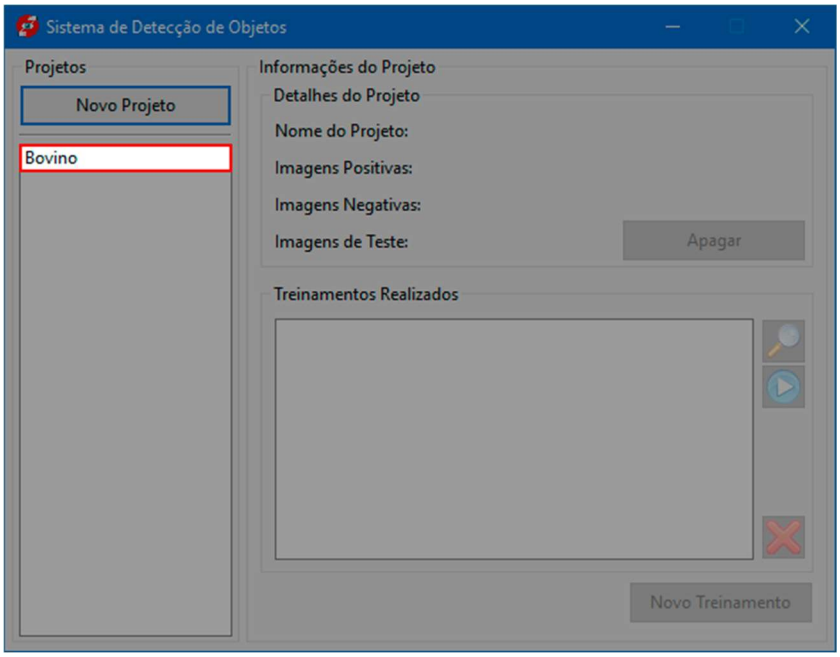


Figura 3.11 - Janela principal do Sistema: Projeto “Bovino” criado

Para verificar informações acerca de um projeto, basta clicar no mesmo.

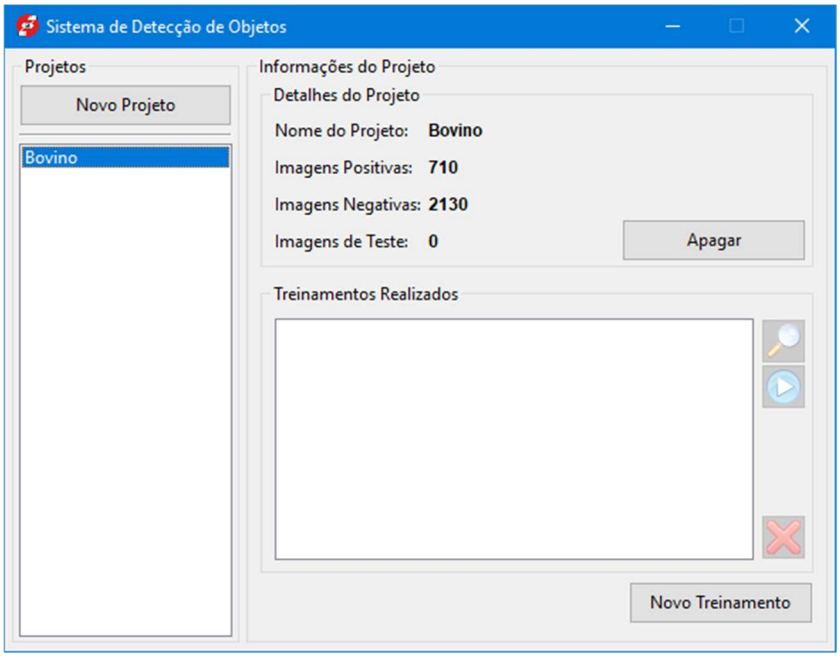


Figura 3.12 - Janela principal do Sistema: Detalhes do projeto "Bovino"

Caso deseje excluir o projeto, primeiramente, deve-se selecionar o projeto desejado e clicar no botão “Apagar”.

Assim que clicar sobre o botão “Apagar”, será exibida a seguinte janela:

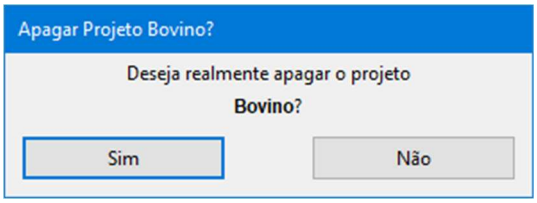


Figura 3.13 - Janela "Apagar Projeto"

Caso clique no botão “Não”, será exibida novamente a janela principal do sistema. Caso clique no botão “Sim”, o projeto será apagado do sistema e será exibida a janela abaixo, do qual, confirma a exclusão do projeto.

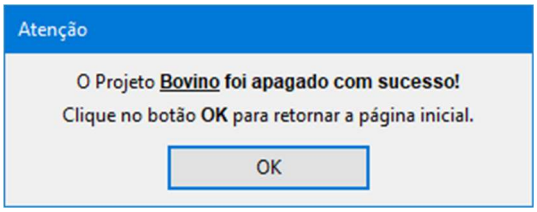


Figura 3.14 - Janela informativa do processo de Apagar Projeto

É nesta etapa em que o projeto de fato é apagado, sendo assim, dependendo da quantidade de arquivos dentro do diretório, pode levar demasiado tempo. Vale ressaltar que não

há um feedback do processo e que podem ocorrer travamentos no sistema, mesmo assim, o processo é finalizado corretamente, basta aguardar sua finalização. Além disso, caso o diretório do projeto estiver sendo usado, serão emitidos vários alertas do sistema, que impossibilitará a conclusão do processo, sendo assim, finalize as atividades no diretório do projeto para que o mesmo seja apagado sem problemas.

4. TREINAMENTO

O treinamento está fundamentado no nicho principal do GPDVC. O intuito inicial era facilitar a criação de arquivos relacionados ao treinamento de um sistema de detecção de objetos a partir da utilização da biblioteca OpenCV.

Partindo deste princípio, para iniciar o processo de Treinamento necessita-se que seja criado um projeto e que o mesmo, apresente imagens categorizadas da seguinte forma:

- **Imagens Positivas**: estão relacionadas a imagens que possuem o objeto de interesse para a detecção. Estas imagens, preferencialmente, devem estar em formato BMP ou JPG, possuírem apenas 1 (um) objeto de interesse por imagem e que o objeto de interesse esteja em evidência, conforme pode ser demonstrado na Figura 4.1.

Figura 4.1 - Exemplo de imagem Positiva do projeto "Bovino"
Em (a) encontra-se a imagem adquirida na internet, em (b) e (c) são as imagens originadas da imagem (a)
Fonte: http://www.image-net.org, adaptado graficamente pelos autores

- **Imagens Negativas**: refere-se a imagens que não possuem o objeto de interesse para a detecção, ou seja, qualquer outro tipo de imagem. É interessante levar em consideração onde o objeto de interesse normalmente se encontra, como por exemplo, na Figura 4.1 o objeto de interesse (Bovino) encontra-se em um ambiente de pastagem, logo, as imagens negativas estão relacionadas à pastagem ou similares (florestas, rios, nuvens e afins), conforme pode ser visto na Figura 4.2.

Figura 4.2 - Exemplo de Imagens Negativas do projeto "Bovino"
Fonte: http://www.image-net.org, adaptado graficamente pelos autores

Vale ressaltar que, independente de aplicar o filtro relacionado à Normalização ou Equalização do Histograma, todas as imagens, sejam elas positivas, negativas ou de teste, ambas são convertidas em tons de cinza para otimizar o processamento das imagens.

Assim que o banco de imagens estiver preparado e separado corretamente, e ter criado um projeto (vide tópico 3. Novo Projeto, cito na página 70), selecione o projeto desejado e clique no botão “Novo Treinamento” conforme mostra a figura abaixo.

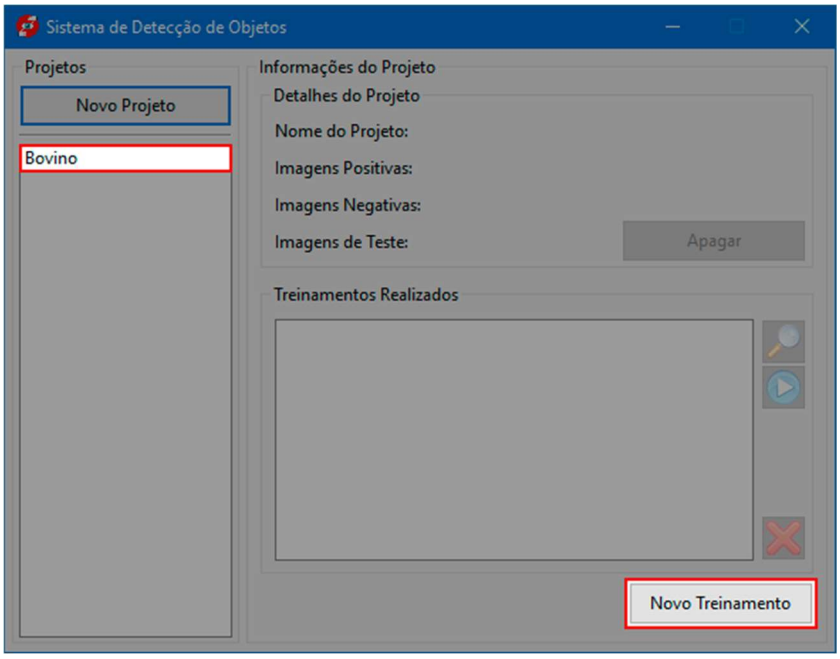


Figura 4.3 - Janela principal do Sistema: Novo Treinamento

Após clicar em “Novo Treinamento”, será apresentado a janela “Treinamento 1-3”, do qual, permite selecionar qual tipo de imagem será utilizada no processo de treinamento.

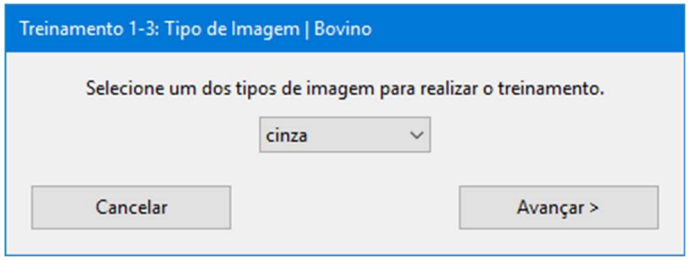


Figura 4.4 - Janela "Treinamento 1-3: Tipo de Imagem"

Vale ressaltar que o sistema não verifica se existem imagens dentro do diretório que armazena as imagens que passaram pelo processo de Normalização ou Equalização do Histograma, sendo assim, caso não tenha aplicado o filtro de Normalização do Histograma, será exibido uma mensagem de erro.

Após selecionar o tipo de imagem que fará parte deste treinamento, clique no botão “Avançar” que será exibida a janela “Treinamento 2-3: Sample”, conforme mostra a Figura 4.4. Nesta janela, é possível identificar alguns elementos relacionados ao projeto, como o local onde será salvo o arquivo vetor das imagens positivas (*vec*), o arquivo contendo as localizações das imagens positivas (*info*) e o número total das imagens positivas.

Além das informações citadas acima, deve-se realizar algumas configurações na área “Dados para a Criação | Tipo Múltipla Imagens”.

- **show** (*True* / *False*): Caso estiver selecionado a opção “*True*”, cada uma das imagens positivas será aberta repassada. Normalmente este parâmetro vem como padrão “*False*” e não sendo aconselhado trocar.
- **w** (*width* – largura): define a largura que as imagens serão redimensionadas para o treinamento. Este valor influencia diretamente na qualidade e no tempo de execução do treinamento, podendo variar de acordo com a problemática.
- **h** (*height* – altura): define a altura que as imagens serão redimensionadas para o treinamento. Assim como o parâmetro w, este parâmetro também influencia diretamente na qualidade e no tempo de execução do treinamento.

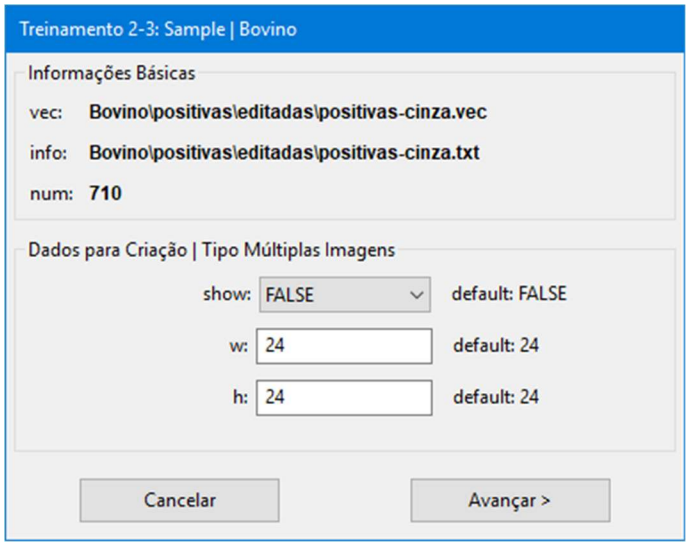


Figura 4.5 - Janela "Treinamento 2-3: Sample"

Assim que preenchido os campos, clique no botão “Avançar” que será aberto a janela “Treinamento 3-3: Cascade”. Esta janela apresenta maior complexidade, levando em consideração a quantidade de parâmetros que podem ser configurados e a variação com relação aos valores aceitos e as especificidades de cada um dos problemas relacionados à Visão Artificial. Tais informações podem e devem ser consultadas no próprio material de documentação da biblioteca OpenCV, que apresenta, inclusive alguns exemplos de uso (OPENCV.ORG, 2019).

Todos os campos já são preenchidos com valores tidos como padrão pela própria biblioteca, o que auxilia no processo de treinamento. Além disso, cada um dos campos está com os nomes dos próprios parâmetros, o que facilita ao estudar na documentação da biblioteca (OPENCV.ORG, 2019).

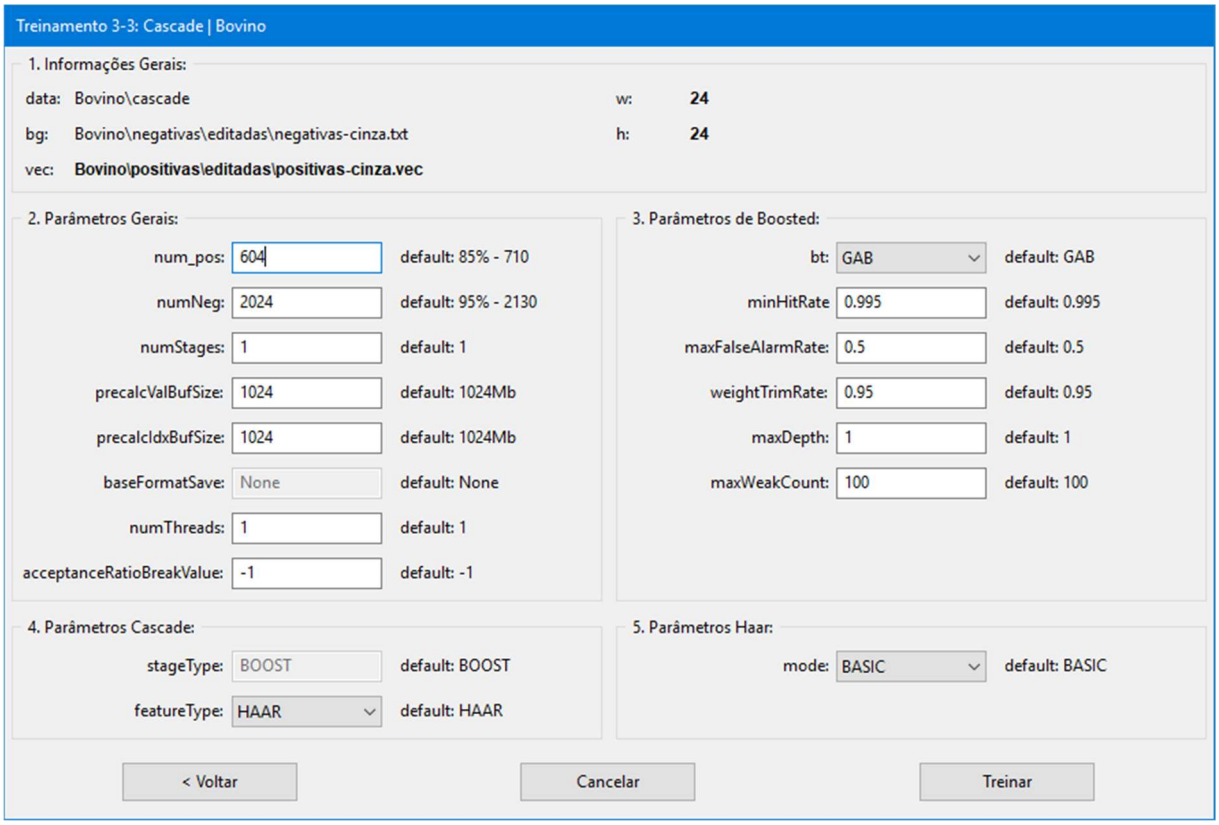


Figura 4.6 - Janela "Treinamento 3-3: *Cascade*"

Assim que configurado com os valores desejados, basta clicar no botão “Treinar” que será dado início ao treinamento. Vale ressaltar que, o tempo de treinamento é influenciado por todos os parâmetros configurados, além de estar relacionado com as configurações do computador em que está sendo executado o treinamento.

Quando concluído o treinamento, será exibido a janela principal do “Sistema de Detecção de Objetos” e, ao selecionar o projeto que foi treinado, na área de “Treinamentos Realizados”, será exibido o arquivo com o nome esquematizado de acordo com a hierarquia dos diretórios, ou seja, sob o nome relacionado às configurações `featureType\bt\TipoImagem\numStages`., conforme pode ser visto na figura abaixo.

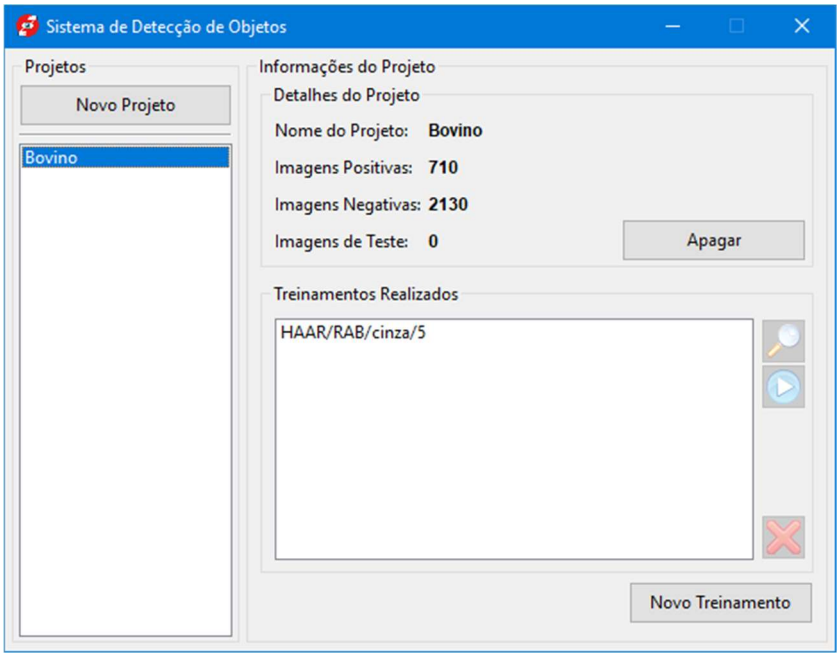


Figura 4.7 - Janela principal do Sistema: Projeto com treinamento realizado

Caso queira saber os detalhes referente ao arquivo de treinamento (*cascade*), selecione o projeto e em seguida, o arquivo de treinamento (*cascade*) desejado. Após selecionar, clique no botão “Detalhes” conforme mostra a figura abaixo.

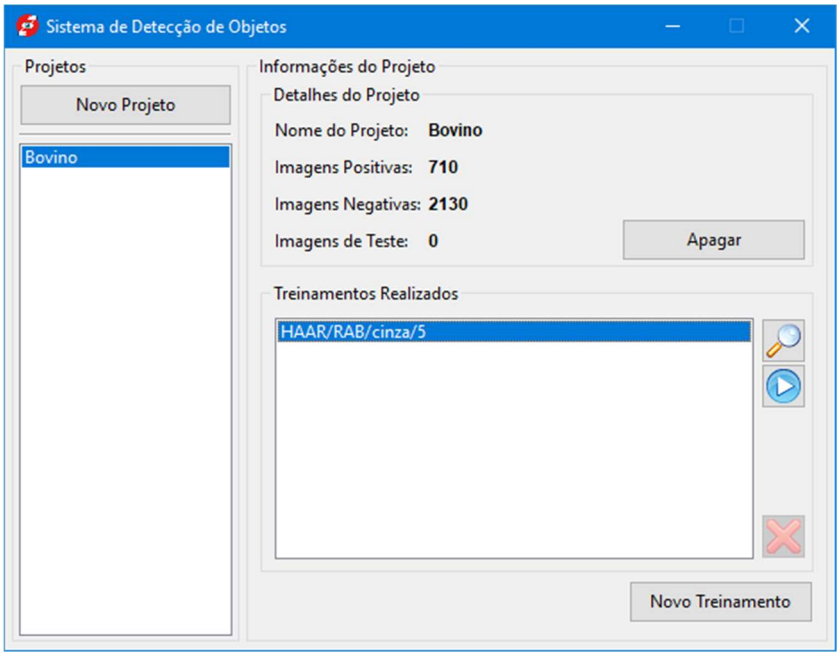


Figura 4.8 - Janela principal do Sistema: Projeto e *Cascade* selecionado

Ao clicar no botão “Detalhes”, será exibida a janela “Detalhes”, contendo as informações sobre as configurações utilizadas durante o treinamento.

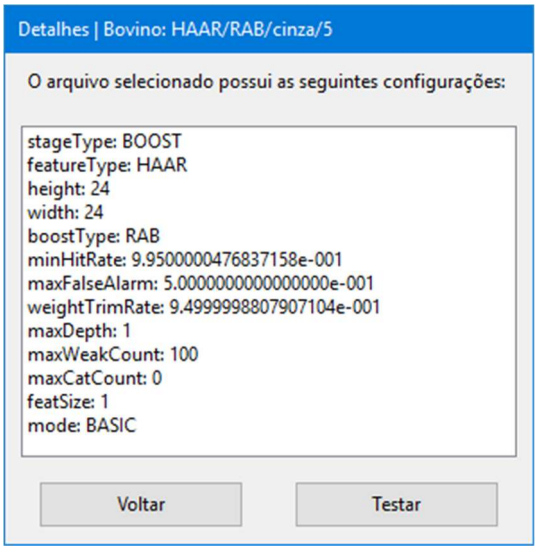


Figura 4.9 - Janela "Detalhes: CASCADE"

É possível, a partir da janela de “Detalhes” iniciar o processo de teste, clicando no botão “Testar” (vide tópico 5. Teste, cito na página 84) ou voltar para a janela principal clicando no botão “Voltar”.

5. TESTE

Após ter criado um projeto (vide tópico 3. Novo Projeto, página 70) e realizado um treinamento (vide tópico 4. Treinamento, página 77), é possível realizar alguns testes utilizando-se de vídeos e principalmente imagens. Para a realização do teste, selecione o projeto do qual deseja-se realizar o teste e, em seguida, selecione o arquivo de treinamento (*cascade*) desejado para realizar o treinamento.

Assim que selecionado o arquivo de treinamento (*cascade*) na lista, serão habilitados alguns botões, sendo o botão "Detalhes", responsável por exibir as informações (configurações utilizadas) do arquivo de treinamento e o botão "Testar", responsável por iniciar o fluxo de teste com o arquivo de treinamento (*cascade*) selecionado.

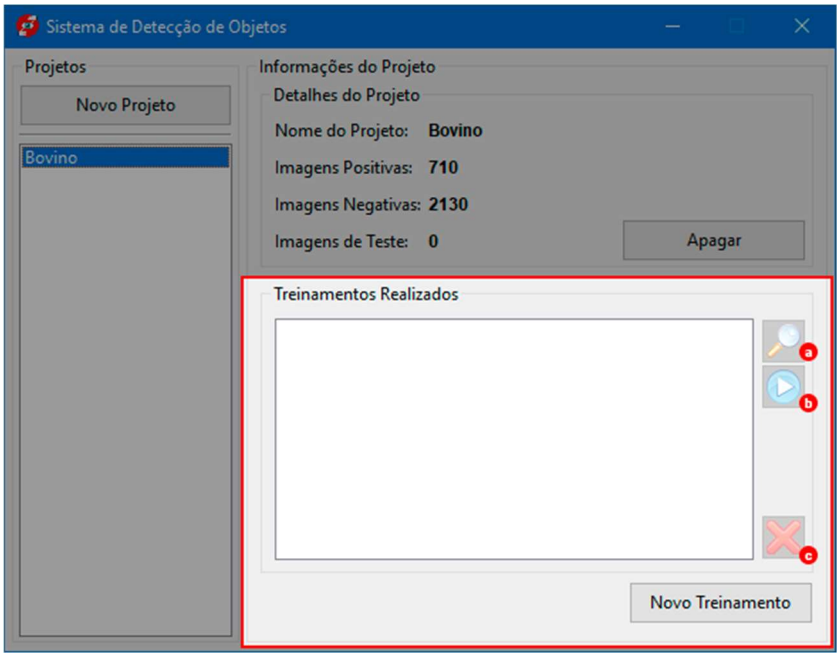


Figura 5.1 - Janela principal do GPDVC: Projeto e treinamento selecionado para teste
Em (a) é apresentado o botão referente aos detalhes do arquivo de treinamento (*cascade*); em (b) o botão para a iniciar a execução do teste; e (c) a opção para apagar o treinamento (*cascade*) selecionado

A funcionalidade relacionada à exclusão (Figura 2.3 C) de um treinamento realizado na área "Treinamentos Realizados" não está disponível e, para que seja realizada a exclusão, deve-se acessar o diretório do projeto e apaga-lo manualmente, tanto o diretório com os arquivos, quanto o cabeçalho no arquivo "listaCascade.txt". Ambos os arquivos (de treinamento e o "listaCascade.txt" podem ser localizados no diretório padrão, de instalação, em:

`C:\Detector de Objetos\sample\projetos\PROJETO\cascade\`

Sendo assim, para iniciar o processo do teste, pressione o botão “Teste”.

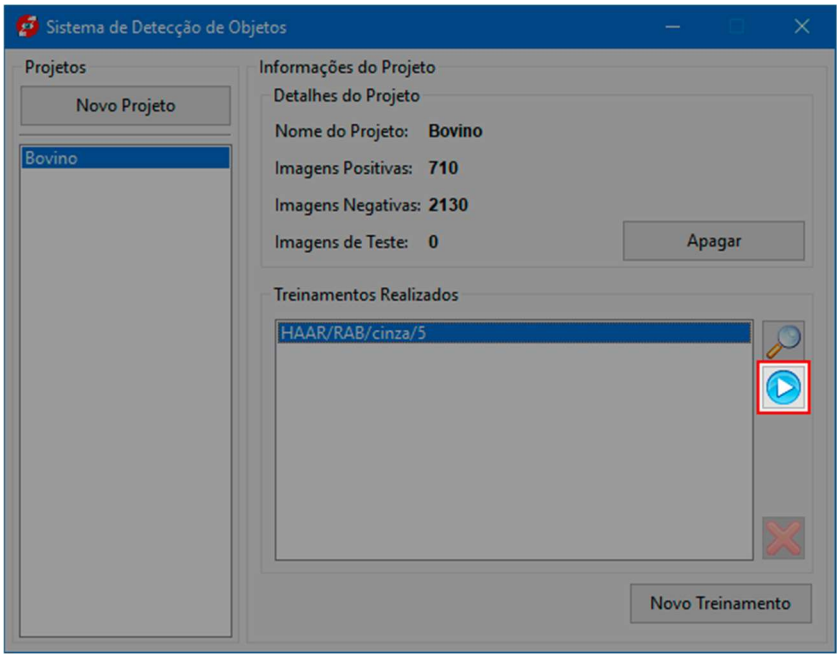


Figura 5.2 - Janela principal do GPDVC: Teste

Ao pressionar o botão “Teste” será exibida a janela “Testar: Selecionar Tipo de Teste”, onde é possível selecionar qual o tipo de teste que se deseja realizar: se é um teste voltado a imagens ou teste com vídeos.

Figura 5.3 - Janela “Testar: Selecionar Tipo de Teste”

5.1 TESTE COM IMAGEM

No caso dos testes aplicados a imagens, estes apresentam maior estabilidade quanto ao processo, além de possibilitar uma análise mais apurada quanto a qualidade do detector. Recomenda-se que, após a realização de um treinamento, execute o processo de teste com imagens para avaliar a precisão da detecção.

Vale ressaltar que, para um bom processo de teste, é recomendado separar algumas imagens que não participaram do processo de treinamento, para que seja possível a realização de uma apuração mais precisa.

Assim que selecionar um projeto, selecionar o arquivo de treinamento (cascade) e pressionar o botão Testar, será exibida a janela abaixo. Deste modo, para realizar o teste com imagens, pressione o botão “Imagens”.

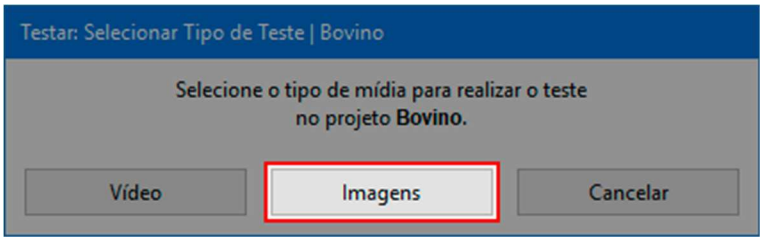


Figura 5.4 - Janela “Testar: Selecionar Tipo de Teste”: Teste tipo Imagem

Após pressionar o botão “Imagens” será exibida a janela “Testar 1-3: Imagens – Selecionar Imagens”, do qual, solicita um diretório que contenha as imagens que serão utilizadas no teste. Além disso, é recomendado a utilização de imagens ou no formato JPG ou no BMP.

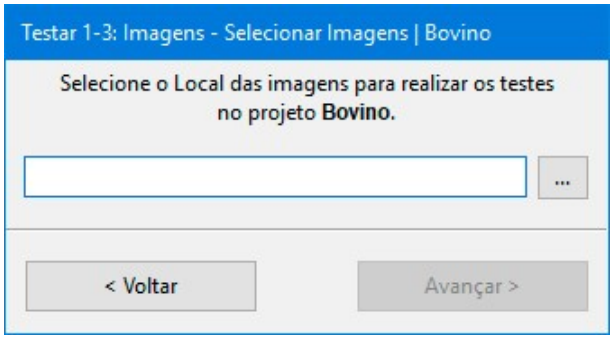


Figura 5.5 - Janela “Testar 1-3: Imagens - Selecionar Imagens”: Diretório de imagens para teste

Assim que selecionado um diretório, será liberado o botão “Avançar” para dar continuidade no processo. Ao pressionar o botão “Avançar”, será exibida a janela “”, o qual solicita determinadas informações para o usuário. Tais informações servirão para o sistema calcular uma taxa de erros e acertos.

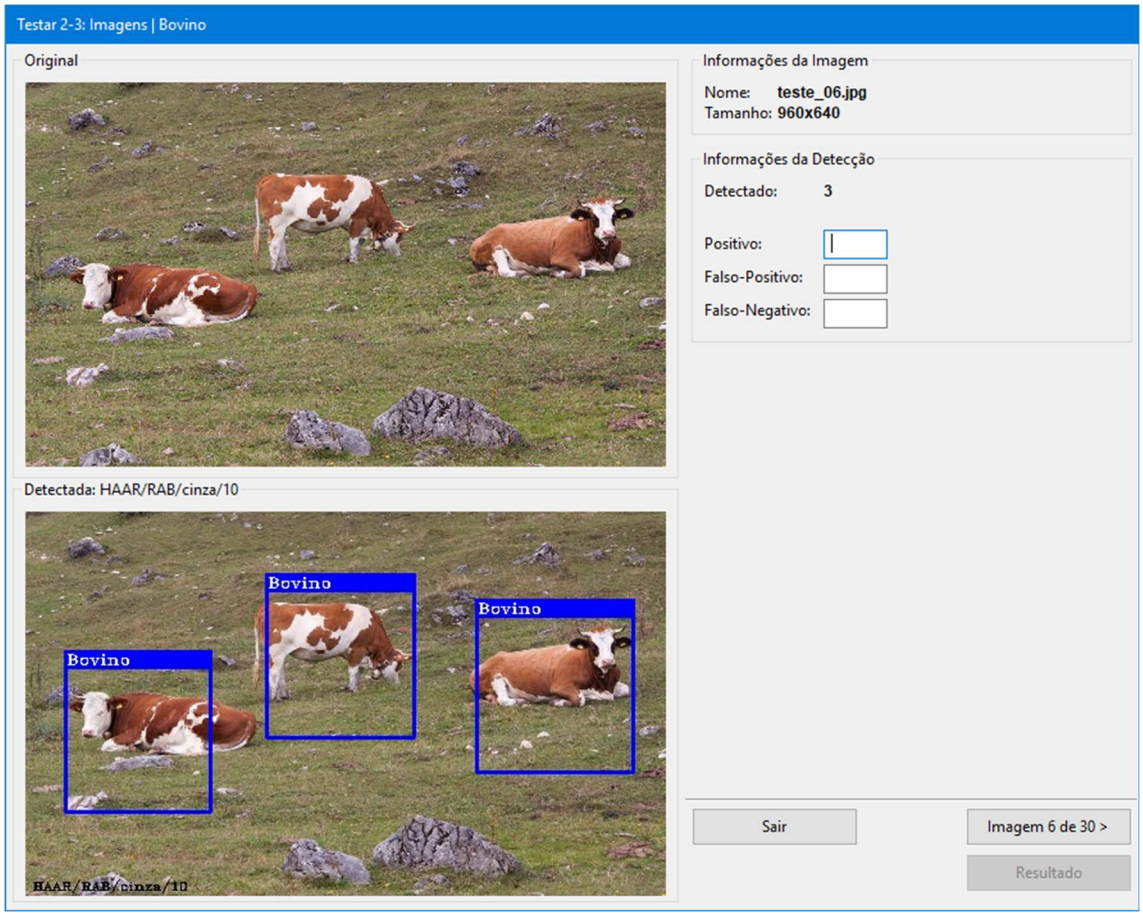


Figura 5.6 - Janela “Testar 2-3: Imagens”: Informações da detecção na imagem

- **Positivo**: deve-se informar a quantidade de detecções realizadas corretamente.
- **Falso-Positivo**: refere-se à quantidade relacionada às detecções incorretas.
- **Falso-Negativo**: relaciona-se com objetos de interesses não selecionados, ou seja, objetos que o sistema deveria detectar e não detectou.

Após informar os dados nos campos acima, basta ir clicando no botão relacionado ao fluxo de imagem, do qual, apresentará o formato “Imagem N de M”, onde:

- **N**: representa o número atual da imagem.
- **M**: representa a quantidade total de imagens que estavam contidas no diretório selecionado anteriormente.

Assim que concluído, será habilitado o botão “Resultado”, que, ao pressioná-lo, será exibida a janela “Testar 3-3: Imagens - Resultado”.

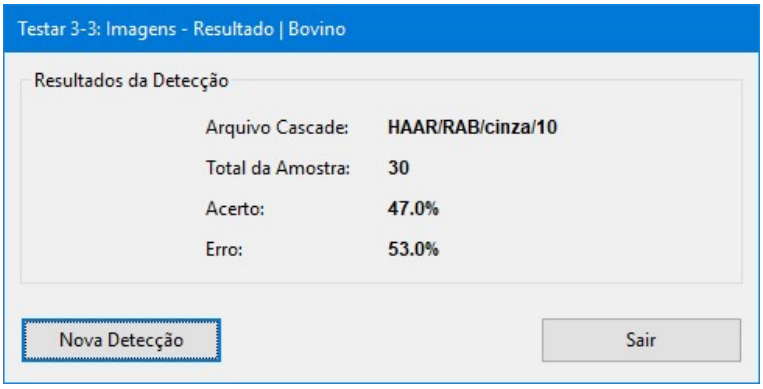


Figura 5.7 - Janela “Testar 3-3: Imagens - Resultado”: Resultado do teste

Após verificar os dados, basta clicar no botão “Sair” para retornar a janela principal, ou pressionar o botão “Nova Detecção”, que retornará para a janela “Testar: Selecionar tipo de Teste” representada na Figura 5.3, onde é possível selecionar o tipo de teste.

Além disso, caso seja executado um novo teste com imagens, as imagens anteriormente testadas serão apagadas do diretório do sistema. As imagens estarão disponíveis dentro do sistema, podendo serem localizadas em: `C:\Detector de Objetos\sample\projetos\PROJETO\teste\imagens\`

Os parâmetros utilizados para a detecção estão fixados no código-fonte e não possuem a possibilidade de modificar. Sendo assim, seus valores padrões dentro do sistema para a detecção nas imagens são:

Tabela 5.1 - Parâmetros e valores utilizados na detecção em imagens

Parâmetro	Valor utilizado
scaleFactor	1.1
minNeighbors	5
minSize	(100, 100)
maxSize	(alturaTotal, larguraTotal)

5.2 TESTE COM VÍDEO

Antes de prosseguir esteja ciente de que, o processo de teste com vídeo apresenta a maior instabilidade dentro do sistema. Esta instabilidade está atrelada pelo fato de que, um vídeo nada mais é do que uma coletânea de imagens sequencializadas e o sistema detecta *frame* a *frame* e salva o novo *frame* com a detecção, causando tal instabilidade. Além disso, recomenda-se a utilização de vídeos curtos e de preferência com baixa resolução, para facilitar no processo e diminuir o tempo de processamento e evitar parcialmente sua instabilidade.

Após clicar no botão “Vídeo” na janela “Testar: Selecionar tipo de Teste”, será exibida a janela “Testar 1-2: Selecionar Vídeo”. Nesta janela, deve-se selecionar o vídeo do qual, deseja-se realizar o teste. Vale ressaltar que os formatos suportados para a execução dos testes em vídeo são: **AVI**, **WMV** e **MP4**.

Quando o vídeo for selecionado, será habilitado o botão “Testar”. Após pressioná-lo, será dado início ao processo de teste. Este processo de teste apresenta grande instabilidade, além de não apresentar um *feedback* de processo ou algo similar. Porém, quando finalizado, será exibida a janela “Testar 2-2: Vídeo”, do qual, será executado o vídeo, porém, com os retângulos referentes à detecção realizada pelo sistema.

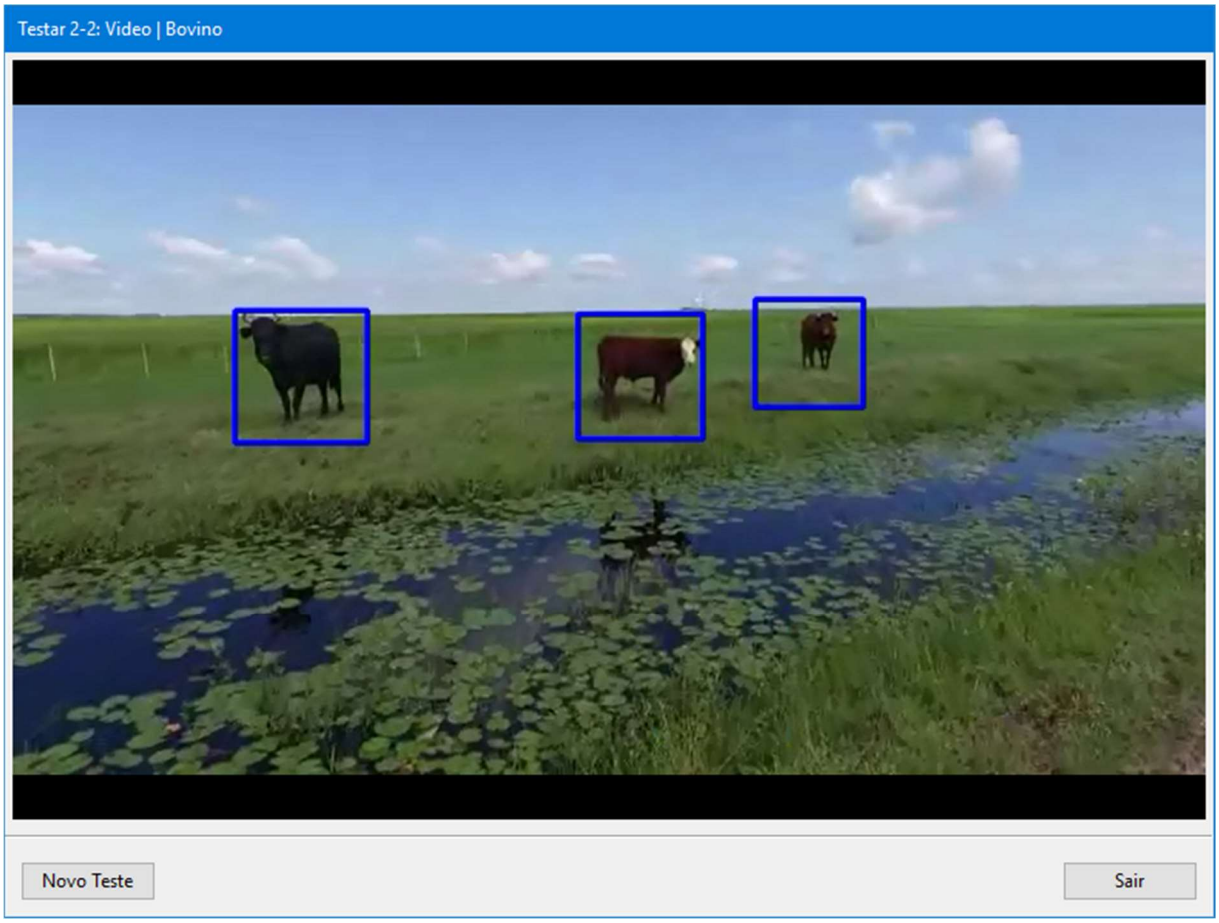


Figura 5.8 - Janela "Testar 2-2: Vídeo": Detecção após o processamento do vídeo

Ao finalizar, basta clicar no botão “Sair” para retornar à janela principal do sistema. Caso seja executado um novo teste com vídeo, a vídeo anteriormente testado será apagado do diretório do sistema. O vídeo estará disponível dentro do sistema, podendo ser localizado em: `C:\Detector de Objetos\sample\projetos\PROJETO\teste\vídeo\`

Os parâmetros utilizados para a detecção estão fixados no código-fonte e não possuem a possibilidade de modificar. Sendo assim, seus valores padrões dentro do sistema para detecção em vídeos são:

Tabela 5.2 - Parâmetros e valores utilizados na detecção em vídeos

Parâmetro	Valor utilizado
scaleFactor	1.1
minNeighbors	35
Minsize	(50, 50)
maxSize	(600, 600)