UNESPAR
Universidade Estadual do Paraná
Campus de Apucarana

MATHEUS C. SOUZA

RECONHECIMENTO DE PLACAS DE VEÍCULOS POR MEIO DE IMAGENS

APUCARANA
2021

MATHEUS C. SOUZA

RECONHECIMENTO DE PLACAS DE VEÍCULOS POR MEIO DE IMAGENS

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual do Paraná para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Ms. Fábio T. Matsunaga

APUCARANA
2021

Dados Internacionais de Catalogação na Publicação (CIP)

S729r Souza, Matheus C.
Reconhecimento de placas de veículos por meio de imagens. / Matheus C. de Souza. – Apucarana, 2021.
46 f. : il. ; 30 cm.

Orientador: Prof. Me. Fábio T. Matsunaga

Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) Universidade Estadual do Paraná – UNESPAR Campus Apucarana, 2021.

1. Reconhecimento Óptico de Caracteres. 2. Reconhecimento de placa de veículo. 3. *YOLO* v4. I. Matsunaga, Fábio T. II. Universidade Estadual do Paraná. III. Título.

CDU 004.8

MATHEUS C. SOUZA

RECONHECIMENTO DE PLACAS DE VEÍCULOS POR MEIO DE IMAGENS

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Ciência da Computação da Universidade Estadual do Paraná para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Prof. Ms. Fábio T. Matsunaga
Universidade Estadual do Paraná
Orientador

Prof. Ms. Edison Antonio Saad Filho
Universidade Estadual do Paraná

Prof. Ms. Jader Maikol Caldonazzo Garbelini
Universidade Estadual do Paraná

Apucarana, 03 de dezembro de 2021

Este trabalho é dedicado aos meus pais, Sonia e Valdevino, por me proporcionarem a oportunidade de realizar o curso e este trabalho.
Também dedico as pessoas que usam a Computação como ferramenta de trabalho.

AGRADECIMENTOS

A todos que estiveram comigo e contribuíram comigo direta ou indiretamente para a realização deste trabalho, muito obrigado.

Os agradecimentos principais são direcionados ao meu orientado, Prof. Ms. Fábio T. Matsunaga, que muitas vezes mesmo sem saber não me deixou desistir nos momentos que eu cogitava a possibilidade, a sua participação foi essencial para o desenvolvimento deste trabalho, tendo contribuído muito para meu crescimento.

Agradeço a todos que fazem parte da comunidade acadêmica, meus professores e todos colegas e ex-colegas da Unespar, pelas experiências que tivemos juntos e por me apoiarem.

Agradeço também aos meus pais e irmãos, por todo o suporte que foi me dado durante todos esses anos, agradeço as minhas sobrinhas Mariana e Isadora por me trazerem tanta felicidade em momentos de preocupação com este trabalho.

Por último, mas não menos importante, agradeço aos meus amigos Luís Fernando Sivirino Gaspar e Igor Benato da Silva por sempre me prestarem suporte quando necessário, principalmente por me ajudarem a realizar o treinamento do detector de placas em seus computadores, o que se tornava inviável treinar no meu computador foi um processo mais simples para eles.

“Podemos ver apenas uma curta distância à frente, mas podemos ver que muito precisa ser feito.
- Alan Turing (TURING, 1950)“

SOUZA, MATHEUS C.. **Reconhecimento de placas de veículos por meio de imagens**. 46 p. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual do Paraná, Apucarana, 2021.

RESUMO

Quando um veículo é furtado a taxa de sucesso para recuperá-lo pode ser menor que 50% dependendo do estado onde o crime aconteceu (Governo do Estado do Paraná, 2019). Um dos motivos para que essa taxa seja tão baixa é dado pela falta de ferramentas que auxiliem as forças de segurança com a busca pelo veículo. Para este fim, o presente trabalho utiliza o *YOLO* na versão 4, para realizar a detecção da placa, junto com um dataset treinado exclusivamente para isso, após a detecção é utilizado o Pytesseract para realizar o *Optical Character Recognition (OCR)*, em português Reconhecimento Óptico de Caracteres (ROC). A partir de imagens capturadas por câmeras em ruas e rodovias deve-se realizar o reconhecimento do código da placa e posteriormente realizar consulta da mesma para verificar a situação legal do carro. Com base na metodologia utilizada, o trabalho obteve resultado promissor na detecção das placas, porém o *OCR* não obteve um resultado tão promissor quanto a detecção.

Palavras-chave: Reconhecimento Óptico de Caracteres. Reconhecimento de placa de veículo. *YOLO* v4.

SOUZA, MATHEUS C.. **Vehicle license plate recognition through images**. 46 p. Final Project (Bachelor of Science in Computer Science) – State University of Paraná, Apucarana, 2021.

ABSTRACT

When a vehicle is stolen, the success rate to recover it can be less than 50% depending on the state where the crime happened (Governo do Estado do Paraná, 2019). One of the reasons why this rate is so low can be attributed to the lack of tools that assist security forces with the search for the vehicle. To make this possible, the present work uses YOLO in version 4, to perform plaque detection, along with a dataset trained exclusively for therefore, after detection, Pytesseract is used to perform the Optical Character Recognition (OCR), in Portuguese Optical Character Recognition (ROC). From images captured used by cameras on streets and highways, recognition of the license plate code and subsequently carry out a consultation to verify the legal status of the car. Based on methodology used, the work had a promising result in the detection of plaques, but the OCR did not yield as promising a result as detection.

Keywords: Optical Character Recognition. License Plate Recognition. *YOLO* v4.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
B.O	Boletim de Ocorrência
CNN	*Convolutional Neural Network*
GPU	*Graphics Processing Units*
IBGE	Instituto Nacional de Geografia e Estatística
KNN	*K-Nearest Neighbor*
OCR	*Optical Character Recognition*
RNA	Redes Neurais Artificiais
ROI	*Region of Interest*
SINESP	Sistema Nacional de Informações de Segurança Pública, Prisionais, de Rastreabilidade de Armas e Munições, de Material Genético, de Digitais e de Drogas
URL	*Uniform Resource Locator*
YOLO	*You Only Look Once*

SUMÁRIO

1 INTRODUÇÃO

No primeiro semestre de 2019, o estado do Paraná foi o segundo estado com menor taxa de furtos e roubos de veículos, sendo contabilizado o número de ocorrências para cada 100 mil habitantes. O Paraná contabilizou 40 veículos roubados por 100 mil habitantes, totalizando, 3.061 casos. Já para furtos, o Paraná teve 102 casos para cada 100 mil habitantes, no total, 7.832 casos, totalizando furtos e roubos, a taxa foi de 142 registros por 100 mil habitantes, em números absolutos, foram 10.893 registros (Governo do Estado do Paraná, 2019).

Dentre esses casos, o Paraná obteve um taxa de recuperação de 61,6%, ou seja, dos 10.893 furtos ou roubos, 6.713 veículos foram recuperados (Governo do Estado do Paraná, 2019). Com uma ferramenta de analise automatizada, poderiam ser encontrados mais veículos que foram furtados ou roubados.

Quando um veículo é furtado é necessário que seja aberto um Boletim de Ocorrência (B.O), um processo que pode ser feito de forma presencial ou online, para que as forças de segurança possam identificar o veículo. Após o registro do B.O, os dados referentes ao carro são atualizados na base de dados fazendo que conste uma situação de furto ou roubo. Com essa informação já é possível realizar uma consulta para verificar a situação do veículo. Atualmente o Sistema Nacional de Informações de Segurança Pública, Prisionais, de Rastreabilidade de Armas e Munições, de Material Genético, de Digitais e de Drogas (SINESP) disponibiliza um aplicativo para celulares, chamado Sinesp Cidadão, onde é possível inserir os dados da placa manualmente e consultar a situação do veículo.

No entanto, existe uma grande dificuldade das forças de segurança a identificar as placas de veículos furtados, fazendo com que a taxa de sucesso de recuperação seja menor que 50% dependendo do estado onde o crime aconteceu (Governo do Estado do Paraná, 2019). Um dos motivos para que essa taxa seja tão baixa é dado pela falta de ferramentas computacionais que identifiquem as placas por meio de imagens.

Um dos métodos computacionais que auxiliam no reconhecimento de texto por meio de imagem é o OCR (*Optical Character Recognition* - Reconhecimento Óptico de Caracteres). O *(OCR)*, só se tornou termo de registro em pesquisas em 1950, quando David Shepard e Louis Tordella pesquisavam formas de automatizar dados da Agência de Segurança Nacional Americana (NSA). Juntamente com Harvey Cook, eles produziram o Gismo, o primeiro software para *OCR* (Cadernos de Tipografia, 2007).

Diante da situação, o presente trabalho visa utilizar técnicas de reconhecimento de imagem combinadas com *Optical Character Recognition (OCR)*, em português Reconhecimento Óptico de Caracteres (ROC). A partir de imagens capturadas por câmeras em ruas e rodovias deve-se realizar o reconhecimento do código da placa e posteriormente realizar a consulta, com

o objetivo de verificar a situação legal do carro.

2 FUNDAMENTAÇÃO TEÓRICA

O objetivo desse capítulo é apresentar o referencial teórico e descrever os conceitos e algoritmos necessários para realizar o reconhecimento de placas de veículos através de imagens.

2.1 Trabalhos Correlatos

Na tese de Borges (BORGES, 2018), ele utilizou segmentação da placa veicular, leitura dos caracteres presentes e também a segmentação da área da cor com o intuito de reconhecer os caracteres presentes na placa do veículo para realizar a autenticação veicular de duas etapas utilizando a placa e a cor. Com esse método, seu projeto obteve 70% de precisão no teste, onde os efeitos de sombreamento e forte luminosidade sobre a lataria do veículo tiveram forte influência sobre os resultados negativos. Para realizar o trabalho foi utilizado o algoritmo de classificação por similaridade chamado K-Nearest Neighbor (KNN), que é um método de aprendizado supervisionado.

Utilizando processamento de imagens juntamente com técnicas de redes neurais, Clavero conseguiu analisar caracteres em imagens (CLAVERO, 2019). Para isso utilizou Redes Neurais Artificiais (RNA), mais especificamente o algoritmo de Perceptron de multicamadas. Em seu trabalho é possível realizar o reconhecimento de letras e textos, sendo eles escritos ou não a mão. Foram utilizadas três redes neurais no trabalho, sendo que a rede A obteve um total de 165 reconhecimento para 496 amostras.

O resultado foi que em 33% de acerto, a rede B conseguiu identificar 114 de 248 amostras, com 45% de precisão e por último, a rede C obteve um acerto de 134 de 248 amostras, totalizando 54% de acerto. A base B recebeu somente letras escritas pela mão esquerda, enquanto a base C recebeu letras escritas pela mão direita, já a base A recebeu as letras escritas por ambas as mãos, isso foi feito para ter uma maior variedade de amostras e letras com padrões diferentes.

Na dissertação de Ferreira (FERREIRA, 2018), em seu trabalho para reconhecer e diferenciar sinalizações de trânsito brasileiras através de imagens, utilizou os algoritmos Viola-Jones e AKAZE. Para isso, foram realizados experimentos em três resoluções diferentes, que são 1920 X 1080, 1080 X 720 e 854 X 480 *pixels* sendo que em todas as resoluções testadas ocorreram detecções e diferenciações. Sendo que a melhor taxa de detecção, foi com a menor resolução, que obteve 99,48%. E na diferenciação, a maior resolução foi quem obteve a melhor taxa de acerto, com 96,63%.

2.2 Imagem Digital

Foi descoberto em 1966, por Sir Isaac Newton, que quando um feixe de luz solar passa através de um prisma de vidro, o feixe de luz que emerge não é branco, mas consiste em um espectro, que varia de uma extremidade Violeta para a outra extremidade Vermelha. Porém a variação de cor que é visível para os olhos humanos é uma parte muito pequena do real variação de cores existentes, onde em uma das extremidades do espectro se encontra as ondas de rádio, com um comprimento de onda bilhões de vezes maiores que os da luz visível. Na outra extremidade, se encontram os raios gama, com comprimento de onda bilhões de vezes menores que o da luz visível. Este espectro pode ser visualizado na Figura 1 (GONZALEZ; WOODS, 2010).

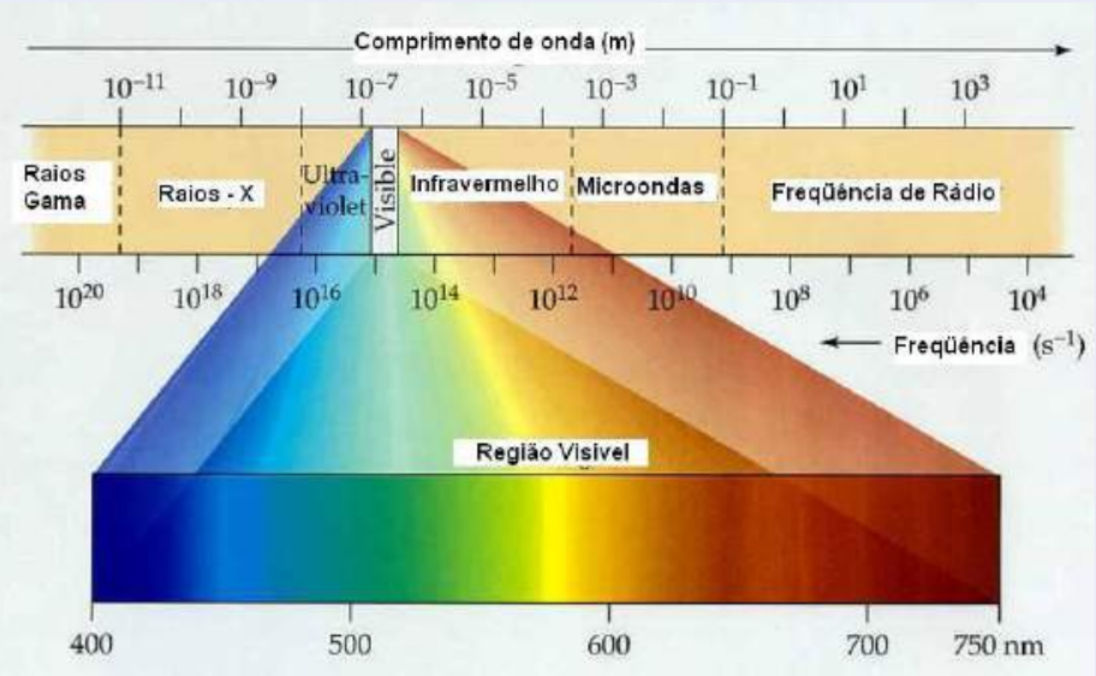

Figura 1 – Comprimento de ondas (GONZALEZ; WOODS, 2010)

Uma imagem digital pode ser definida por uma matriz de pontos, onde cada ponto dessa matriz é chamado de *pixel*. Cada um desses *pixels* carrega informações de cor, coordenadas espaciais, quando agrupados, podemos obter outras informações como coordenadas no espaço de escala, área de adjacência, orientação, ângulo. Após obter essas informações, pode se analisar e processar elas afim de realizar processos como classificação, realçamento, reconhecimento de padrões e a detecção de objetos (GONZALEZ; WOODS, 2010).

Como uma imagem pode ser representada por uma matriz de pontos, ela pode ser manipulada através de operações matemáticas, cada ponto pode ser definido pela sua coordenada de (X,Y) no plano cartesiano. Podemos observar essas coordenadas de um ponto e seus vizinhos através da Figura 2.

X-1,Y+1	X,Y+1	X+1,Y+1
X-1,Y	X,Y	X+1,Y
X-1,Y-1	X,Y-1	X+1,Y-1

Figura 2 – Coordenadas no plano cartesiano de um ponto e seus vizinhos

Cada *pixel* da matriz de pontos carrega sua informação de cor, sendo que na prática a maioria das imagens é exibida utilizando 8 bits, no caso das imagens coloridas de 24 bits, são utilizados 3 canais separados de 8 bits. Dessa forma, é esperado que os valores desses canais estejam no intervalo de 0 e 255 (GONZALEZ; WOODS, 2010).

As cores são formadas a partir desse conjunto de valores, cada conjunto representa uma das cores primárias, que são o Azul, Verde e Vermelho. Quando se obtém o valor máximo, ou seja, 255 nos 3 conjuntos se obtém a cor Branca, já por outro lado, quando os 3 conjuntos estão no outro extremo é obtido a cor Preta. Portanto, o que determina a cor é a combinação de valores dos conjuntos, parte das combinações podem ser vistas na Figura 3.

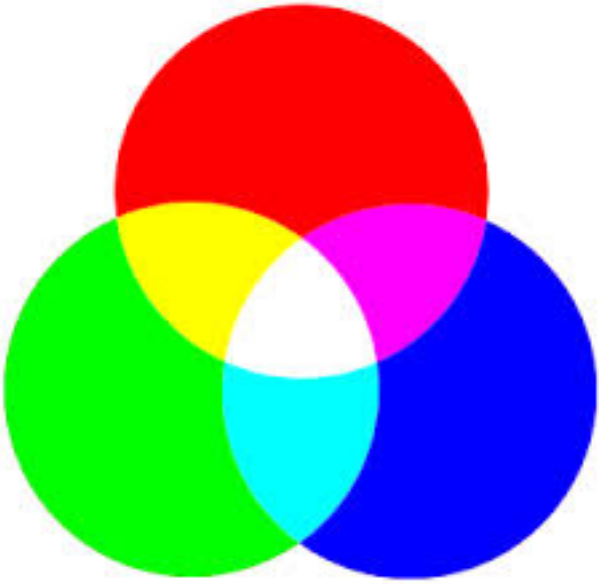

Figura 3 – Combinação de cores

2.2.1 Filtragem

Na maior parte dos casos as imagens estão no padrão *RGB(Red, Green, Blue)*, que representa uma imagem colorida, porém, nesses casos temos um baixo nível de contraste o que dificulta as operações sobre a imagem. Para resolver esse problema de contraste, podem ser aplicadas transformações na imagem.

As transformações da imagem pixel a pixel são chamadas de técnicas de filtragem, elas não dependem somente do nível de cinza do *pixel* como também dependem do nível de cinza

de seus *pixels* vizinhos (Instituto Nacional de Pesquisas Espaciais, s.d). Dentre as técnicas de filtragem, existem as chamadas técnicas de realce de imagens, as quais são as responsáveis por melhorar a aparência de determinadas características da imagem, fazendo com que a análise sobre a imagem fique mais simples.

Não é necessário usar o realce em todas as imagens, somente em alguns casos, como quando a imagem sofre com degradação ou perda de qualidade e causada por ruído, perda de contraste, distorção na aquisição, problemas com iluminação ou borramento. O objetivo de usar transformações de contraste na imagem é obter uma melhor qualidade sob critérios subjetivos ao sistema visual humano, deixando a percepção de informações contidas nas imagens mais fácil.

2.2.1.1 Escala de Cinza

Uma das técnicas mais utilizadas com o objetivo de melhorar o contraste é a escala de cinza, onde o *pixel* é analisado junto com seu conjunto de vizinhos. Nessa análise é calculado o intervalo de contraste, que é a diferença entre os valores de intensidade máximo e mínimo que o *pixel* pode assumir, geralmente o intervalo pode ser representado com o nível mínimo de cinza em 0 e o máximo em 255 (PEDRINI, 2021). Pode-se ver o resultado da aplicação desse filtro através da Figura 4.

Figura 4 – Área de interesse antes e depois de aplicar o filtro de escala de cinza.

Para uma melhor visualização da quantidade de *pixels* de cada tonalidade na imagem, pode ser gerado um histograma dela. Na imagem abaixo, podemos analisar o histograma da Figura 5, sendo possível identificar que a maioria dos *pixels* (Eixo Y) estão na intensidade entre 30 e 70 (Eixo X), o que representa a intensidade do vermelho na escala de cinza.

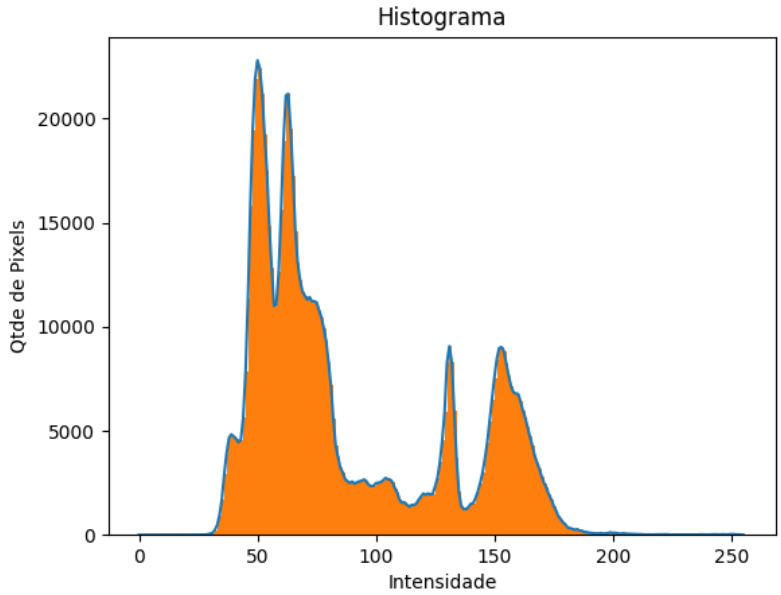

Figura 5 – Histograma da escala de cinza

2.2.1.2 Binarização

Na escala de cinza é possível atingir 256 tons de cinza, já no caso da binarização o processo limita ainda mais o intervalo de cores que é possível assumir, sendo somente possível duas cores, o branco e preto. Após a aplicação dos filtros, o contraste da imagem se torna mais forte, realçando as características e facilitando as operações sobre a imagem. A binarização pode ser visualizada na Figura 6.

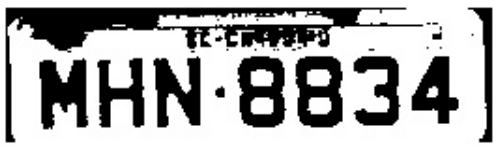

Figura 6 – Área de interesse após aplicar o filtro de binarização.

2.3 Segmentação

A segmentação é o processo responsável por subdividir uma imagem em regiões ou objetos que a compõem. Onde o nível de detalhe dessa subdivisão é realizada depende do problema a ser resolvido. Basicamente, a segmentação só deve parar quando o objeto ou *region of interest (ROI)*, em potuguês, região de interesse, for detectada (GONZALEZ; WOODS, 2010).

A descontinuidade e similaridade são propriedades básicas de valores de intensidade, onde na primeira a imagem é dividida de acordo com mudanças bruscas de intensidade, como as bordas. Já na segunda, a imagem é dividida em regiões que são semelhantes a um conjunto de parâmetro predefinidos. Existem outras abordagens, a que será abordada no trabalho é uma delas,

onde a imagem não é dividida em regiões e sim é feita apenas uma varredura para encontrar todos os objetos da imagem, ganhando um bom desempenho computacional.

2.3.1 *YOLO* v4

O *YOLO* é um projeto totalmente *opensource* que hoje se encontra na versão 4 e no qual testes já estão sendo realizados com a versão 5, sendo considerado o estado da arte para o reconhecimento de objetos em imagens, a detecção e localização da região do objeto é realizada pelo *YOLO* somente em uma varredura da imagem, por isso seu nome são as iniciais de *You Only Look Once*, em português, você olha apenas uma vez.

Essa abordagem de varrer a imagem apenas uma vez é sua grande vantagem quando comparada a outros métodos de detecção como o *Haar Cascade* e o *HOG*, onde eles usam a técnica de *Sliding Window*, que faz com que a imagem seja dividida em várias regiões e cada área era submetida para um classificador, o que poderia acontecer milhares de vezes tendo um custo computacional consideravelmente alto (ALESSANDRO FARIA, 2021).

Sua técnica lançada em 2015 por Joseph Redmon e Ali Farhadi foi muito inovadora pelo fato de processar com precisão similar ou superior comparados aos seus concorrentes, mas com uma performance de tempo muito performática em tempo real (em torno de 30 fps – frames per second). Muitas pesquisas relataram desempenho de tempo 10 vezes comparados aos métodos mais precisos na época de seu lançamento. Inclusive em 2017 foram realizados testes com super computadores da época envolvendo IBM e NVIDIA (Lucas Agrela, 2017).

O ganho de performance, velocidade de inferência e assertividade são consideradas as principais vantagens comparadas com as outras versões, sendo que na versão 4 também foram aplicadas técnicas mais eficazes para rodar o processamento em GPUs, baseado na otimização e utilização com um menor uso da memória. Podemos comparar o seu desempenho com os métodos *EfficientDet, RetinaNet/MaskRCNN* onde todos fazem uso do COCO dataset.

O COCO dataset é um conjunto pré-treinado capaz de detectar 80 objetos diferentes, dentre esses objetos é possível detectar pessoas, carros, motos, ônibus, bicicleta, avião, trem, entre outros, mas não é possível detectar placas de veículos (Amikelive, 2018). Através da Figura 7 é possível notar a grande diferença na quantidade de quadros por segundo que é possível obter com o *YOLO* quando comparado com os outros métodos (Alexey Bochkovskiy, 2021).

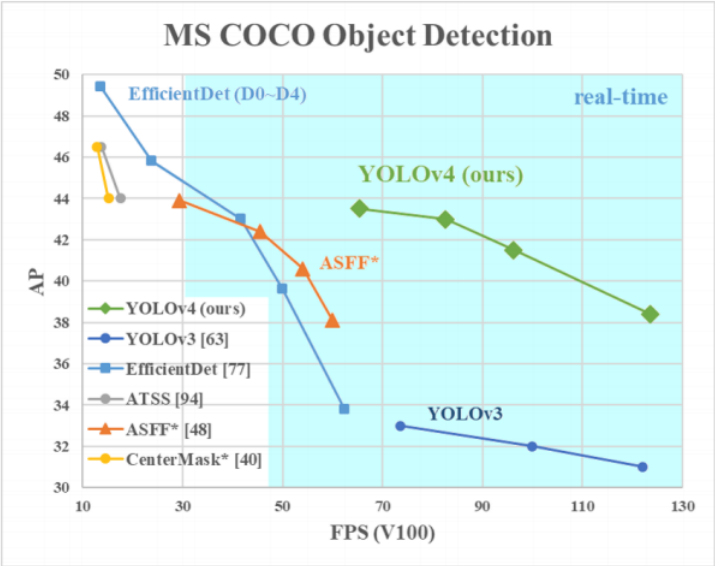

Figura 7 – Comparação de desempenho (Alexey Bochkovskiy, 2021)

Além de detectar o objeto com o *YOLO*, também é possível demarcar a posição de cada objeto na imagem, classificá-los de acordo com suas respectivas classes e mostrar a taxa de certeza sobre aquele objeto, essa taxa são valores entre 0 e 1, sendo 1 o maior nota possível. Podemos verificar através da Figura 8, que na imagem foi encontrado somente uma placa com uma nota de 0,99 o que é uma precisão muito boa levando em consideração que o valor máximo é 1. Para obter esses resultados o *YOLO* faz uso de uma rede neural convolucional, mas conhecida como *Convolutional Neural Network (CNN)*.

Figura 8 – Detecção da placa em uma imagem

2.4 Redes Neurais Artificiais

O tema de Redes Neurais Artificiais (RNA) só começou a ser pesquisado de forma mais efetiva no começo dos anos 1990, mesmo que os primeiros trabalhos sobre RNA foram publicados em meados dos anos 1960. O sistema nervoso dos seres vivos foi a inspiração para as RNA que são um modelo computacional. Elas tem a capacidade de adquirir e realizar a

manutenção do conhecimento, elas também podem ser definidas como um conjunto de unidades de processamento, que são caracterizadas por neurônios artificiais, esses são interligados por um grande número de interconexões, chamadas de sinapses artificiais), sendo representadas por vetores ou matrizes de pesos sinápticos (SILVA et al., 2016).

2.4.1 Características principais

Quando pensamos em RNA devemos saber que as características mais relevantes estão listadas abaixo.

- Adaptação por experiência
  - A partir de uma apresentação sucessiva de exemplos relacionados ao comportamento do processo, os parâmetros internos da rede sofrem adaptações, esses parâmetros, normalmente são os pesos sinápticos. Esses exemplos usados na apresentação sucessiva podem ser padrões, amostras ou medidas.
- Capacidade de aprendizado
  - Por meio da aplicação de método de treinamento, dentre as diversas variáveis que compõem a aplicação a RNA consegue extrair o relacionamento existente.
- Habilidade de generalização
  - Após o treinamento da RNA, ela é capaz de generalizar o conhecimento adquirido, tendo a possibilidade de propor soluções que eram desconhecidas até então.
- Organização de dados
  - Com base em características essenciais envolvendo um determinado conjunto de informações a respeito do processo, a RNA é capaz de realizar a organização interna para possibilitar o agrupamento de padrões que apresentam particularidades em comum.
- Tolerância a falhas
  - Levando em consideração ao alto nível de interconexões entre os neurônios artificiais, a RNA se torna um sistema tolerante a falhas quando parte da sua estrutura interna é sensivelmente corrompida.
- Armazenamento distribuído
  - Através de uma forma distribuída entre as diversas sinapses de seus nerônios artificiais, é possível representar o conhecimento a respeito do comportamento de

determinado processo dentre de uma arquitetura neural. Isso permite então um incremento da robustez da arquitetura frente a eventuais neurônios que se tornaram importantes.

- Facilidade de prototipagem

  - Dependendo da especificidade da aplicação, a implementação das arquiteturas neurais, pode ser tranquilamente prototipada em hardware ou em software, pois, após o processo de treinamento, os resultados são obtidos por algumas operações matemáticas elementares.

2.4.2 Rede Neural Convolucional

A Rede Neural Convolucional, também conhecida como *CNN* que é a sigla do seu nome em inglês *Convolutional Neural Network*, é uma variação das *Multilayer Perceptrons (MLP)*, em português, Perceptrons de Múltiplas Camadas que foi inspirada no processamento de dados visuais em processos biológicos. Uma *CNN* é capaz de aplicas filtros em dados visuais, mantendo a relação de vizinhança entre os *pixels* da imagem ao longo do processamento da rede.

A CNN consiste em múltiplas partes com funções diferentes. De inicio é comum aplicar sobre o dado de entrada camadas de convolução, essa camada de convolução é composta por diversos neurônios, onde cada um tem a responsabilidade de aplicar um filtro em um pedaço específico da imagem. Podemos ver isso como cada neurônio sendo conectado a um conjunto de *pixels* da camada anterior e em cada uma dessas conexões se atribui um peso.

A saída passada para a camada seguinte é produzida a partir de uma combinação das entradas de um neurônio, utilizando os pesos respectivos de cada uma de suas conexões. Estes pesos que foram atribuídos as conexões de um neurônio podem ser interpretados como uma matriz que representa o filtro de uma convolução de imagens no domínio espacial, que também é conhecido como *kernel* ou máscara (VARGAS et al., 2016).

2.5 Etapas de um sistema de processamento de imagens

Para alcançar o objetivo, é necessário estabelecer alguns métodos e processos, podemos descrever as etapas a serem seguidas utilizando a Figura 9 descrita por Filho e Neto (FILHO; NETO, 1999). O primeiro passo é fazer a aquisição da imagem, com a imagem já capturada ela é repassada para o pré-processamento que é a etapa responsável por preparar a imagem para próximos passos, os filtros usados com maior frequência são a escala de cinza e a binarização da imagem.

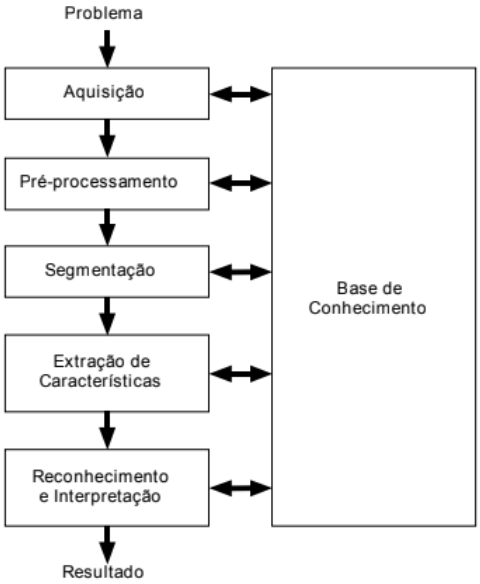

Figura 9 – Etapas básicas de um sistema de processamento de imagem (FILHO; NETO, 1999)

Na etapa de segmentação, os objetos do fundo da imagem são extraídos ou identificados, nesse processo, os *pixels* que possuem características semelhantes são agrupados. Após a segmentação é realizado o processo de extração de características, onde o objetivo é utilizar descritores para diferenciar com precisão entre os padrões que apresentam semelhanças (FILHO; NETO, 1999). É nesse etapa que o *ROI* é encontrado.

Na última etapa, no reconhecimento buscamos atribuir um rótulo a um objeto baseado em suas características, que são traduzidas pelos seus descritores. Já a interpretação busca atribuir um significado a um conjunto de objetos reconhecidos (FILHO; NETO, 1999). Após o *ROI* ser encontrado ele é classificado como o objeto correspondente as características encontradas.

Pressupõe que cada etapa descrita anteriormente tem um conhecimento sobre o problema a ser resolvido, armazenado em uma base de conhecimento. Essa base não deve somente guiar o funcionamento de cada etapa, mas também deve permitir a realimentação entre elas (FILHO; NETO, 1999).

3 MÉTODO DE PESQUISA

Neste capitulo se encontram todos os processos para a criação dos sistema desde a base de conhecimento até o resultado.

3.1 Base de conhecimento

Pressupõe que cada etapa descrita anteriormente tem um conhecimento sobre o problema a ser resolvido, armazenado em uma base de conhecimento. Essa base não deve somente guiar o funcionamento de cada etapa, mas também deve permitir a realimentação entre elas (FILHO; NETO, 1999).

Para o treinamento da base de conhecimento foram usadas cerca de 600 imagens de veículos, onde é possível encontrar diversos modelos e cores de veículos além da base conter imagens com placas no modelo do Mercosul que foi implantado recentemente, também conta com o modelo antigo e com as placas exclusivas, como o caso da placa para carro de coleção.

O treinamento para a detecção da placa nos veículos foi realizado pelo *YOLO*, sendo realizadas 6000 interações, essa quantidade de interações foi escolhida de acordo com o recomendado pelo *YOLO*, que é a quantidade de classes*2000, mas não menos que 6000 interações, finalizando o treinamento com uma taxa média de perda de 0.0162 e levando em média 3 horas para finaliza-lo utilizando a versão *tiny* do *YOLO*, podemos acompanhar a linha do tempo do treinamento através da Figura 10.

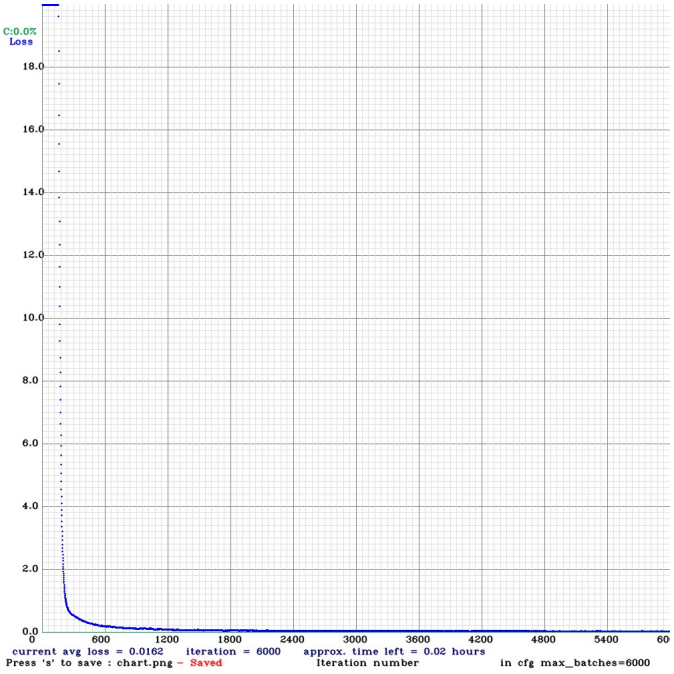

Figura 10 – Linha do tempo do treinamento.

3.2 Etapas do processamento de imagens

Todas as etapas podem ser acompanhadas através da figura 11. Inicialmente, foi realizada a aquisição de imagens realizada por meio de uma câmera cuja a localização, resolução e quantidade de quadros por segundo são desconhecidos, sendo que a grande maioria das placas apresentadas são do estado de Santa Catarina. Depois da aquisição da imagem é definido o valor padrão de resolução de 416X416 *pixels*.

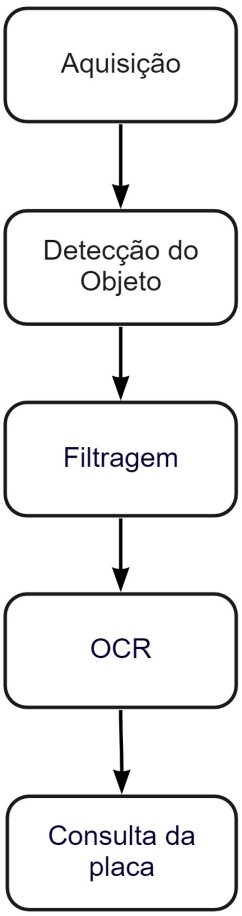

Figura 11 – *Pipeline* do projeto.

Na etapa de detecção, os objetos do fundo da imagem são extraídos ou identificados, nesse processo, os *pixels* que possuem características semelhantes são agrupados. O responsável por todo o processo de reconhecimento do objeto na imagem é o *YOLO*, sendo que após a varredura dele na imagem é possível obter todos os objetos e as suas posições na imagem. Após a detecção do objeto é realizado o processo de extração de características, onde o objetivo é utilizar descritores para diferenciar com precisão entre os padrões que apresentam semelhanças (FILHO; NETO, 1999). É nesse etapa que o *ROI* é encontrado. Na última etapa, no reconhecimento buscamos atribuir um rótulo a um objeto baseado em suas características, que são traduzidas pelos seus descritores. Já a interpretação busca atribuir um significado a um conjunto de objetos

reconhecidos (FILHO; NETO, 1999). Após o *ROI* ser encontrado ele é classificado como o objeto correspondente as características encontradas.

3.3 Filtragem

As imagens serão submetidas ao processo de pré-processamento, onde a imagem é realçada para evidenciar características de interesse. Tanto para a detecção da placa como para o *OCR* é feito o uso de técnicas de pré-processamento. Para a detecção, o *YOLO* é o responsável para realizar todo o processo, já para o *OCR* os filtros para realçar a imagem foram utilizados para deixar a imagem em escala de cinza, onde ela passa a ter 256 cores e após isso é feito a binarização da imagem para deixar ela com somente duas cores, o branco e o preto.

3.4 OCR

Após os passos de processamento de imagem, o *ROI* é encontrado e a partir dele é possível realizar o *OCR*. Mas antes disso são aplicados mais alguns filtros no *ROI*, que são filtros para deixar a imagem em escala de cinza e depois binarizada, ou seja, em preto e branco, com esses filtros aplicados é obtido um melhor destaque dos caracteres, o que facilita o *OCR*.

Para realizar o OCR de fato, foi utilizado o Pytesseract do Python, que é uma biblioteca do Tesseract. O Tesseract é uma ferramenta, de código aberto, para reconhecimento óptico de caracteres onde o desenvolvedor principal é Ray Smith e o mantenedor é Zdenko Podobny (Ray Smith, 2019).

3.5 Consulta da Placa

Após o *OCR* ser concluído, obtemos como resultado uma *string* com a placa. Essa *string* não é mais que um dado no formato de texto, com isso é possível realizar a consulta da placa através de uma *Application Programming Interface (API)* ou até mesmo diretamente em uma base de dados.

```
{
  "ano": "2010",
  "anoModelo": "2010",
  "chassi": "*****26407",
  "codigoRetorno": "0",
  "codigoSituacao": "0",
  "cor": "Vermelha",
  "data": "2021-01-25T09:35:46.73-03:00",
  "dataAtualizacaoAlarme": "",
  "dataAtualizacaoCaracteristicasVeiculo": "",
  "dataAtualizacaoRouboFurto": "",
  "extra": {},
  "marca": "VW/POLO 1.6 SPORTLINE",
  "mensagemRetorno": "Sem erros.",
  "modelo": "VW/POLO 1.6 SPORTLINE",
  "municipio": "Cambori\u00fa",
  "placa": "MHN8834",
  "situacao": "Sem restri\u00e7\u00e3o",
  "uf": "SC"
}
```


Figura 12 – Exemplo de consulta realizada via *API*

Para este trabalho foi utilizada uma *API* pública, acessível através da *Uniform Resource Locator (URL)* https://apicarros.com/v1/consulta/mhn1883/json onde é possível realizar algumas consultas sem custo e ela retorna os dados sobre a placa consultada, um exemplo de retorno desses dados pode ser visto acima onde foi realizada a consulta da placa presente na Figura 12.

4 EXPERIMENTOS

Para o treinamento da detecção das placas utilizando o *YOLO* v4 foi utilizado um computador com as seguintes configurações: Processador AMD Ryzen 3 3200G 3.60 GHz, integrado ao processador uma placa de vídeo AMD Radeon Vega 2 GB, contando com memória *RAM* de 8 GB com uma frequência de 2666 MHz DDR4. Porém apenas 5.93 GB foi disponível pois 2 GB foram alocados para a *GPU*. O Sistema Operacional utilizado foi o *Windows* 10 *Home*, porém o treinamento levava cerca de 130 horas e não conseguia obter um resultado viável ou ocorria problemas durante o treinamento.

Devido a esse problema, apenas o treinamento foi realizado através de outro computador que possui mais recursos, que possui as seguintes configurações: Processador AMD Ryzen 5 1600 3.2 GHz, uma *GPU* GTX 1060 com 3GB de *VRAM*, contando com 16 GB de memória *RAM* com um frequência de 3200 MHz mas o processador limita o uso em 2666 MHz DDR4 e o Sistema Operacional também é o *Windows* 10 mas na versão Pro.

Com o treinamento realizado neste computador, o tempo de treinamento foi reduzido para 3 horas e o *YOLO* v4 obteve uma taxa de detecção melhor como pode ser observado através da figura 10. Vale lembrar que o treinamento foi realizado levando em consideração somente o objeto placa e não foi separado entre objeto placa no padrão Mercosul e objeto placa no padrão do Brasil, sendo assim, o detector trata os dois tipos de placas como o mesmo objeto.

As imagens utilizadas no trabalho foram coletadas através de uma câmera posta em uma via com localização desconhecida, tendo sua resolução fixa em 416X416 *pixels*, para a obtenção dos resultados foram utilizados os 50 primeiros veículos que passaram pela via. Após a aquisição é realizada detecção da placa na imagem para obter o *ROI*, no caso de mais de uma placa na imagem, é obtido um *ROI* para cada placa. No início dos experimentos o detector estava considerando alguns falsos positivos, no qual ele acreditava que alguns carros eram o objeto a ser encontrado, porém com uma baixa taxa de certeza que o carro seria o objeto que o detector buscava, com isso foi necessário a aplicação de verificação sobre a taxa de certeza para o objeto, que foi configurada para aceitar objetos com no mínimo 0,90 ou seja, 90% de certeza que o objeto encontrado é de fato o objeto que estamos procurando.

Uma vez que a região de interesse foi encontrada foram realizados testes do *OCR* com a placa, porém os resultados obtidos não foram satisfatórios, o que resultou na inclusão de uma etapa de pré processamento da imagem antes da realização do *OCR*. Nessa etapa foram aplicados os filtros de escala de cinza e binarização da imagem, onde os melhores resultados foram obtidos, também foram realizados testes aplicando reajuste no tamanho da imagem, onde ela foi aumentada, porém os resultados pioraram.

Após concluir as etapas de pré processamento e *OCR* é realizada a consulta da placa

via *API*, a qualidade da consulta está ligada fortemente com a precisão da etapa anterior, onde é retornada a placa no formato de texto. Podemos visualizar melhor essas etapas através das Figuras 4, 6, 8 e 12.

Nos experimentos usando vídeos a taxa de quadros por segundo ficou na média de 15 com um porém, quando placas são detectadas, a taxa cai para menos de 10 quadros por segundo, afetando negativamente o processo do algoritmo. Essa baixa taxa de quadros por segundo é causada pelo *hardware* do computador que não é compatível com a otimização do *YOLO* que existe somente para *GPUs* da *NVIDIA* e também pelo hardware do computador utilizado não ser o mais recente do mercado, tudo isso afeta o desempenho do algoritmo.

Quando os experimentos foram realizados no computador com *hardware* mais potente que foi responsável por realizar o treinamento a taxa de quadros por segundo aumento para cerca de 25 mas o problema de queda nos quadro continuou acontecendo. Pelo *YOLO* se tratar do estado da arte na detecção, apesar das melhorias vem recebendo com as atualizações de versões ele ainda não é um método que exige pouco do *hardware* do computador.

5 RESULTADOS

O primeiro resultado obtido é referente a detecção da placa no quadro, dentre os 50 veículos que foram analisados todos tiveram sua placas detectadas, resultando em uma precisão de 100% para a detecção, dentre os veículos analisados estão 43 carros, 5 motos e 2 caminhões.

Após a detecção das placas na imagem é possível analisar os resultados do processo do *OCR*, onde o resultado não foi tão positivo quanto a detecção. Como podemos ver na figura 13, dentre as 50 placas o *OCR* conseguiu acertar todos os caracteres da placa em apenas 18 casos, o que representa 36% das placas.

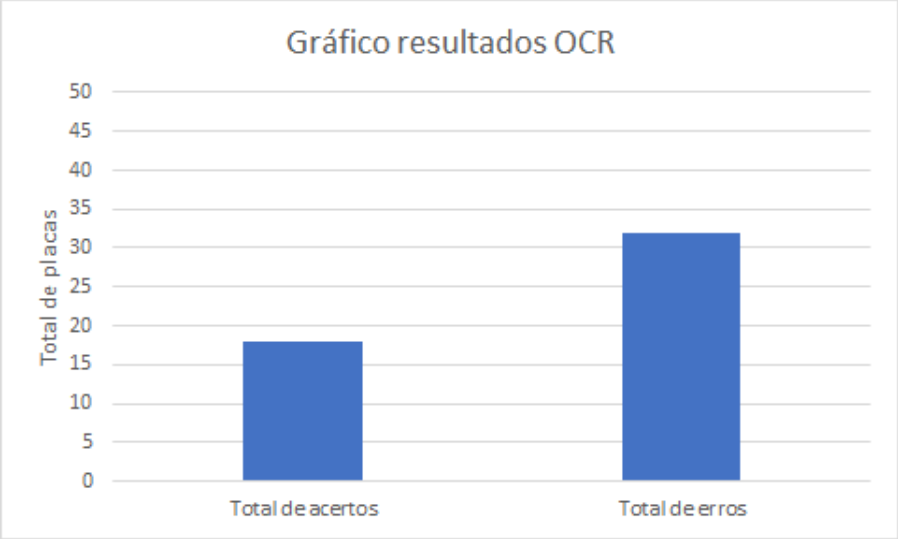

Figura 13 – Taxa de acerto do *OCR*

Dentre os 32 casos de erros do *OCR*, 28 foram em carros, 4 em motos e não ocorreram erros ao analisar as placas dos caminhões, vale ressaltar que a maior incidência de erros em carros pode ser relacionado a maioria dos veículos analisados serem carros. Dentre os 28 erros identificados nas placas de carros, sendo que 18 dos erros foram de interpretação do *OCR*, ou seja, os casos em que as letras ou números foram trocados, a maior parte dos erros estão nas letras, sendo que as mais trocadas foram a letra I pelo número 1, a letra U por V e a letra V por U. Através da Figura 14 podemos ver a quantidade de erros relacionados com a interpretação de letras, números e erros em ambos simultaneamente.

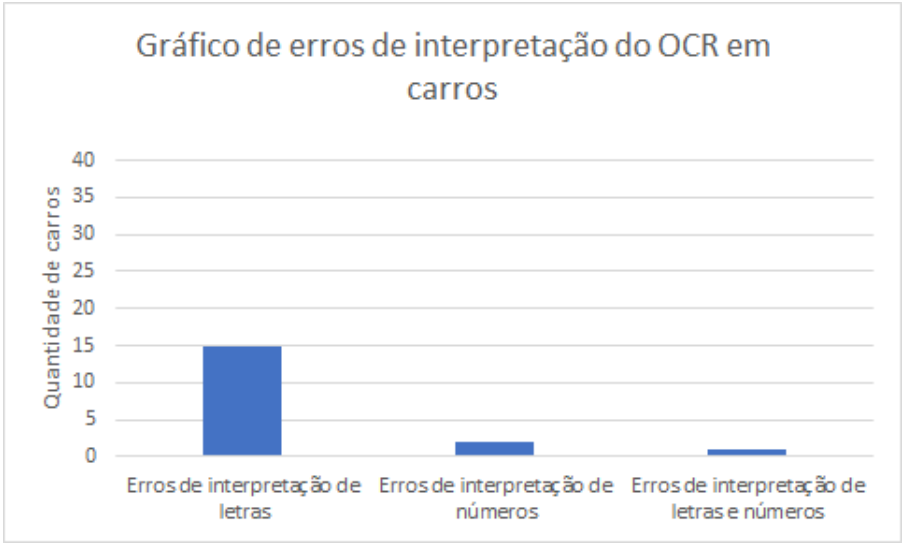

Figura 14 – Erros de interpretação do *OCR* em carros

O restante dos erros do *OCR* identificados nas placas de carros foram na detecção dos caracteres, ou seja, em casos que foram encontrados caracteres a mais ou a menos na placa. Como é visível na Figura 15, em 6 casos esta faltando pelo menos 1 caractere e em 4 casos esta sobrando pelo menos 1 caractere.

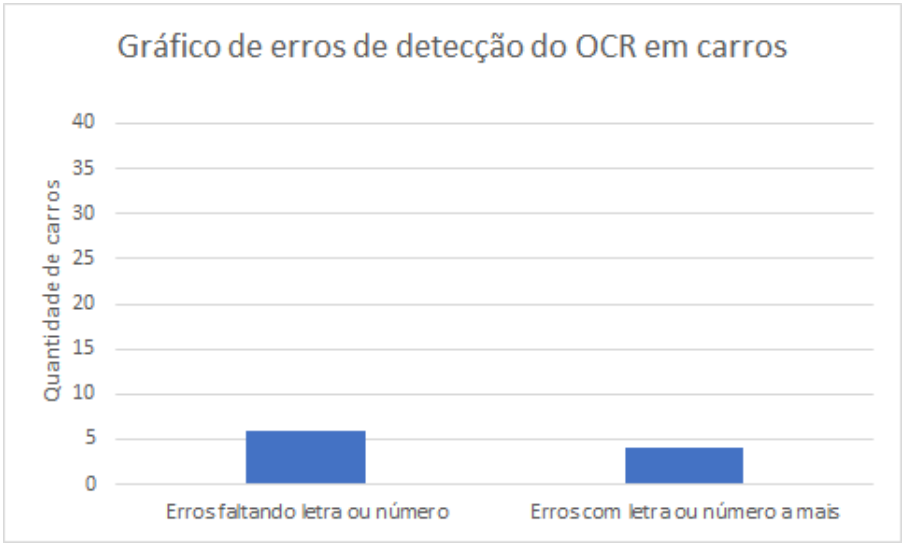

Figura 15 – Erros de detecção do *OCR* em carros.

Dos 4 erros identificados no *OCR* das placas de motos, apenas um foi de interpretação dos caracteres da placa, onde foi identificado a letra A no lugar do que seria o número 4.

Nos outros três casos de erros identificados nas motos, foram de detecção de caracteres, onde somente as letras não foram identificadas na placa e os números foram identificados corretamente.

A taxa de sucesso da consulta da placa na *API* está fortemente ligada ao *OCR*, em todos

casos a placa foi consultada e quando o *OCR* foi realizado com sucesso, a consulta da placa retornava os dados corretos referente ao veículo.

6 CONSIDERAÇÕES FINAIS

Conclui-se que o método utilizado para a identificação da placa na imagem com o intuito de obter o *ROI* obteve um ótimo resultado, porém os métodos utilizados para o *OCR* não tiveram um resultado tão satisfatório quanto a detecção, o que acabou afetando a qualidade da consulta da placa do veículo via *API*. O resultado da identificação dos caracteres melhorou após a aplicação dos filtros de escala de cinza e de binarização da imagem, conseguindo identificar mais caracteres. Mesmo com resultados aceitáveis, existe pontos a serem melhorados, principalmente para a identificação dos caracteres.

O principal problema encontrado na identificação dos caracteres está relacionado com as letras da placa, nesse caso, é necessário analisar técnicas de segmentação que consigam dar um maior realce nos caracteres. Com isso, o resultado do *OCR* pode ter uma melhora significativa.

Para trabalhos futuros, é necessário ser realizado uma tratativa para que o *OCR* obtenha uma taxa de acerto maior, como por exemplo treinar um alfabeto. Outra alternativa é trabalhar com *whitelist* separadas para partes da placa, como por exemplo usar um conjunto de caracteres para as 3 primeiras posições da placa que sempre serão letras e usar outro conjunto para as outras posições que podem ser somente números ou números e letras dependendo do padrão da placa, com essa abordagem a taxa de sucesso pode melhorar.

REFERÊNCIAS

ALESSANDRO FARIA. *Conheçam o YOLO v4, o estado da arte em Visão Computacional*. 2021. Disponível em: <https://www.i2ai.org/content/blog/2021/1/conhecam-o-yolo-v4-o-estado-da-arte-em-visao-compu/>. Acesso em: 06 jun. 2021.

Alexey Bochkovskiy. *Yolo v4, v3 and v2 for Windows and Linux*. 2021. Disponível em: <https://github.com/AlexeyAB/darknet>. Acesso em: 27 abr. 2021.

Amikelive. *What Object Categories / Labels Are In COCO Dataset?* 2018. Disponível em: <https://tech.amikelive.com/node-718/what-object-categories-labels-are-in-coco-dataset/>. Acesso em: 16 set. 2021.

BORGES, K. K. W. *Autenticação veicular de duas etapas utilizando placa e cor*. Dissertação (TCC em Ciência da Computação) — Universidade Estadual do Paraná, 2018.

Cadernos de Tipografia. *A legibilidade das letras para leitura automática*. 2007. Disponível em: <http://www.tipografos.net/cadernos/cadernos03.html>. Acesso em: 14 abr. 2021.

CLAVERO, P. V. de O. B. *Analise dos descritores de furos e área em caracteres cursivos*. Dissertação (TCC em Ciência da Computação) — Universidade Estadual do Paraná, 2019.

FERREIRA, R. F. *Reconhecimento e diferenciação de sinalizações de trânsito brasileiras por imagem*. Dissertação (TCC em Ciência da Computação) — Universidade Estadual do Paraná, 2018.

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

GONZALEZ, R. C.; WOODS, R. E. *Processamento Digital de Imagens*. São Paulo: Pearson, 2010. ISBN 9788581435862.

Governo do Estado do Paraná. *Paraná está entre os estados com menores taxas de furtos e roubos de veículos*. 2019. Disponível em: <http://www.aen.pr.gov.br/modules/noticias/article.php?storyid=103800&tit=Parana-esta-entre-os-estados-com-menores-taxas-de-furtos-e-roubos-de-veiculos>. Acesso em: 12 abr. 2021.

Instituto Nacional de Pesquisas Espaciais. *Teoria: Processamento de Imagens*. s.d. Disponível em: <http://www.dpi.inpe.br/spring/teoria/filtrage/filtragem.htm>. Acesso em: 1 set. 2021.

Lucas Agrela. *Brasileiro ajuda IBM e Nvidia a dar olhos para a computação*. 2017. Disponível em: <https://exame.com/tecnologia/brasileiro-ajuda-ibm-e-nvidia-a-dar-olhos-para-a-computacao/>. Acesso em: 28 jul. 2021.

PEDRINI, H. Introdução ao processamento digital de imagem mc920 / mo443. *Unicamp*, 2021.

Ray Smith. *Tesseract Open Source OCR Engine (main repository)*. 2019. Disponível em: <https://github.com/tesseract-ocr/tesseract>. Acesso em: 30 abr. 2021.

SILVA, I. N. da; SPATTI, D. H.; FLAUZINO, R. A. *Redes Neurais Artificiais para engenharia e ciências aplicadas*. São Paulo: ArtLiber, 2016. ISBN 9788588098879.

TURING, A. M. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX, n. 236, p. 433–460, 10 1950. ISSN 0026-4423. Disponível em: <https://doi.org/10.1093/mind/LIX.236.433>.

VARGAS, A. C. G.; PAES, A.; VASCONCELOS, C. N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. In: SN. *Proceedings of the xxix conference on graphics, patterns and images*. [S.l.], 2016. v. 1, n. 4.