

**MINISTÉRIO DA EDUCAÇÃO SECRETARIA DE EDUCAÇÃO PROFISSIONAL E
TECNOLÓGICA INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E
TECNOLOGIA FARROUPILHA CAMPUS PANAMBI**

SISTEMA DE DETECÇÃO DE VEÍCULOS E ANÁLISE DE SITUAÇÃO CADASTRAL

TRABALHO DE CONCLUSÃO DE CURSO

Carlos Cristiano Wojahn

Panambi, RS, Brasil

2022

Carlos Cristiano Wojahn

SISTEMA DE DETECÇÃO DE VEÍCULOS E ANÁLISE DE SITUAÇÃO CADASTRAL/ Carlos Cristiano Wojahn. – Panambi, RS, Brasil, 2022-
33 p. : il. (algumas color.) ; 30 cm.

Orientador: Cléber Rubert, Me.

Trabalho de Conclusão de Curso (TCC) – Instituto Federal Farroupilha, 2022.

1. Detecção veicular. 2. Reconhecimento de caracteres. I. Cléber Rubert. II. Instituto Federal de Educação, Ciência e Tecnologia Farroupilha Campus Panambi. IV. Sistema de Detecção de Veículos e Análise de Situação Cadastral Aplicada a Vigilância Residencial

CDU 02:141:005.7

**Ministério da Educação
Secretaria de Educação Profissional e Tecnológica
Instituto Federal de Educação Ciência e Tecnologia Farroupilha**

A Comissão Examinadora, abaixo assinada,
aprova a Monografia

SISTEMA DE DETECÇÃO DE VEÍCULOS E ANÁLISE DE SITUAÇÃO CADASTRAL

elaborada por
Carlos Cristiano Wojahn

como requisito parcial para obtenção do título de
Tecnólogo em Sistemas para Internet

COMISSÃO EXAMINADORA

Cléber Rubert, Me.
Presidente/Orientador

Táisa Belzarena Monteiro, Me.
Instituto Federal Farroupilha

Éverton Lutz, Dr.
Instituto Federal Farroupilha

Conceito Final: Aprovado

Panambi, 29 de novembro de 2022.

*“Computadores são inúteis,
eles conseguem apenas lhe dar respostas.”
(Pablo Picasso)*

RESUMO

A utilização de câmeras de vigilância é amplamente difundida e aplicada em diversas situações e ambientes, sendo ela utilizada em grandes centros inclusive para o monitoramento de situação irregular de veículos, porém a utilização destes sistemas integrados em regiões residenciais depende de altos investimentos. A utilização de veículos furtados/roubados em outros crimes tem se tornado corriqueiro, o presente trabalho tem como intuito demonstrar as capacidades existentes no campo de inteligência artificial, visão computacional e aprendizagem de máquinas na detecção de veículos e sua situação cadastral, sendo que estes dados podem ser utilizados em inúmeras outras possibilidades. Neste trabalho foram adotadas imagens capturadas por câmeras instaladas nas cabinas de cobrança nos pedágios da rodovia ES-060, no Espírito Santo, Brasil, sob concessão da empresa Rodovia do Sol (RodoSol). Tais imagens passaram por tratamento e posterior treinamento de um modelo de detecção de veículos e suas placas de identificação. Processo que possibilitou o posterior reconhecimento de caracteres e verificação da situação cadastral do veículo através da *ApiPlacas*, sistema que disponibiliza inúmeras informações referentes aos veículos pesquisados.

Palavras-chaves: visão computacional. inteligência artificial. aprendizagem de máquina. detecção de objetos. detecção de placas. reconhecimento de caracteres.

ABSTRACT

The use of surveillance cameras is widespread and applied in different situations and environments, being used in large centers including for monitoring the irregular situation of vehicles, however the use of these integrated systems in residential areas requires high investments. The use of theft/stolen vehicles in other crimes has become commonplace, the present work aims to demonstrate the existing capabilities in the field of artificial intelligence, computer vision and machine learning in the detection of vehicles and their registration situation, and these data are used in countless other possibilities. In this work, images captured by cameras installed in the tollbooths of the ES-060 highway, in Espirito Santo, Brazil, under concession of the company Rodovia do Sol (RodoSol), were adopted. Such images underwent treatment and subsequent training of a vehicle detection model and their identification plates. Process that enabled the subsequent recognition of characters and verification of the registration status of the vehicle through ApiPlacas, a system that provides numerous information regarding the vehicles surveyed.

Keywords: computer vision. artificial intelligence. deep learning. object detection. license plate detection. character recognition.

LISTA DE ILUSTRAÇÕES

Figura 1 – Fluxo do Sistema	11
Figura 2 – Fluxo do Sistema e suas Tecnologias	14
Figura 3 – Fluxo do Sistema de Diferenças Acumulativas das Imagens (PRIADANA; HARJOKO, 2017)	16
Figura 4 – Fluxo de Detecção de Veículos e Placas de Identificação Veicular (LAROCA et al., 2021)	17
Figura 5 – Placas de Identificação Veicular conforme Resolução N°780/2019 (NACIONAL, 2019)	18
Figura 6 – Placas de Identificação Veicular conforme Resolução N°372/2011 (NACIONAL, 2011)	18
Figura 7 – Exemplo de imagens do <i>dataset</i> (Laroca et al., 2022)	21
Figura 8 – Exemplo de uso do LabelImg	22
Figura 9 – Plataforma <i>Google Colab</i> - Instalação/Configuração do <i>framework Darknet</i>	23
Figura 10 – Plataforma <i>Google Colab</i> - Configuração da biblioteca <i>YOLOv3</i>	24
Figura 11 – Plataforma <i>Google Colab</i> - Carregamento das imagens de treinamento	24
Figura 12 – Plataforma <i>Google Colab</i> - Carregamento das imagens de treinamento	25
Figura 13 – Detecção de placa veicular com a confiança calculada pelo <i>DNN</i>	26
Figura 14 – Região recortada e inversamente binarizada.	26
Figura 15 – Integração ao sistema <i>ApiPlacas</i>	28
Figura 16 – Retorno obtido pelo sistema	29

SUMÁRIO

1	Introdução	1
	Introdução	1
1.1	OBJETIVOS	2
	Objetivos	2
1.1.1	Objetivo Geral	2
	Objetivo Geral	2
1.1.2	Objetivos Específicos	2
	Objetivos Específicos	2
1.2	ORGANIZAÇÃO DO TRABALHO	2
	Organização do Trabalho	2
2	FUNDAMENTAÇÃO TEÓRICA	4
2.1	TRABALHOS RELACIONADOS	4
2.1.1	Métodos de Detecção de Movimento	4
2.1.2	Métodos de Detecção de Placas de Identificação	5
2.1.3	Métodos de Reconhecimento de Caracteres	6
2.2	FERRAMENTAS E BIBLIOTECAS	8
2.2.1	Python	8
2.2.2	Labelimg - Libxml2	8
2.2.3	Darknet e YOLOv3	8
3	METODOLOGIA	10
3.1	CAPTURA DE IMAGENS	12
3.1.1	Meio Físico de Captura	12
3.1.2	Ferramenta de Captura e Processamento das Imagens	12
3.1.3	Detecção de Movimento	15
3.1.4	Detecção de Veículos e Placas de Identificação Veicular	17
3.1.5	Reconhecimento de Caracteres	18
3.1.6	Análise de Situação Cadastral do Veículo	19
3.2	FERRAMENTAS	19
4	DESENVOLVIMENTO E RESULTADOS	21
4.1	DESENVOLVIMENTO DO SISTEMA	21
5	CONCLUSÕES E TRABALHOS FUTUROS	30
	Conclusão e Trabalhos Futuros	30
5.1	CONCLUSÕES	30
5.2	TRABALHOS FUTUROS	30

Referências	31
------------------------------	-----------

1. INTRODUÇÃO

Conforme dados da SSP (2021) do Estado do Rio Grande do Sul, entre os anos de 2015 e 2020 foram registrados 820149 furtos e 442027 roubos, números estes que trazem preocupação a população quanto a segurança em seu cotidiano. Na tentativa de inibir a prática destes delitos, a implantação de sistemas de monitoramento por câmeras de vídeo tem sido amplamente utilizadas pelo setor público (VASCONCELLOS; MENDES, 2018), porém a utilização destes sistemas de monitoramento além de acarretarem em mais custos públicos na contratação de servidores, a sua instalação ocorre em regiões com grande fluxo de pessoas e veículos, ficando as regiões residenciais desassistidas se comparadas aos centros comerciais. A instalação em propriedades particulares de sistemas de CFTV (Circuito Fechado de Televisão) em regiões residenciais é amplamente difundida e segundo a Abese (2021) teve um crescimento de 13% em 2020 com relação ao ano anterior. O questionamento portanto é acerca de como tais sistemas podem colaborar com a redução de delitos de furto e roubo.

Dentre as diversas técnicas que auxiliam nessa tarefa está a visão computacional que tem sido empregada em diversas aplicações desde o diagnóstico de doenças através de imagens de ultrassonografia e radiografias a detecção de veículos, placas de sinalização e pedestres. Existem diversas técnicas de visão computacional, dentre elas estão as redes neurais convolucionais, que utilizam um grande acervo de imagens catalogadas e categorizadas as quais serão utilizadas para treinamento da rede neural convolucional. A convolução é o processo de filtrar, a partir de uma operação matemática, a matriz de pixels da imagem de entrada em função de uma matriz filtro, o produto dessa operação é conhecida como *Feature Maps*, produto esse que é utilizado para comparação com o acervo de imagens catalogadas. A configuração da matriz filtro é a fase mais desafiadora do processo, pois é ela que define a eficiência do processo de categorização da imagem (BRANDIZZI, 2020). Neste sentido a adoção da linguagem de programação *Python* no desenvolvimento de sistemas teve um crescimento de aproximadamente 10% entre o início de 2015 e final de 2020 (STACK OVERFLOW, 2021) e com a popularização de bibliotecas como *OpenCV*, *YOLO* e *Tesseract OCR* com suas sub-bibliotecas de ferramentas especializadas em tratamento de imagens e modelos pre-treinados para detecção de objetos conhecidos e reconhecimento de caracteres, tornaram mais simples o processo de desenvolvimento na hora de processar, tratar e utilizar as imagens recebidas pelo sistema de captura (BRADSKI; KAEHLER, 2008; REDMON; FARHADI, 2018; SMITH, 2013).

A utilização de sistemas computacionais aliados a sistemas de inteligência artificial tem crescido nos decorrer dos últimos anos, tendo aplicações em análise de textos de teor jurídico como o "Projeto Victor", desenvolvido pelo Supremo Tribunal de Justiça em parceria com a Universidade de Brasília, com o intuito de acelerar o enquadramento das matérias de processos aos temas de repercussão geral (BRASIL, 2018). Sendo utilizado na análise de imagens em tempo real utilizadas em sistemas biomecânicos como os veículos autônomos no transporte de diferentes tipos de materiais e produtos, substituindo operários em funções perigosas (TAVARES,

2008).

O presente trabalho utiliza da visão computacional aliada a sistemas de armazenagem de dados na identificação de veículos automotores e sua situação cadastral junto ao órgão regulatório brasileiro, na tentativa de identificar veículos com situação cadastral irregular com registro de alerta e/ou roubo/furto, veículos normalmente utilizados para novos crimes e/ou desmanche. Para o presente trabalho serão utilizadas, para treinamento e verificação, imagens disponibilizadas pela Universidade Federal do Paraná (Laroca et al., 2022).

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Côncio das problemáticas apresentadas na seção de introdução, este trabalho visa desenvolver um sistema responsável pelo tratamento de *frames* de vídeos de câmeras de segurança para detecção de veículos e sua placa de identificação com posterior análise cadastral dos mesmos.

1.1.2 Objetivos Específicos

Na busca em atingir o objetivo principal deverão ser atingidos objetivos intermediários, que seguem:

- Identificar e definir quais bibliotecas *Python* são as mais adequadas para o caso;
- Treinar um modelo para classificação e detecção placas veiculares;
- Desenvolver a função de classificação de objetos;
- Desenvolver a função de detecção de placas de identificação dos veículos;
- Desenvolver a função de busca da situação cadastral do veículo;

1.2 ORGANIZAÇÃO DO TRABALHO

No intuito de atingir os objetivos especificados na sessão anterior, a divisão de capítulos deste trabalho é a que segue:

No Capítulo 2, são apresentadas as fundamentações teóricas do estado da arte sendo categorizados em métodos de detecção de movimento, com a técnica básica de detecção de movimento, a *One Gaussian*, a diferença de mínimo, máximo e máximo entre quadros, além do método não estruturado para modelar uma função de densidade de probabilidade multimodal. Já para a detecção de placas, são elencados os métodos *Edge-Based*, *Color-Based*, *Texture-Based*, *Character-Based*, *Statistical Classifier* e o método de *deep-learning* como o *YOLO*. Para o reconhecimento de caracteres são abordados os métodos de segmentação por conexão entre

pixels, a segmentação por projeção de perfis, o método de segmentação por conhecimento prévio e a segmentação por contornos. Tendo realizada a abordagem de segmentação de caracteres partimos para a abordagem de métodos de reconhecimento de caracteres sendo os métodos de binarização heurística, sendo que esse método é subdividido em métodos globais, locais e híbridos. Os métodos híbridos ainda sofrem uma nova subcategorização sendo o método baseado na forma do histograma, baseado em agrupamento, baseado em entropia, baseado nos atributos do objeto, método espacial e o método local. Ainda são abordados os métodos baseados em aprendizagem de máquina, como o método de aprendizagem não supervisionada, aprendizagem supervisionada e a aprendizagem profunda.

No Capítulo 3 é apresentada a metodologia desenvolvida para o presente trabalho, contendo todo o processo de treinamento e obtenção das imagens binarizadas através do método proposto, busca de dados e por fim a armazenagem em arquivos de tipo JSON dos valores obtidos na análise de cadastro junto ao órgão regulatório.

2. FUNDAMENTAÇÃO TEÓRICA

Nesta seção é apresentada a revisão teórica de todas as técnicas, bibliotecas, ferramentas, algoritmos utilizados na elaboração do trabalho.

2.1 TRABALHOS RELACIONADOS

Na atualidade existe uma gama diversa de técnicas de processamento de imagens, detecção de placas de identificação veicular. Neste capítulo são apresentados diversos trabalhos relacionados a detecção de movimento, identificação de placas veiculares, reconhecimento de caracteres, sendo apresentadas diversas técnicas de processamento.

2.1.1 Métodos de Detecção de Movimento

O trabalho realizado por Benezeth et al. (2010) compara as diversas técnicas de detecção de movimento a partir de algoritmos de subtração de fundo. A abordagem passa pela técnica básica de detecção de movimento, onde é realizada a comparação entre a imagem sem objetos em movimento com a variação de luminosidade quando ocorre uma movimentação, essa técnica sofre em comparação as demais pois ocorrem variações naturais de luminosidade durante o dia.

A técnica *One Gaussian* utiliza da função de densidade de probabilidade, sendo essa função treinada com diversos quadros de um vídeo, calculando assim um peso para cada pixel da imagem de fundo em uma distribuição gaussiana, onde são representadas a cor de fundo média e a matriz de variação para cada pixel em função do tempo. Quanto maior o valor apresentado na matriz de variação em conjunto com o gradiente temporal, maior a probabilidade do pixel em questão estar em movimento, essa técnica acaba tendo a desvantagem de exigir uma capacidade de processamento e memória superior a técnica básica de detecção de movimentos apresentada por Benezeth et al. (2010).

Na técnica de diferença de mínimo, máximo e máximo entre quadros, cada pixel é representado por um valor mínimo, máximo e máximo entre uma sequência de quadros de treinamento, onde esses valores são comparados a um valor definido pelo usuário multiplicado pela média da maior diferença entre quadros em toda a imagem. O algoritmo original dessa técnica apenas era funcional a vídeos em escala de cinza perdendo parte das informações constantes nos quadros originais.

Um método não estruturado pode ser usado para modelar uma função de densidades de probabilidade multimodal, onde é implementado um algoritmo que estima uma janela de Parzen para cada pixel, a definição de movimento através desta técnica vem do resultado obtido na janela de Parzen for inferior a um valor pre-definido pelo usuário.

Já no artigo escrito por Ramadhan, Sari e Sari (2018), ainda é abordado o método de *Accumulative Difference Images* (ADI), onde são comparadas diferenças entre vários quadros

consecutivos, reduzindo assim os erros e a detecção de imagens imprecisas por realizar a acumulação de diferenças entre quadros. O método utiliza dois subprocessos, um que realiza a busca por mudanças na imagem pela alteração de iluminação combinado com o corte da imagem reduzindo a área de observação. O método captura o primeiro quadro, o qual será utilizado como referência, realiza a transformação para escala de cinza. O algoritmo então segue capturando uma sequência de quadros por um período de tempo, os quais também são transformados em escala de cinza, onde algumas dessas imagens são comparadas a imagem de referência o que resulta em valores acumulativos de diferença. O valor acumulativo é então comparado a um valor limiar, caso o valor acumulativo for superior ao valor limiar então é assumido que existe algum movimento na imagem. Caso não houver a detecção de movimento o algoritmo roda novamente a aquisição de imagens em sequência de quadros. Caso houver movimento o sistema armazena a imagem e realiza a busca por movimento novamente, caso o movimento não persistir por mais de 10 segundos o sistema realiza a aquisição de nova imagem de referência, ou através de captura de imagem ou por geração de imagem a partir das imagens capturadas anteriormente. A utilização deste método reduz os efeitos de ruídos ou falsos movimentos, e a taxa de assertividade do método é próxima de 95%.

2.1.2 Métodos de Detecção de Placas de Identificação

No âmbito de detecção de placas de identificação de veículos Shashirangana et al. (2020) cita os seis principais métodos de detecção de placas veiculares, iniciando pelo método de baseado em contornos (*Edge-Based*) onde é considerado que uma placa de identificação veicular tem formato e proporções conhecidos, técnica comumente utilizada porém com um grande número de falsos positivos provocados por ruídos em imagens.

O método baseado em cores (*Color-Based*) leva em consideração que placas de identificação veicular tem a cor de seu fundo diferente da cor de veículos, esse método dificilmente é utilizado de forma isolada, porém como um complemento a algum outro método, em casos em que a placa de identificação está deformada ou inclinada.

O método baseado em textura (*Texture-Based*) usa a presença dos caracteres como fator determinante, em imagens em escala de cinza a diferença entre a cor dos caracteres e do fundo da placa de identificação é maior do que em outras regiões, o que causa uma distribuição intensa de píxeis no entorno da placa de identificação. Porém é um método que utiliza uma carga demasiada de computação, além de sofrer com imagens com fundos complexos e condições de iluminação diferentes.

No método baseado em caracteres (*Character-Based*), são extraídos todas as regiões onde parece haver um caractere, com a ajuda de uma rede neural são verificados os extremos dessa região e caso alguma configuração linear seja encontrada então é pressuposto que essa região é uma placa de identificação. Este método obtém uma assertividade de 95% porém demanda um tempo elevado no processamento além de retornar falsos positivos caso existam mais caracteres na imagem além dos contidos na placa de identificação.

O método baseado em classificadores estatísticos (*Statistical Classifier*) utilizam-se de *Harr-like Features* com *Adaptive Boost (AdaBoost)* para treinar uma árvore de decisões em cascata pois os recursos estatísticos trazem mais simplicidade ao processo. Conforme o estudo realizado por Shashirangana et al. (2020) a utilização deste método atingiu uma assertividade de 94,5% com diferentes ângulos de visão e diferentes condições de luminosidade.

O sexto método abordado utiliza de técnicas de *deep-learning* os quais tem sido extremamente difundidas no assunto de visão computacional. A utilização de redes neurais convolucionais (*Convolutional Neural Network - CNN*) aliadas ao pre-processamento da imagem para redução de ruídos e redução de objetos menores atingem uma assertividade entre 91,3 e 93,8%, outro exemplo desse método é a biblioteca de redes neurais convolucionais *YOLO* (REDMON et al., 2016), iniciais de *You Only Look Once*, onde uma rede neural convolucional baseada em região (R-CNN) gera regiões com potenciais objetos logo após classifica cada uma dessas regiões, na sequencia são refinadas as bordas de cada região e eliminadas regiões duplicadas. O estado da arte em R-CNN, *YOLO*, tem fornecido as ferramentas da forma mais simplificada possível para aplicação em sistemas robustos detecção de placas de identificação de veículos (LAROCA et al., 2018).

2.1.3 Métodos de Reconhecimento de Caracteres

Nesta subseção serão apresentados métodos específicos de reconhecimento de caracteres como abordado por Du et al. (2013). Iniciando-se pelo método de segmentação usando a conexão entre pixels, neste método são analisados os pixels com o mesmo tamanho e formato de caracteres, o método não é muito efetivo quando os caracteres estão conectados ou quebrados.

A segmentação usando a projeção de perfis usa da teoria em que a cor dos caracteres e a cor de fundo da placa de identificação são diferentes, logo terão valores binários opostos, após é realizada uma projeção vertical deste binário como intuito de determinar a posição inicial e final de cada carácter, para então projetar cada uma dessas regiões horizontalmente para extrair cada carácter.

Na segmentação utilizando o conhecimento prévio de caracteres, o binário da imagem é escaneado horizontalmente em busca das posições iniciais e finais de cada carácter, quando a proporção entre os pixels do carácter e os pixels do fundo da placa de identificação excede um limiar depois de ser inferior a esse limiar então é considerado como sendo o inicio de um carácter o oposto também é valido para a identificação do final de um carácter, com essa etapa concluída a placa de identificação veicular é redimensionada para o tamanho padrão, como a posição de cada carácter em uma placa padrão é conhecida é realizada a segmentação em caracteres. Esse método traz a simplicidade ao segmentação de caracteres, porém qualquer mudança causada a placa de identificação ao ser redimensionada pode ocasionar a segmentação de forma errônea.

No método de segmentação utilizando-se do contorno de caracteres, utiliza de técnicas para definição do contorno dos caracteres em conjunto com equações de diferencial parcial de valor inicial para uma função de conjunto de níveis de propagação aliadas as leis de conservação

hiperbólicas, conseguindo dessa forma identificar alterações topológicas e contornos (SETHIAN, 1996). Este método é considerado lento e propenso a gerar contornos incompletos ou distorcidos (DU et al., 2013).

Com a segmentação dos caracteres abordada o próximo passo é a realização do reconhecimento de cada carácter, para esse desafio Yanque (2018) aborda os métodos de binarização heurística e métodos baseados em aprendizagem de máquina. Para o método de binarização heurística a autora realiza a classificação em subcategorias sendo elas o método global, método local e o método híbrido.

No método global o limiar é determinado através da análise do histograma da escala em cinza da imagem, buscando o mínimo ou máximo de uma função sobre o histograma, esse método tem a vantagem de ser extremamente rápido, porém sofre muito com a presença de ruídos na imagem.

No método local é calculado o limiar para cada pixel da imagem considerando as informações dos pixels adjacentes, corrigindo até certo ponto o problema encontrado no método global, a desvantagem desse método se apresenta ao trabalhar com imagens com iluminação desigual, cores similares entre caracteres e o fundo além de fundos com muito ruído.

Os métodos híbridos utilizam de informações do método global aliados ao método local, esses métodos são utilizados para solução de casos mais complicados. Esses métodos foram reclassificados no artigo de Ismail, Abdullah e Fauzi (2018), conforme descritos abaixo.

Baseado na forma do histograma: neste método são utilizadas as propriedades suavizadas do histograma.

Baseado em agrupamento: neste método são agrupados os níveis de cinza do carácter e do fundo, modelando as amostras com uma mistura de dois gaussianos.

Baseado em entropia: neste método são utilizadas as entropias das áreas do carácter e do fundo, bem como a entropia de cruzamento entre a imagem original e binária.

Baseado nos atributos do objeto: neste método são analisadas as similaridades entre a escala de cinza e a imagem binária.

Método espacial: neste método é utilizada a distribuição de probabilidade de ordem superior e/ou correlação entre pixels.

Método local: neste método são adaptados o valor da limiar em cada pixel de acordo com as características locais da imagem.

No artigo da Yanque (2018) ainda são abordados os métodos baseados em aprendizagem de máquina, esses métodos são o método de aprendizagem não supervisionada, onde o método agrupa cada um dos pixels em três distintas classes: o texto, o fundo e o incerto, a classe de incertos ainda é classificada entre texto e fundo de acordo com medida de distância entre as duas classes.

Já na aprendizagem supervisionada, são observados cada pixel, extraídas características locais e utilizadas no algoritmo de aprendizagem. A partir daí é gerado o mapeamento que decide em qual classe (texto ou fundo) o pixel será considerado.

Na aprendizagem profunda são utilizadas redes neurais convolucionais (CNN) e redes

neurais recorrentes (RNN) abordadas por Goodfellow et al. (2016) assim como redes profundas supervisionadas (DSN) hierárquica (VO et al., 2018) para a binarização de imagens. As grandes vantagens desse método são: a facilidade de execução, não necessitar de pré e/ou pós processamento e após treinadas, as redes neurais, o processo é rápido.

2.2 FERRAMENTAS E BIBLIOTECAS

Nesta seção serão abordados fundamentos teóricos relacionados a linguagem de programação escolhida e suas bibliotecas.

2.2.1 Python

No âmbito de desenvolvimento que exige confiabilidade, velocidade de desenvolvimento, fácil compreensão e manutenção a linguagem *Python* tem se destacado, tendo sido criada nos anos 90 pelo pesquisador Guido van Rossum, sendo desde seu princípio *open source*, podendo ser utilizada inclusive em desenvolvimentos comerciais. O desenvolvimento da linguagem é constante e desde 2001 é mantida pela *Python Software Foundation* responsável pela propriedade intelectual da linguagem (PSF, 2022). A utilização da linguagem *python* ganhou popularidade a partir de 2018 e tem estado entre as 20 linguagens de programação mais populares segundo TIOBE (2022). A utilização da mesma tem tido diversas aplicações desde simples aplicações de cunho educacional a desenvolvimentos extremamente complexos como o mecanismo de pesquisa do *Google*, serviço de compartilhamento de vídeo do *Youtube* além de desenvolvimento de tarefas científicas na *NASA*, *Los Alamos*, *Fermilab*, *JPL* entre outras inúmeras aplicações (SRINATH, 2017).

2.2.2 Labelimg - Libxml2

A ferramenta *Labelimg* foi criada em 2015 por Tzutalin, escrita em *Python*. A definição de contornos e legendas para cada objeto de interesse na imagem é fundamental para o treinamento de modelos de aprendizagem de máquina, utilizados em projetos de visão computacional, sendo capazes de reconhecer objetos simples como uma casa de cachorro em uma imagem até a distinção de um Husky Siberiano e um Pinscher. A utilização do *Labelimg* torna-se facilitada em função de sua interface gráfica, retornando arquivos em formatos de texto com as especificações necessárias o posterior treinamento de modelos (BOESCH, 2022).

2.2.3 Darknet e YOLOv3

O treinamento de um modelo de inteligência artificial é sem dúvidas um pré requisito com grandes desafios. E nesse quesito a utilização da biblioteca *Darknet* desenvolvida na linguagem C e CUDA, é considerada a biblioteca de treinamento mais rápida do mercado, em

seu modo de treinamento utilizando GPU (placa de vídeo) ela se torna 500x mais rápida que em modo CPU, utilizando apenas o processador do computador, para realização dos cálculos matemáticos responsáveis pelo treinamento do modelo com pesos(*weights*) de cada nó da rede neural convolucional. A utilização da biblioteca YOLOv3 (*You Only Look Once* versão 3) permite a utilização de modelos treinados na detecção de objetos em tempo real, considerada o estado da arte na detecção de objetos o *YOLO* alcança a marca de ser 1000x mais rápida na detecção de objetos do que a obsoleta biblioteca *Fast R-CNN*(*Fast Region-based Convolutional Network*) utilizando um *dataset* muito menor de imagens. (REDMON et al., 2016)

3. METODOLOGIA

Neste capítulo são apresentados os métodos a serem utilizados no desenvolvimento do sistema final. Na primeira seção será abordada a metodologia na captura das imagens, definindo qual o meio físico de obtenção das imagens, ferramenta de captura e algoritmo para armazenagem e análise das imagens obtidas. Na subseção 3.1.3 é explicitado o método de detecção de movimento nas imagens obtidas na seção anterior, realizando a análise de regiões de interesse nas imagens, com o intuito de reduzir o tempo de processamento e a detecção de movimentos ruidosos. Nas subseções 3.1.4 e 3.1.5 são apresentadas as ferramentas utilizadas para detecção de veículos, placas de identificação veicular e o posterior reconhecimento de caracteres. Na última subseção deste capítulo é definido o algoritmo de análise de anomalias a ser usado na elaboração do sistema. O fluxo da metodologia proposta é apresentada na Figura 1.

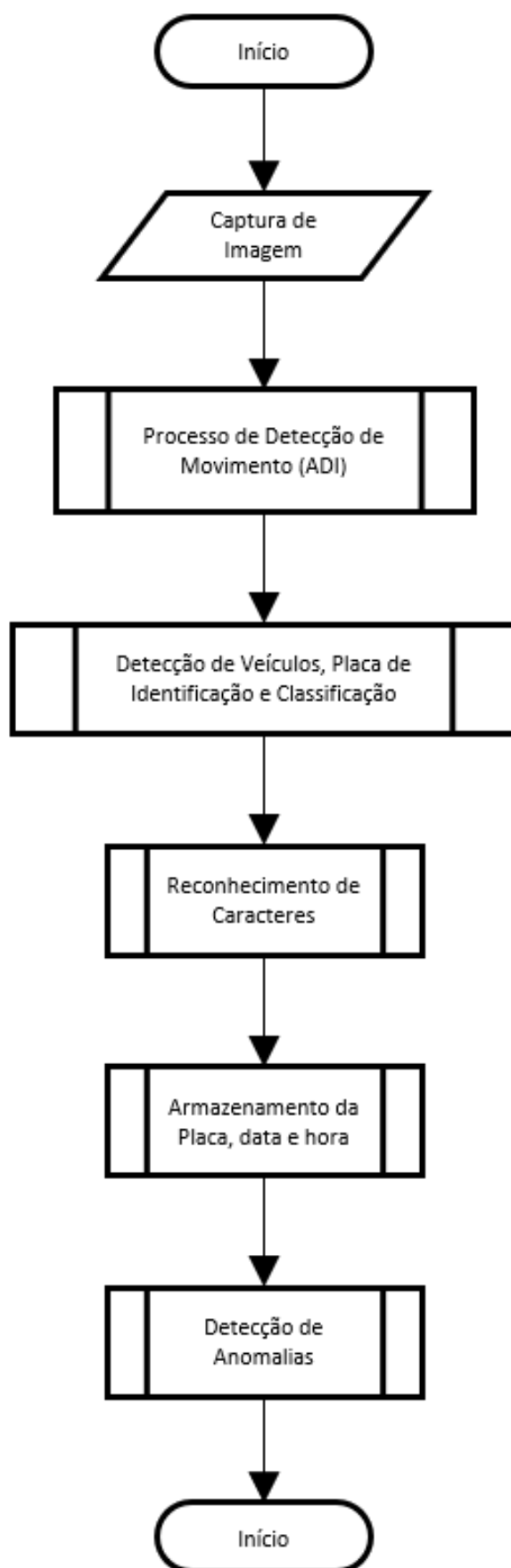


Figura 1 – Fluxo do Sistema

3.1 CAPTURA DE IMAGENS

3.1.1 Meio Físico de Captura

Nos atuais sistemas de vigilância é amplamente difundida a utilização de *Network Video Recorder* (NVR), câmeras analógicas, câmeras IP e controle remoto através da internet. O aumento na utilização destes equipamentos impulsionaram a tecnologia aplicada a eles, desde os aparelhos *DVR's* com aumento na capacidade de armazenagem e controle de um número ainda maior de câmeras, até a evolução dos sensores ópticos utilizados nas câmeras reduzindo seu tamanho e aumentando a qualidade das imagens em diferentes gamas de luminosidade.

O padrão de instalação existente na grande maioria dos casos residenciais, utiliza-se de câmeras analógicas conectadas ao *Digital Video Recorder* (DVR), o qual realiza o gerenciamento e gravação dos equipamentos a ele conectados. Através do *DVR's* é possível a transmissão das imagens de vídeo via internet, sendo acessadas em qualquer lugar com conexão disponível. Já as câmeras analógicas são portadoras de baixa tecnologia e o seu custo é bem acessível, porém a necessidade de um equipamento como o *DVR* para transmissão das imagens capturadas eleva o valor do conjunto consideravelmente.

As câmeras IP, também conhecidas como câmeras de rede, contam com uma estrutura de rede, dessa forma ela apresenta o seu próprio endereço IP, sendo de instalação de forma simples e sem a necessidade de ferramentas adicionais. Seu gerenciamento, visualização e gravação, necessitando de um *NVR* no ultimo caso, é realizado através da rede em que a mesma estiver conectada. As mesmas contam com processador de imagem que permite a transmissão de imagens através de protocolo *FTP*, acesso via internet, disparo de e-mail, detecção de movimentos entre outras funcionalidades. Tendo seu valor superior aos valores aplicados a câmeras analógicas, porém com o acesso podendo ser realizado sem a necessidade de um *DVR* o valor torna-se compatível e com maior facilidade de instalação.

Côncio dos valores, forma de instalação, tecnologia aplicada, em constante evolução, e a facilidade de acesso via rede a utilização de câmeras IP com qualidade de imagem *Full HD*, ou superior, na aplicação desde trabalho serão os pontos determinantes na escolha do modelo da câmera a ser utilizada. Para o presente trabalho serão utilizadas imagens capturadas em diversas condições ambientais por câmeras de vigilância instaladas em guichês de pagamento na rodovia ES-060, Espírito Santo, operada pela concessionária Rodovia do Sol (Laroca et al., 2022) e armazenadas em ambiente controlado.

3.1.2 Ferramenta de Captura e Processamento das Imagens

A utilização de sistema computadorizado para captura de imagens é amplamente utilizado, partindo de computadores a sistemas embarcados como *Raspberry Pi* (LEVEZ; BENINI; GOMES, 2018), em ambos os casos a utilização da biblioteca de manipulação de imagens *OpenCV* é unanimidade. A biblioteca *open source OpenCV*, conta com mais de 500 funções que abrangem

várias áreas da visão computacional, além de contar com biblioteca de aprendizagem de máquina baseada em reconhecimento de padrões estatísticos e agrupamento. Visão computacional é a transformação de dados de uma imagem ou vídeo em decisões ou em uma nova representação, que pode ser a criação de uma imagem em escala de cinza ou a remoção de movimento da câmera em uma sequência de quadros. A imagem recebida na visão computacional diferentemente da imagem recebida pelo cérebro, é composta apenas de uma tabela de números que trazem diversos componentes ruidosos que fornecendo poucas informações relevantes. Esses ruídos podem vir de várias formas, como variações climáticas, imperfeições nas lentes e até ruídos elétricos no sensor captor. As ações e/ou decisões realizadas a partir de dados coletados pela visão computacional são executadas em um contexto específico para cada problema. A biblioteca *OpenCV* conta com as ferramentas básicas para resolver os mais diversos problemas de visão computacional e em alguns casos as funcionalidades de alto nível são o suficiente para resolver os problemas mais complexos na área (BRADSKI; KAEHLER, 2008).

Ciente da grande abrangência apresentada pela biblioteca *OpenCV*, a utilização da mesma é indispensável para a conclusão do presente trabalho. Na Figura 2, é apresentado o fluxo geral deste trabalho dando ênfase as tecnologias aplicadas.

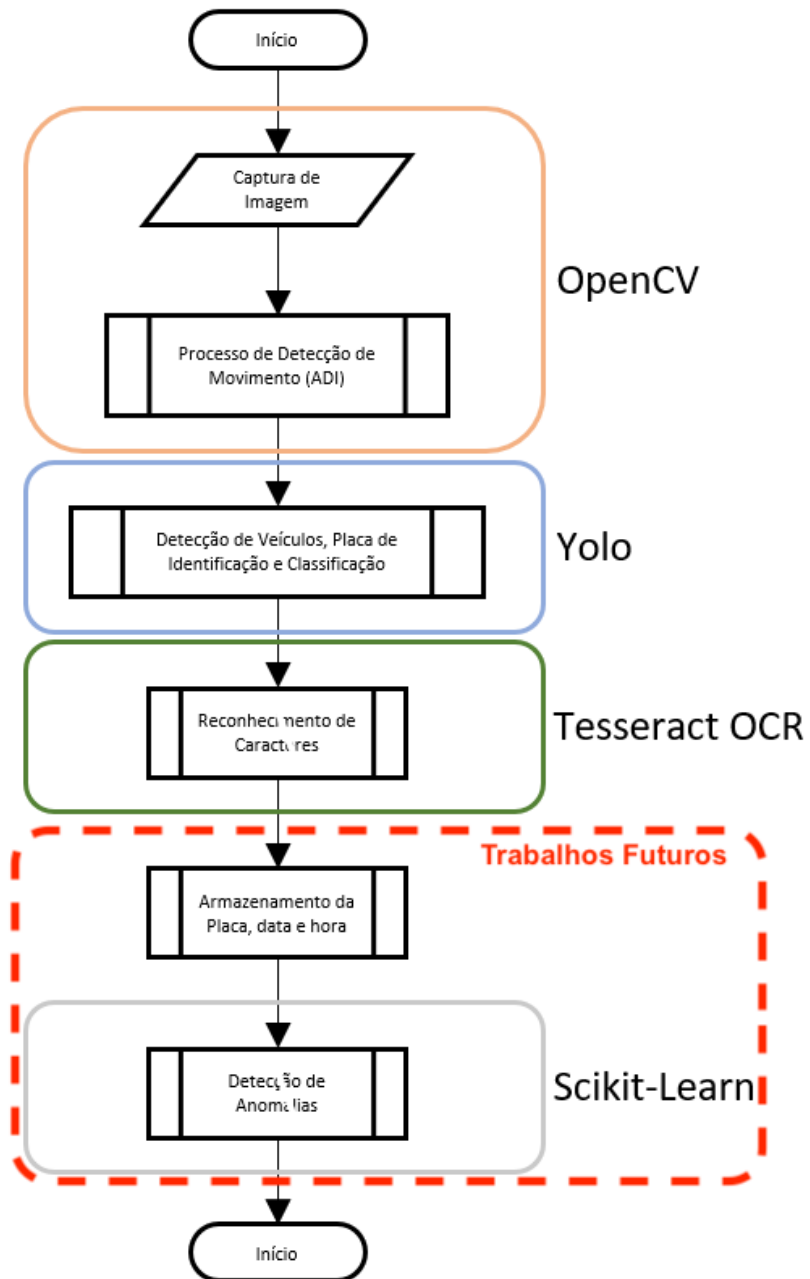


Figura 2 – Fluxo do Sistema e suas Tecnologias

3.1.3 Detecção de Movimento

Como descrito na Seção 2.1.1 inúmeras são as técnicas de detecção de movimento, em função da execução do trabalho ser realizada em ambiente externo e sem a possibilidade de controle de luminosidade, a utilização do método básico de detecção de movimento apresenta inúmeros problemas relacionados a alternância de luminosidade no ambiente, gerando falsos positivos para movimentos, sendo assim a utilização do método acumulativo de diferença é a escolha que traz os melhores resultados sendo que ele é gerado a partir de uma acumulação de diferença entre quadros reduzindo a interferência de luminosidade e objetos com movimento cíclico dentro dos quadros do vídeo. O método de detecção por acumulo de diferenças tem seu fluxo de execução conforme a Figura 3.

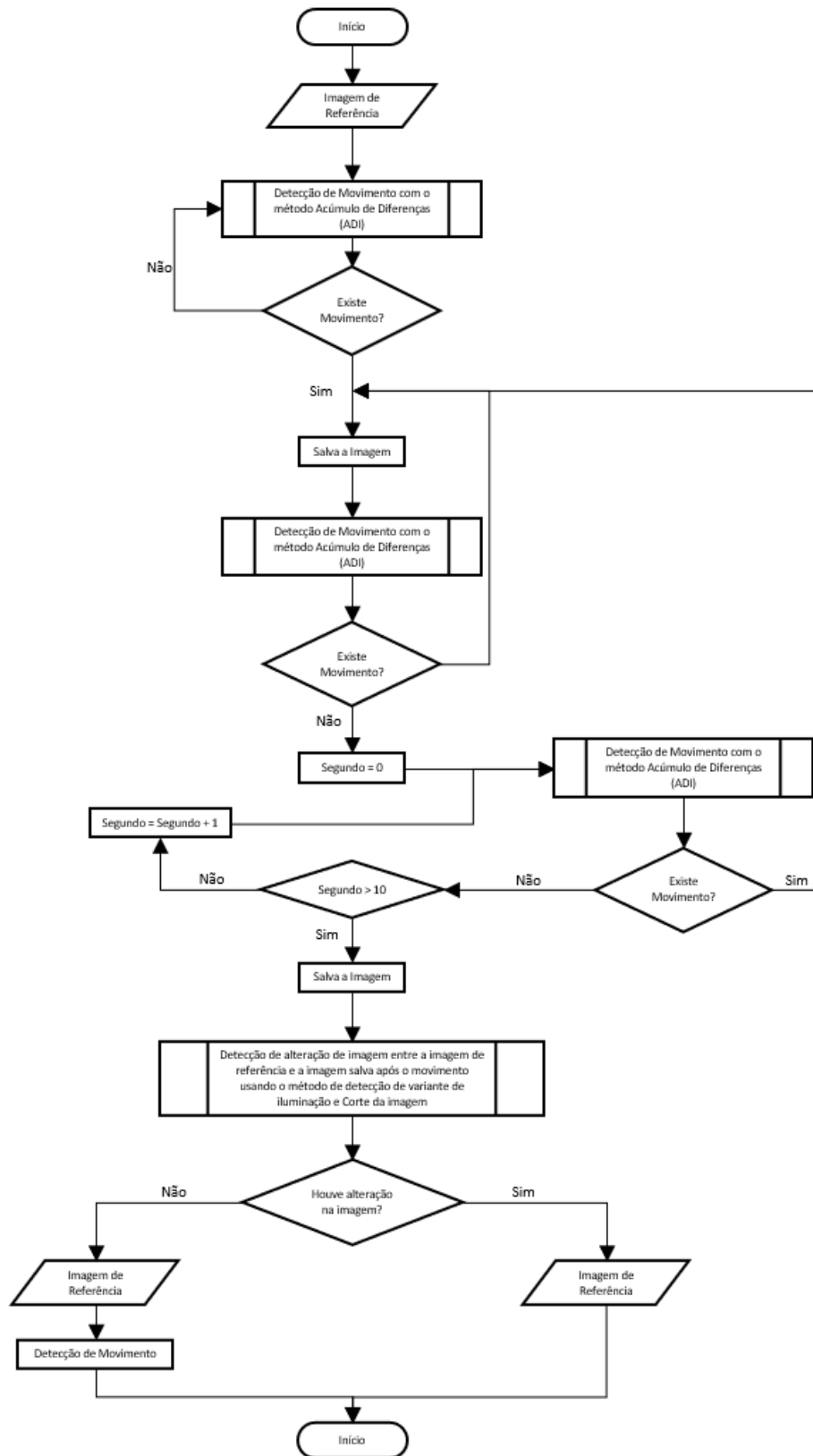


Figura 3 – Fluxo do Sistema de Diferenças Acumulativas das Imagens (PRIADANA; HARJOKO, 2017)

3.1.4 Detecção de Veículos e Placas de Identificação Veicular

A utilização de bibliotecas de redes convolucionais como o *YOLO* é amplamente difundida na área de reconhecimento de veículos e placas de identificação, tendo uma grande quantidade de *datasets* para treinamento das redes neurais. Os resultados da utilização da biblioteca *YOLO* são apresentadas no artigo escrito por Laroca et al. (2021), onde o mesmo utiliza da biblioteca tanto para a detecção de veículos quanto na detecção de placas de identificação veicular e sua classificação. A classificação da placa de identificação é realizada no intuito de identificar o modelo e país de origem da mesma, facilitando o reconhecimento de caracteres o qual será abordado na subseção 3.1.5. O fluxo de trabalho a ser adotado na detecção de veículos e placas de identificação é apresentado na Figura 4.

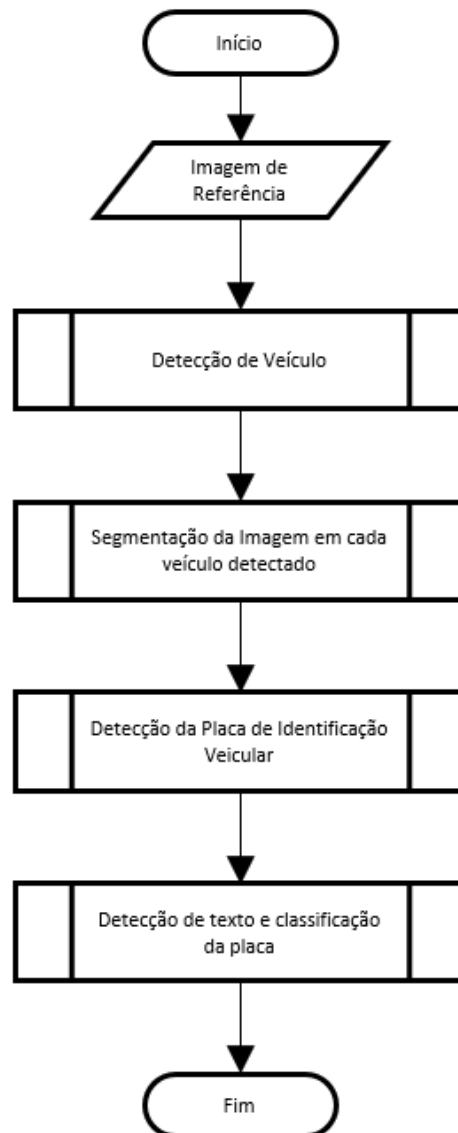


Figura 4 – Fluxo de Detecção de Veículos e Placas de Identificação Veicular (LAROCA et al., 2021)

3.1.5 Reconhecimento de Caracteres

A utilização de redes neurais aliadas ao reconhecimento óptico de caracteres com o auxílio da biblioteca *Tesseract OCR* teve seu início nos meados de 1985, sendo mantida até 2005 pela empresa *Hewlett-Packard*, recebendo grande auxílio, na evolução do sistema, da comunidade de desenvolvedores ao redor do mundo. Entre 2006 e 2018 o desenvolvimento da mesma foi realizado pelo time de desenvolvedores da *Google* (SMITH, 2021).

Conforme apresentado na subseção anterior a utilização da biblioteca *YOLO* traz a função de detectar texto e realização da classificação do tipo de placa de identificação visível na imagem. A classificação do tipo de placa é de extrema necessidade no processo de reconhecimento de caracteres, pois para cada tipo de placa de identificação existe um formato de apresentação e disposição de letras e números, tendo esses tipos em mãos é reduzido tempo de processamento e erros de similaridade entre letras e números, como nos casos de Q-0 e G-0, aumentando dessa forma a assertividade do modelo. Para o presente trabalho será utilizada a imagem e sua classificação resultantes da subseção 3.1.4 para o reconhecimento de caracteres a ser realizado pela biblioteca *Tesseract OCR*, com padrões de disposição letras e números conforme as placas de identificação presentes no Brasil, em que o padrão de disposição é dividido em dois grupos, a placa de identificação do Brasil formada por 3 (três) letras e 4 (quatro) números no formato LLL-NNNN e a placa de identificação do Mercosul formada por 4 (quatro) letras e 3 (três) números no formato LLLNLNN. Cômico destes formatos a definição de padrões de reconhecimento de caracteres torna-se mais assertiva. Na Figura 5 são demonstrados os modelos de placas de identificação veicular do tipo Mercosul e na Figura 6 são demonstrados os modelos do tipo padrão reflexivo.



Figura 5 – Placas de Identificação Veicular conforme Resolução N°780/2019 (NACIONAL, 2019)



Figura 6 – Placas de Identificação Veicular conforme Resolução N°372/2011 (NACIONAL, 2011)

3.1.6 Análise de Situação Cadastral do Veículo

O ponto chave do presente trabalho está na detecção de veículos suspeitos, para tanto a análise de veículos com situação cadastral como alerta furto/roubo é essencial para atendimento final dos objetivos deste trabalho. A verificação da situação cadastral é realizada no sistema disponibilizado pelo SENATRAN, Secretaria Nacional de Trânsito. Em função da quantidade de requisições e critérios de segurança utilizados nos sistemas do Governo Brasileiro, impedindo desta forma a busca direta da situação cadastral de veículos.

O método utilizado nesse ponto então será através de API, *Application Programming Interface*, de terceiro o qual tem acesso ao sistema SENATRAN de forma homologada e autorizada.

3.2 FERRAMENTAS

Ao findar deste capítulo são elencadas as ferramentas utilizadas em cada etapa do desenvolvimento deste trabalho. Iniciando a apresentação das ferramentas partimos do ponto desafiador deste trabalho a complexidade de treinamento de modelos e definição dos pesos utilizados posteriormente na análise das imagens. Para este processo foram utilizadas as seguintes ferramentas:

- *LabelImg -Libxml2*
Ferramenta responsável pela categorização e definição das bordas das placas veiculares, de forma manual(LIBXML2, 2022), nas imagens utilizadas posteriormente no treinamento.
- *Darknet*
Framework open source de rede neural escrito em C e CUDA.(REDMON, 2013–2016)
Responsável pelo treinamento do modelo e definição dos pesos(*weights*) utilizados durante a validação e produção do sistema.
- *Google Colab*
Plataforma *online* completa para desenvolvimento e pesquisa em aprendizagem de máquina e inteligência artificial.
- *Python CE*
Interface de desenvolvimento completa especializada na linguagem *Python*.(PyCharm. . . ,)
- *Opencv-python(Open Source Computer Vision Library)*
Biblioteca que proporciona a infraestrutura para o desenvolvimento de visão computacional e aprendizagem de máquina.(BRADSKI; KAEHLER, 2008)

- *Tesseract-ocr*
Biblioteca especializada em reconhecimento de texto, tendo em sua base os pesos necessários para reconhecimento de caracteres em diversos idiomas.(SMITH, 2021)
- *ApiPlacas*
Application Programming Interface para aquisição da situação cadastral dos veículos.

de forma manual. O processo de categorização torna-se penoso em função do tamanho do *dataset* de imagens, sendo no caso específico 20000 imagens. O processo para cada imagem do pacote de imagens é exatamente o mesmo, sendo necessária a abertura da imagem, definição das bordas do objeto que se pretende "encontrar" na imagem - a placa veicular no caso deste trabalho, e por fim a categorização. Para cada imagem do *dataset* é criado um arquivo, em formato de texto, contendo o índice da categoria e a posição de cada vértice do retângulo que envolve o objeto em questão. Abaixo é demonstrado o processo descrito neste parágrafo.

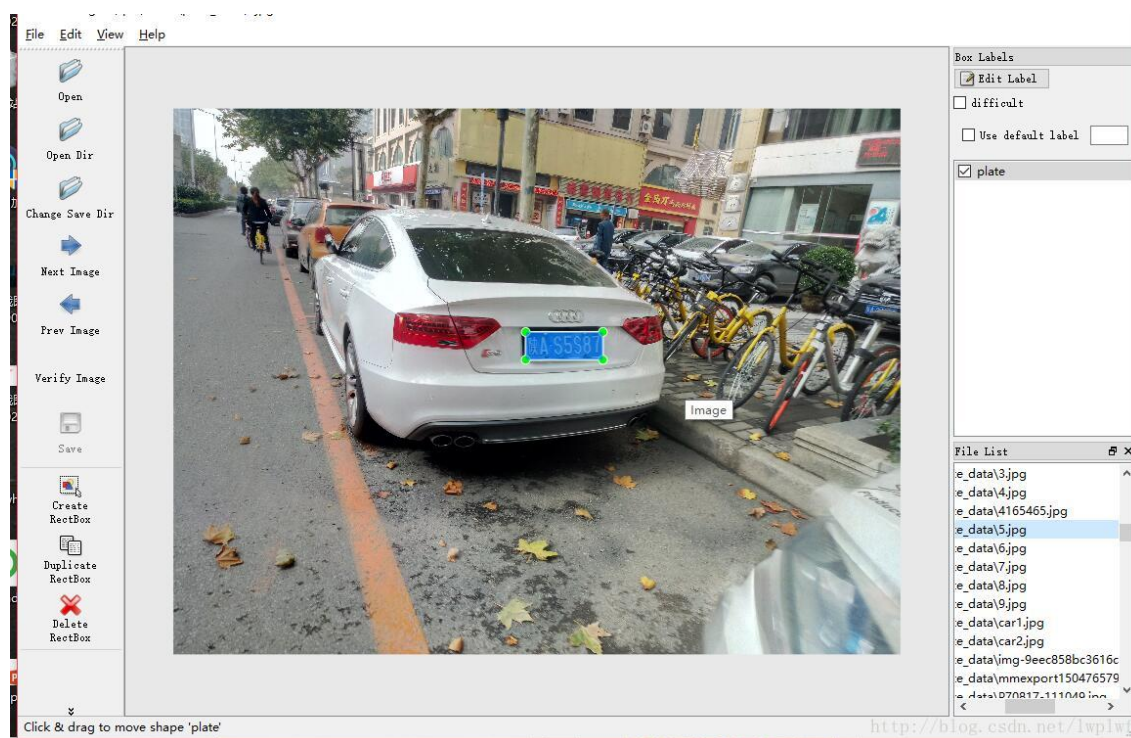


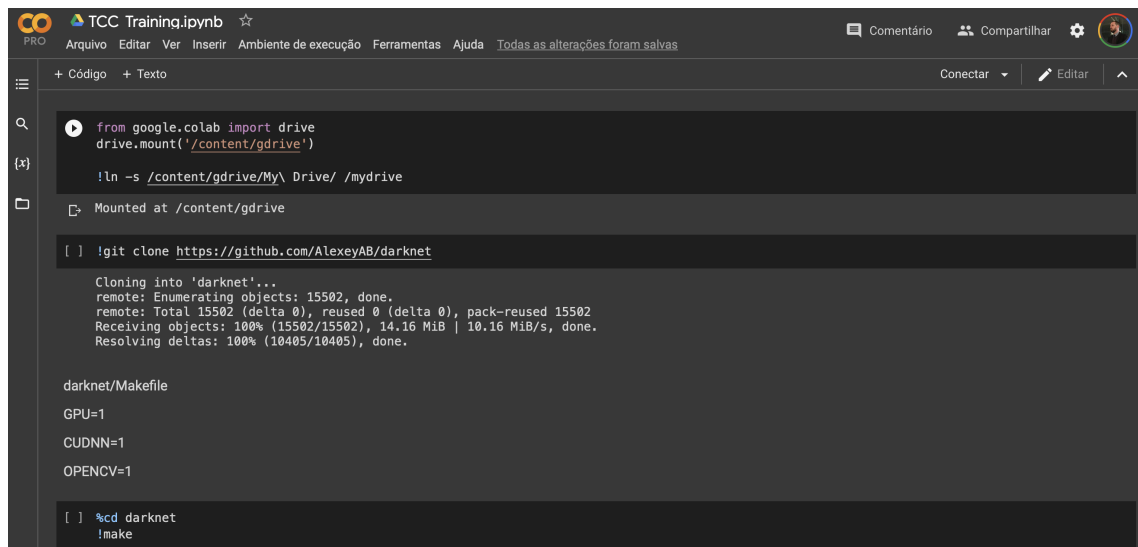
Figura 8 – Exemplo de uso do LabelImg

Finalizado o processo de categorização das imagens, é realizado o treinamento do modelo e definição dos pesos (*weights* para cada nó da rede neural. Neste processo foi de fundamental importância a utilização da plataforma *Google Colab*, plataforma desenvolvida para fomentar a aprendizagem e pesquisa relacionada a aprendizagem de máquina e inteligência artificial. Neste processo foi necessária a aquisição de um plano intermediário, conhecido como *Colab Pro*, em função do tamanho do *dataset* de imagens de treinamento, que para o propósito deste trabalho foi de 1215 imagens. Para este plano intermediário são disponibilizados 100 créditos de computação em uma configuração de hardware que conta com 12Gb de ram e placa de vídeo T4 com 16Gb de memória. Para o processo de treinamento foi utilizado o *framework Darknet*, sendo necessários alguns ajustes no arquivo de configuração do mesmo para que o processo de treinamento fosse realizado através de computação em *GPU* com a utilização da biblioteca *OpenCV*. Após a configuração do *Darknet* realizada, realizou-se a configuração do ambiente de treinamento, onde são informados ao *Darknet*, qual biblioteca de treinamento se deseja utilizar durante o treinamento, no caso deste trabalho utilizou-se a biblioteca *YOLOv3*, sendo necessários alguns ajustes no arquivo de configurações da mesma, ajustando parâmetro de quantidade de

filtros e classes a serem treinadas.

O carregamento das imagens de treinamento é realizada e posteriormente é iniciado o processo de treinamento para as 1215 imagens, com as configurações de *hardware* e das bibliotecas já mencionadas, foram necessárias 20 horas e 1 minuto para finalizar o processo de treinamento do modelo. Neste processo é aconselhável que os arquivos com os pesos de cada nó sejam salvos em um local diferente do ambiente de treinamento, pois o ambiente é completamente excluído assim que a plataforma *Google Colab* é finalizada. Com este processo finalizado temos o arquivo de extensão *.weights* com o peso de cada nó da rede neural, arquivo este que será utilizado posteriormente no processo de extração da placa veicular da imagem a ser analisada. A seguir é demonstrado o processo de treinamento de modelo utilizando a plataforma *Google Colab*.

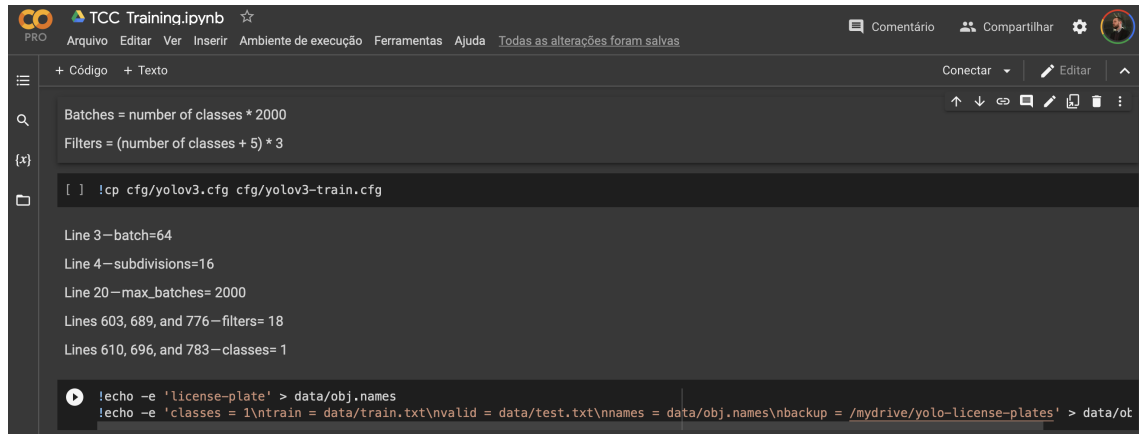
Primeiro passo necessário é a integração entre o o *Google Drive* e o *Google Colab* com o intuito de armazenar os modelos treinados no futuro. Em seguida é realizada a "clonagem" do repositório do *Darknet*, rede neural que utilizaremos para o treinamento. Na Figura 9 é demonstrado o processo descrito com algumas alterações necessárias no arquivo *Makefile*, responsável por informar a rede neural que estaremos utilizando a *GPU* e *OPENCV*, após essa etapa é criado o arquivo de configuração.



```
TCC Training.ipynb
Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Todas as alterações foram salvas
+ Código + Texto Conectar Editar
from google.colab import drive
drive.mount('/content/gdrive')
!ln -s /content/gdrive/My Drive/ /mydrive
Mounted at /content/gdrive
[ ] !git clone https://github.com/AlexeyAB/darknet
Cloning into 'darknet'...
remote: Enumerating objects: 15502, done.
remote: Total 15502 (delta 0), reused 0 (delta 0), pack-reused 15502
Receiving objects: 100% (15502/15502), 14.16 MiB | 10.16 MiB/s, done.
Resolving deltas: 100% (10405/10405), done.
darknet/Makefile
GPU=1
CUDNN=1
OPENCV=1
[ ] %cd darknet
!make
```

Figura 9 – Plataforma *Google Colab* - Instalação/Configuração do *framework Darknet*

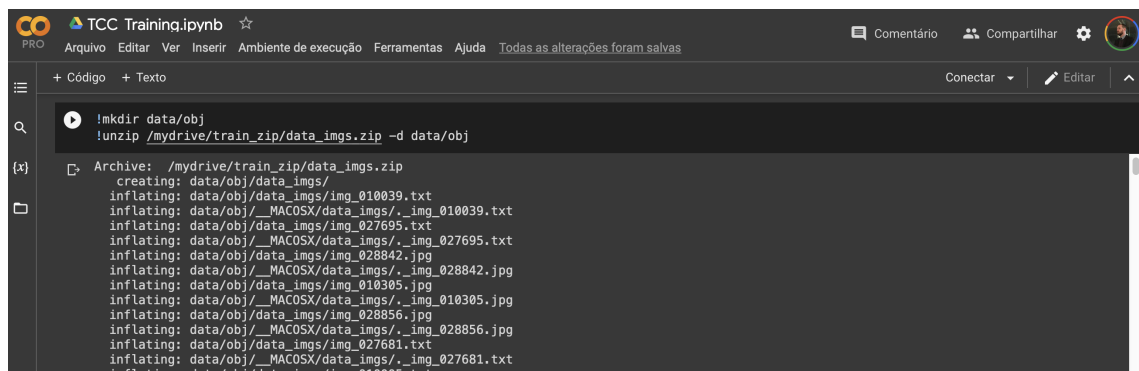
O passo seguinte, demonstrado pela Figura 10, é a configuração da biblioteca *Yolov3*, repassando a quantidade de classes e inferências esperados durante o treinamento. Na sequência é definido o endereço de onde o modelo treinado deve ser salvo.



```
TCC Training.ipynb
Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Todas as alterações foram salvas
+ Código + Texto
Batches = number of classes * 2000
Filters = (number of classes + 5) * 3
[ ] !cp cfg/yolov3.cfg cfg/yolov3-train.cfg
Line 3--batch=64
Line 4--subdivisions=16
Line 20--max_batches= 2000
Lines 603, 689, and 776--filters= 18
Lines 610, 696, and 783--classes= 1
!echo -e 'license-plate' > data/obj.names
!echo -e 'classes = 1\ntrain = data/train.txt\nvalid = data/test.txt\nnames = data/obj.names\nbackup = /mydrive/yolo-license-plates' > data/ot
```

Figura 10 – Plataforma *Google Colab* - Configuração da biblioteca *YOLOv3*

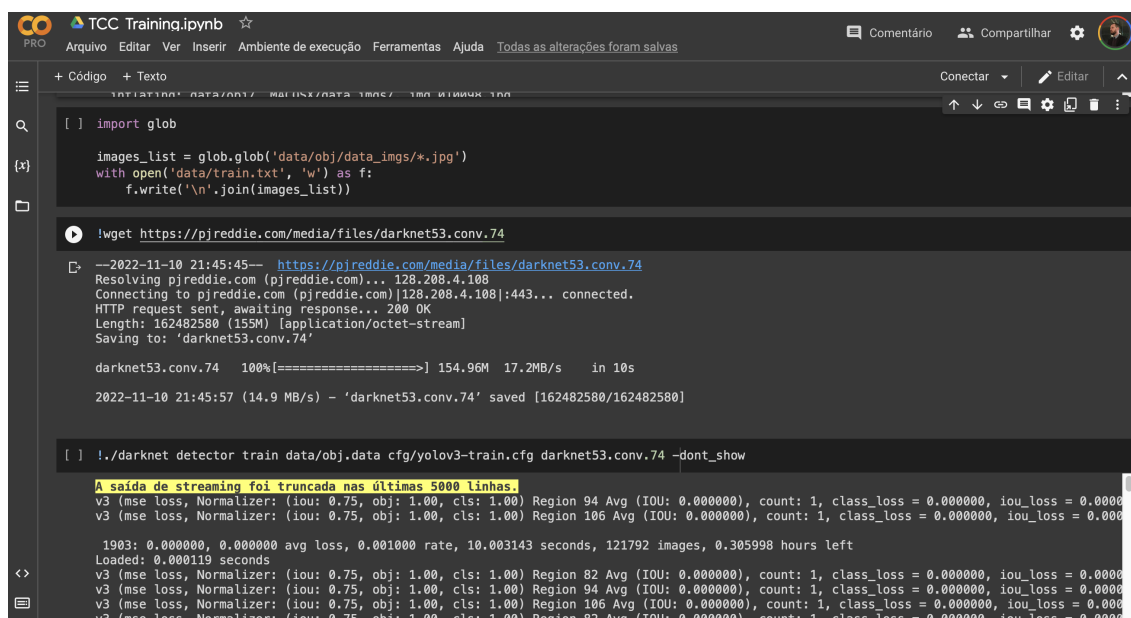
Na Figura 11 é dado sequência ao processo de treinamento onde é realizada a descompactação das imagens de treinamento e seus respectivos arquivos de categorização.



```
TCC Training.ipynb
Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Todas as alterações foram salvas
+ Código + Texto
!mkdir data/obj
!unzip /mydrive/train_zip/data_imgs.zip -d data/obj
Archive: /mydrive/train_zip/data_imgs.zip
creating: data/obj/data_imgs/
inflating: data/obj/data_imgs/img_010039.txt
inflating: data/obj/_MACOSX/data_imgs/.img_010039.txt
inflating: data/obj/data_imgs/img_027695.txt
inflating: data/obj/_MACOSX/data_imgs/.img_027695.txt
inflating: data/obj/data_imgs/img_028842.jpg
inflating: data/obj/_MACOSX/data_imgs/.img_028842.jpg
inflating: data/obj/data_imgs/img_010305.jpg
inflating: data/obj/_MACOSX/data_imgs/.img_010305.jpg
inflating: data/obj/data_imgs/img_028856.jpg
inflating: data/obj/_MACOSX/data_imgs/.img_028856.jpg
inflating: data/obj/data_imgs/img_027681.txt
inflating: data/obj/_MACOSX/data_imgs/.img_027681.txt
inflating: data/obj/data_imgs/img_010005.txt
```

Figura 11 – Plataforma *Google Colab* - Carregamento das imagens de treinamento

Antes de darmos início ao treinamento é necessária a criação de um arquivo que conterá todos os endereços das imagens e seus arquivos de categorização, conforme demonstrado na Figura 12. Em seguida é realizado o *download* do arquivo com a estrutura da rede neural padrão a ser utilizada. Por fim são passados os arquivos de configuração a rede neural a ser treinada, iniciando o processo logo em seguida.



```
[ ] import glob

images_list = glob.glob('data/obj/data_imgs/*.jpg')
with open('data/train.txt', 'w') as f:
    f.write('\n'.join(images_list))

!wget https://pjreddie.com/media/files/darknet53.conv.74

--2022-11-10 21:45:45-- https://pjreddie.com/media/files/darknet53.conv.74
Resolving pjreddie.com (pjreddie.com)... 128.208.4.108
Connecting to pjreddie.com (pjreddie.com)|128.208.4.108|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 162482580 (155M) [application/octet-stream]
Saving to: 'darknet53.conv.74'

darknet53.conv.74 100%[=====] 154.96M 17.2MB/s in 10s
2022-11-10 21:45:57 (14.9 MB/s) - 'darknet53.conv.74' saved [162482580/162482580]

[ ] !./darknet_detector train data/obj.data cfg/yolov3-train.cfg darknet53.conv.74 -dont_show

A saída de streaming foi truncada nas últimas 5000 linhas.
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000)
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000)
1903: 0.000000, 0.000000 avg loss, 0.001000 rate, 10.003143 seconds, 121792 images, 0.305998 hours left
Loaded: 0.000119 seconds
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000)
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000)
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000)
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000)
```

Figura 12 – Plataforma *Google Colab* - Carregamento das imagens de treinamento

Para o desenvolvimento do sistema de verificação da situação cadastral do veículo analisado foi utilizada a interface de desenvolvimento *PyCharm Community*, interface gratuita especializada em desenvolvimento e geração de ambiente para programação na linguagem *Python*. Para o desenvolvimento deste sistema foi necessário a instalação da biblioteca *opencv-python*. A biblioteca agora citada, necessitaria de um capítulo por completo neste trabalho para sua correta apresentação e demonstração de todas as suas funcionalidades, porém para este trabalho foram utilizados os módulos de leitura e tratamento de imagens e o módulo de *Deep Neural Network (DNN)* responsável pela análise da imagem.

Com o processo de instalação do *opencv-python* concluída, foi realizada a configuração do módulo *DNN*, o qual será responsável pela análise da imagem, testando o modelo em cada segmento da imagem em busca do padrão estipulado retornando para cada região encontrada um nível de confiança, a partir deste nível de confiança é definida se foi encontrada a placa veicular na imagem ou não, para este trabalho o limiar de confiança (*confidence threshold*) foi estipulado em 0.5, desta forma onde o modelo detectar uma região com um limiar de confiança acima do valor estipulado e mais próximo de 1 será assumido com uma placa veicular. Através da lógica atribuída neste trabalho, o sistema desenha cada uma das regiões encontradas, fazendo a verificação quando ao limiar de confiança no final do processo, removendo da memória o restante das regiões com limiar de confiança inferior ao estipulado. Abaixo é demonstrado um exemplo de detecção da placa veicular junto com a confiança calculada pelo *DNN*.



Figura 13 – Detecção de placa veicular com a confiança calculada pelo *DNN*

Conforme apresentado na Figura 13 pode-se verificar que o modelo encontrou a placa veicular na imagem com uma confiança de 0.929, valor considerado aceitável dentro dos estudos de visão computacional (Laroca et al., 2022). Após este momento é realizado o recorte desta região sendo salvo como uma imagem para posterior tratamento. Este tratamento amplia a região em 4x utilizando interpolação cúbica fazendo com que a imagem aumente de tamanho e mantenha a qualidade de imagem. Após esse procedimento é realizada a conversão da imagem em uma imagem em preto e branco, facilitando que o a biblioteca *opencv-python* encontre os limites de cada região dentro da imagem. Realizado este processo, é realizada a conversão destes limites em uma combinação binária com posterior inversão, fazendo com que todas as regiões detectadas sejam preenchidas na cor branca, conforme é apresentada na Figura 14.

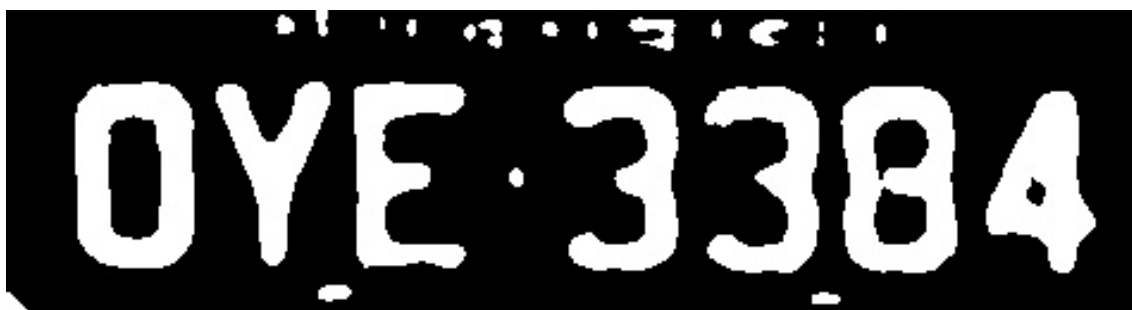


Figura 14 – Região recortada e inversamente binarizada.

Após este processo temos a imagem tratada e preparada para ser submetida ao processo de detecção de caracteres através da biblioteca *tesseract-ocr*, biblioteca essa com necessidade de instalação no ambiente *python* de desenvolvimento e no sistema operacional como aplicativo.

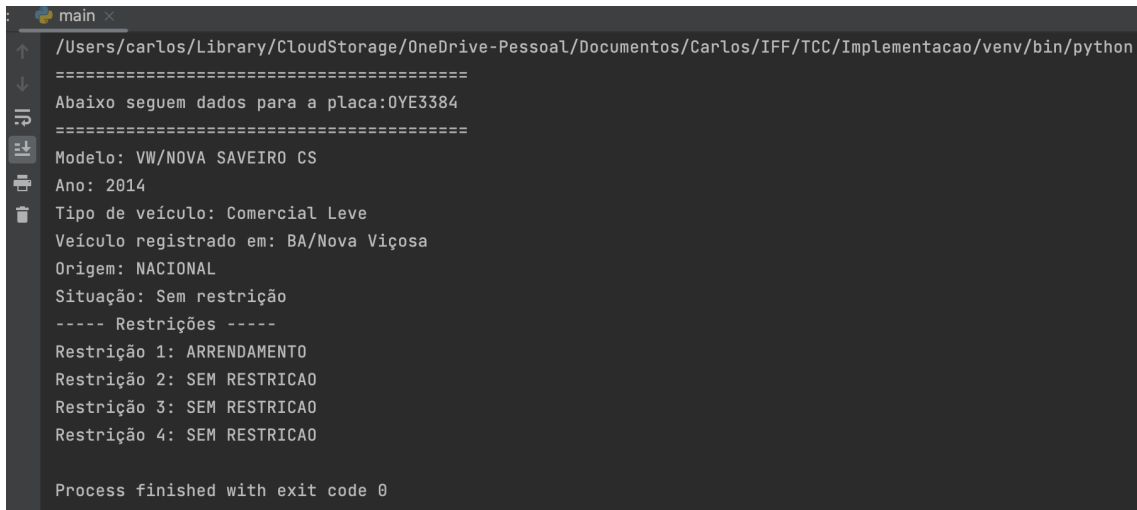
O processo de detecção de caracteres é de forma mais simplificada, pois a biblioteca responsável por esta finalidade já vem acompanhada com o modelo treinado para diversos idiomas, sendo necessário apenas configurar a biblioteca para que de ênfase na língua inglesa utilizando apenas caracteres alfanuméricos desconsiderando de sua análise os caracteres especiais, pontuações e acentos. Outra configuração atribuída neste trabalho é a união de todos os os caracteres em uma única *string*, evitando que o processo retorne uma sequência em *array*. Para validação do texto encontrado o mesmo é submetido a um conferência na forma de expressão regular, sendo que o *layout* das placas veiculares brasileiras seguem apenas dois padrões possíveis a conferência é realizada em relação a duas expressões regulares. Caso o texto encontrado não passe pela validação o sistema retorna uma placa genérica "ABC1234" dando continuidade nas verificações cadastrais.

Com o texto validado é realizada a busca pelo texto no sistema mantido por terceiro com acesso ao sistema do Ministério da Infraestrutura, responsável pelo cadastro e atualização cadastral de toda a frota de veículos no território brasileiro. A integração ao sistema *ApiPlacas* é realizada através de uma requisição simples, sendo apenas necessário um *token* de autenticação, enviado junto da *URL* da requisição.

```
main.py × plateSituation.py × plateDetect.py × plateRecognition.py ×
1 import requests
2 import json
3 import os
4 from config.defs import ROOT_DIR
5 from creds import tkn
6
7
8 def CheckPlateSituation(plate):
9     # call API to check license plate situation
10    status_code = 0
11    if not os.path.isfile(os.path.join(ROOT_DIR, 'datas', plate + '.json')):
12        status_code = getplatedata(plate)
13        if status_code == 200:
14            return platedata(plate)
15    else:
16        return platedata(plate)
17    return status_code
18
19 def getplatedata(plate: str):
20    plate = plate
21    api_url = f"https://wdapi.com.br/placas/{plate}/{tkn}"
22    response = requests.get(api_url)
23    status_code = response.status_code
24    if status_code == 200:
25        json_data = response.json()
26        json_pth = os.path.join(ROOT_DIR, 'datas', plate + '.json')
27        with open(json_pth, 'w') as file_writer:
28            json.dump(json_data, file_writer)
29    return status_code
```

Figura 15 – Integração ao sistema *ApiPlacas*

O retorno é dado em formato *JSON*, formato amplamente utilizado em aplicações *web*. Estes dados são armazenados localmente, em arquivo no formato *JSON* podendo ser acessados posteriormente na ocorrência de uma mesma placa de identificação ser detectada novamente, reduzindo dessa forma a quantidade de requisições ao sistema *ApiPlacas*. Na Figura 16 é demonstrada uma fração dos dados retornados pela requisição, sendo apresentados ao usuário apenas os dados de interesse para este trabalho.



```
main x
/Users/carlos/Library/CloudStorage/OneDrive-Pessoal/Documentos/Carlos/IFF/TCC/Implementacao/venv/bin/python
=====
Abaixo seguem dados para a placa:0YE3384
=====
Modelo: VW/NOVA SAVEIRO CS
Ano: 2014
Tipo de veículo: Comercial Leve
Veículo registrado em: BA/Nova Viçosa
Origem: NACIONAL
Situação: Sem restrição
---- Restrições ----
Restrição 1: ARRENDAMENTO
Restrição 2: SEM RESTRICAO
Restrição 3: SEM RESTRICAO
Restrição 4: SEM RESTRICAO

Process finished with exit code 0
```

Figura 16 – Retorno obtido pelo sistema

5. CONCLUSÕES E TRABALHOS FUTUROS

O presente trabalho teve como objetivo a demonstração das inúmeras possibilidades para o vasto campo da tecnologia de visão computacional e aprendizagem de máquina no tocante a detecção de objetos e reconhecimento de caracteres.

5.1 CONCLUSÕES

Ao findar do trabalho, pode-se chegar a um sistema funcional com resultados satisfatórios, considerando-se que este trabalho teve tempo limitado para treinamento, obtenção de imagens e dados que auxiliariam na obtenção de detecções ainda mais precisas, tendo sido possível o treinamento de uma rede neural com imagens em diversas situações ambientais, obtendo-se o reconhecimento de caracteres com boa acurácia.

A integração realizada para obtenção da situação cadastral do veículo, foi possível sem percalços, retornando de forma satisfatória os dados do veículo.

Por fim, o trabalho obteve êxito em sua proposta, sendo capaz de detectar veículos e sua placa de identificação além da posterior verificação da situação cadastral do mesmo.

5.2 TRABALHOS FUTUROS

Na presente seção o autor apresenta propostas de trabalhos futuros com base no trabalho realizado, sendo estes voltados a detecção de veículos, reconhecimento de caracteres e segurança.

O adendo de uma simples função ao sistema já traria novos resultados ao sistema e como primeira proposta estaria a implantação de banco de dados para armazenagem mais estruturada dos dados. Para desenvolvimentos com maior grau de dificuldade e pesquisa, sugere-se a detecção de veículos e sua placa de identificação em vídeos, sendo possível a verificação destes quesitos em tempo real. Sugere-se também o desenvolvimento de sistema com registro de data, horário, frequência e velocidade de cada detecção, sendo submetido posteriormente a um modelo de padrões ambientais, verificando a existência de eventos anômalos ao padrão ambiental modelado, contando com um módulo para envio de mensagem de texto e imagem ao administrador do sistema e/ou interessados caso detectado veículo com situação cadastral irregular ou considerado com evento anômalo.

REFERÊNCIAS

- ABESE. *Revista Segurança Inteligente* 18. 2021. Pag.08-11. Disponível em: <https://issuu.com/abeseoficial/docs/revista_seguranca_inteligente_n18pgs>.
- BENEZETH, Y. et al. Comparative study of background subtraction algorithms. *Journal of Electronic Imaging*, v. 19, p. 033003–033003, 07 2010.
- BOESCH, G. *LabelImg for Image Annotation*. 2022. Disponível em: <<https://viso.ai/computer-vision/labelimg-for-image-annotation/>>.
- BRADSKI, G.; KAEHLER, A. *Learning OpenCV: Computer Vision with the OpenCV Library*. [S.l.]: "O'Reilly Media, Inc.", 2008. Google-Books-ID: seAgiOfu2EIC. ISBN 978-0-596-55404-0.
- BRANDIZZI, L. E. N. *Visão computacional: O que é? Como funciona?* 2020. Disponível em: <<https://www.serpro.gov.br/menu/noticias/noticias-2020/o-que-eh-visao-computacional>>.
- BRASIL. *STJ - Notícias: Integração entre as instâncias, o caminho para a Justiça brasileira*. 2018. Disponível em: <https://www.stj.jus.br/sites/portalp/Paginas/Comunicacao/Noticias-antigas/2018/2018-10-08_19-58_Integracao-entre-as-instancias-o-caminho-para-a-Justica-brasileira.aspx>.
- DU, S. et al. Automatic license plate recognition (alpr): A state-of-the-art review. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 23, n. 2, p. 311–325, 2013.
- GOODFELLOW, I. et al. *Deep learning*. [S.l.]: MIT press Cambridge, 2016. v. 1.
- ISMAIL, S.; ABDULLAH, S. S.; FAUZI, F. Statistical binarization techniques for document image analysis. *Journal of Computer Science*, v. 14, p. 23–36, 01 2018.
- Laroca, R. et al. On the cross-dataset generalization in license plate recognition. In: *International Conference on Computer Vision Theory and Applications (VISAPP)*. [S.l.: s.n.], 2022. p. 166–178. ISBN 978-989-758-555-5. ISSN 2184-4321.
- LAROCA, R. et al. A robust real-time automatic license plate recognition based on the yolo detector. In: *2018 International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2018. p. 1–10.
- LAROCA, R. et al. An efficient and layout-independent automatic license plate recognition system based on the yolo detector. 02 2021.
- LEVEZ, F. B.; BENINI, F. A. V.; GOMES, G. H. F. Aplicando opencv no raspberry pi. In: *3º Workshop de Inovação, Pesquisa, Ensino e Extensão*. [S.l.: s.n.], 2018.
- LIBXML2. 2022. Disponível em: <<https://gitlab.gnome.org/GNOME/libxml2/-/wikis/home>>.
- NACIONAL, I. *RESOLUÇÃO Nº 372, DE 18 DE MARÇO DE 2011 - DOU*. 2011. Disponível em: <<https://www.in.gov.br/web/dou>>.
- NACIONAL, I. *RESOLUÇÃO Nº 780, DE 26 DE JUNHO DE 2019 - DOU - Imprensa Nacional*. 2019. Disponível em: <<https://www.in.gov.br/web/dou>>.

PRIADANA, A.; HARJOKO, A. Deteksi perubahan citra pada video menggunakan illumination invariant change detection. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, v. 11, n. 1, p. 89–98, 2017.

PSF, P. S. F. *Python - History and License*. 2022. Disponível em: <<https://docs.python.org/3/license.html>>.

PyCharm: o IDE Python da JetBrains para desenvolvedores profissionais. Disponível em: <<https://www.jetbrains.com/pt-br/pycharm/>>.

RAMADHAN, D.; SARI, I. P.; SARI, L. Comparison of background subtraction, sobel, adaptive motion detection, frame differences, and accumulative differences images on motion detection. *SINERGI*, v. 22, p. 51, 02 2018.

REDMON, J. *Darknet: Open Source Neural Networks in C*. 2013–2016. <<http://pjreddie.com/darknet/>>.

REDMON, J. et al. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. *arXiv*, 2018.

SETHIAN, J. A. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, National Academy of Sciences, v. 93, n. 4, p. 1591–1595, 1996. ISSN 0027-8424.

SHASHIRANGANA, J. et al. Automated license plate recognition: A survey on methods and techniques. *IEEE Access*, v. 9, p. 11203–11225, 2020.

SMITH, R. History of the tesseract ocr engine: what worked and what didn't. *Proceedings of SPIE - The International Society for Optical Engineering*, p. 02–, 02 2013.

SMITH, R. tesseract-ocr, 2021. Disponível em: <<https://github.com/tesseract-ocr/tesseract>>.

SRINATH, K. Python—the fastest growing programming language. *International Research Journal of Engineering and Technology*, v. 4, n. 12, p. 354–357, 2017.

SSP. *Indicadores Criminais - Secretaria da Segurança Pública*. 2021. Disponível em: <<https://www.ssp.rs.gov.br/indicadores-criminais>>. Acesso em: 20 mai. 2021.

STACK OVERFLOW. *Stack Overflow Trends*. 2021. Disponível em: <<https://insights.stackoverflow.com/trends>>.

TAVARES, J. M. R. *Processamento e análise de imagem em biomecânica*. 2008.

TIOBE. 2022. Disponível em: <<https://www.tiobe.com/tiobe-index/>>.

VASCONCELLOS, H.; MENDES, L. *Um em cada quatro municípios do RS tem câmeras para combater a criminalidade | GZH*. 2018. Disponível em: <<https://gauchazh.clicrbs.com.br/seguranca/noticia/2018/06/um-em-cada-quatro-municipios-do-rs-tem-cameras-para-combater-a-criminalidade-cji6bclqa0bp601panx7u.html>>.

VO, Q. N. et al. Binarization of degraded document images based on hierarchical deep supervised network. *Pattern Recognition*, Elsevier, v. 74, p. 568–586, 2018.

YANQUE, N. Y. A. *Um estudo comparativo de métodos de segmentação de documentos antigos*. Tese (Doutorado) — Universidade de São Paulo, Nov 2018. Disponível em: <<http://www.teses.usp.br/teses/disponiveis/45/45134/tde-25092019-140704/>>.