Pró-reitoria de Pesquisa e Pós-Graduação
INSTITUTO FEDERAL São Paulo Câmpus Cubatão

12º Congresso de Inovação, Ciência e Tecnologia do IFSP - 2021

Reconhecimento Automático de Placas de Veículos Automotores

ABNER HENRIQUE DOS SANTOS SIMAS[1], ANDERSON DUARTE BETIOL[2]

[1]Graduando em Bacharel de Engenharia da Computação, IFSP, Câmpus Birigui, abner.h@aluno.ifsp.edu.br
[2]Mestre em Engenharia Elétrica, Docente, IFSP, Câmpus Birigui, betiol.anderson@ifsp.edu.br

Área de conhecimento (Tabela CNPq): 1.03.03.05-7 Processamento Gráfico (*Graphics*)

RESUMO: Este artigo apresenta o desenvolvimento de um sistema de reconhecimento automático de placas veiculares, que pode ser útil para a identificação de veículos, através de imagens capturadas por câmeras. As imagens utilizadas no trabalho foram adquiridas por smartphones em diferentes condições de ambientes, aproximando os experimentos de um ambiente real. O sistema utiliza um algoritmo de redes neurais convolucionais conhecido como YOLO, que foi treinado utilizando uma parte das imagens adquiridas, para localizar a placa em uma imagem e então extraí-la. A imagem da placa é binarizada por um algoritmo de limiar adaptativo e os caracteres são segmentados a partir da projeção vertical dos pixels, para encontrar o início e o fim de cada caractere. O tempo médio de execução para detecção da placa e segmentação dos caracteres é de 1,40 segundo, em um processador dual-core 2.5GHz e GPU integrada, com acurácia de 82,4%. As próximas etapas do trabalho consistem em melhorar a acurácia da segmentação e desenvolver a etapa de reconhecimento dos caracteres.

PALAVRAS-CHAVE: imagens; redes neurais; detecção da placa; segmentação.

Automatic License Plate Recognition for Motor Vehicles

ABSTRACT: This article presents the development of an automatic license plate recognition system, which can be useful for vehicle identification, through images captured by cameras. The images used in the work were acquired by smartphones in different environmental conditions, bringing the experiments closer to a real environment. The system uses a convolutional neural network algorithm known as YOLO, which has been trained using a portion of the acquired images, to locate the license plate in an image and then extract it. The plate image is binarized by an adaptive threshold algorithm and characters are segmented from the vertical projection of pixels to find the beginning and end of each character. Average runtime for plate detection and character selection is 1.40 seconds, on a 2.5 GHz dual-core processor and integrated GPU, with 82.4% accuracy. The next steps of the work are to improve the production accuracy and develop a character recognition step.

KEYWORDS: images; neural networks; plate detection; segmentation.

INTRODUÇÃO

O processo de reconhecimento de placas de automóveis, que pode ser automatizado a partir de sistemas computacionais, é de grande importância em diferentes situações, como identificação de veículos roubados ou irregulares, cobrança automática em pedágios, aplicação da lei de trânsito e controle de acesso a estacionamentos (DU et al., 2012).

Segundo (DU et al., 2012), o Reconhecimento Automático de Placas Veiculares *(ALPR)* é realizado em imagens capturadas por uma câmera, sendo capaz de identificar e ler a placa de um ou mais veículos. Em geral, esse processo é constituído de quatro etapas, aquisição da imagem, detecção da placa, segmentação dos caracteres e reconhecimento dos caracteres segmentados.

Um sistema ALPR deve ser capaz de processar imagens com condições adversas que provem do ambiente real, como, por exemplo, diferentes iluminações, sombras, inclinações, etc (FERNANDES, 2019).

Diante do exposto, este trabalho tem o objetivo de desenvolver um sistema ALPR capaz de reconhecer placas no padrão brasileiro de veículos a partir de imagens dos carros.

MATERIAIS E MÉTODOS

Inicialmente, foram adquiridas 205 imagens digitais de veículos diferentes com boa visão da placa, realizada por câmeras de smartphones em ambientes não controlados, que serviram tanto como base para treinamento da rede neural como para testes de execução. Essas imagens apresentam diferentes resoluções e condições.

De acordo com Gonzalez e Woods (2009), "uma imagem pode ser definida como uma função bidimensional, $f(x, y)$, em que x e y são coordenadas *espaciais* (plano), e a amplitude de f em qualquer par de coordenadas (x,y) é chamada de *intensidade* ou *nível de cinza* da imagem nesse ponto". As coordenadas x e y tem domínio em $\{1, ..., m\} \times \{1, ..., n\} \subseteq \mathbb{N} \times \mathbb{N}$, onde m e n são, respectivamente, a largura e a altura da imagem.

A partir das imagens armazenadas, foi feito o treinamento do YOLO (*You Only Look Once*), um algoritmo de redes neurais convolucionais para detecção de objetos em imagens, utilizando um conjunto de 51 imagens de exemplo, na configuração padrão do YOLOv3. O processamento foi realizado na plataforma Google Colab em Python 3.

Ao fim do treinamento, obteve-se um arquivo com os pesos sinápticos da rede neural e então foram realizados testes em todas as imagens. A Figura 1 apresenta um exemplo de placa localizada pelo algoritmo.

Após a obtenção da placa, é necessário segmentar os caracteres, utilizando processamento de imagens com *OpenCV*. O primeiro passo é o redimensionamento da imagem para 90×30 pixels, a fim de padronizar o tamanho das placas, mantendo a proporção de 40×13 centímetros das placas em tamanho real, e em seguida é feita a conversão de cores para a escala de cinza, dada pela equação:

$$Y = 0.299 * R + 0.587 * G + 0.114 * B \tag{1}$$

em que,

Y - intensidade de luminosidade;

R, G, B - intensidade das cores vermelha, verde e azul, respectivamente.

O segundo passo é a binarização da imagem, que foi realizada utilizando um limiar adaptativo,

Figura 1: Exemplo de placa localizada na imagem.

que calcula um limiar para cada vizinhança de pixels, tendo um resultado melhor que um limiar constante, visto que as imagens possuem diferentes graus de intensidade. A função utilizada é a *cv2.adaptiveThreshold*, na qual cada pixel é definido como 0 (preto), caso sua intensidade seja menor que o limiar, ou como 255 (branco), caso contrário. A Figura 2 apresenta a imagem original da placa em escala de cinza (esquerda) e binarizada (direita).

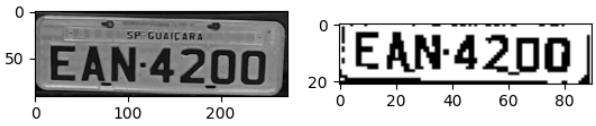

Figura 2: Placa extraída em escala de cinza (esquerda) e posteriormente recortada e binarizada (direita)

Observando-se as medidas das placas, pode-se recortar cerca de 30% do topo da placa, tamanho aproximado da sarjeta. Em seguida são feitas a projeção vertical e a projeção horizontal da imagem, a fim de identificar os espaços em que estão os caracteres. A projeção horizontal, que é a contagem de pixels brancos em cada linha, auxilia a encontrar os espaços em branco em cima e embaixo dos caracteres, para remover essas parte da imagem, enquanto a projeção vertical, contagem de pixels pretos em cada coluna, é responsável por localizar a posição de cada caractere.

A Figura 3 apresenta as projeções horizontais de duas placas, enquanto abaixo se encontram as respectivas placas depois de feitos os corte superior e inferior. As barras vermelhas representam as coordenadas em y (verticais) nas quais a imagem é recortada.

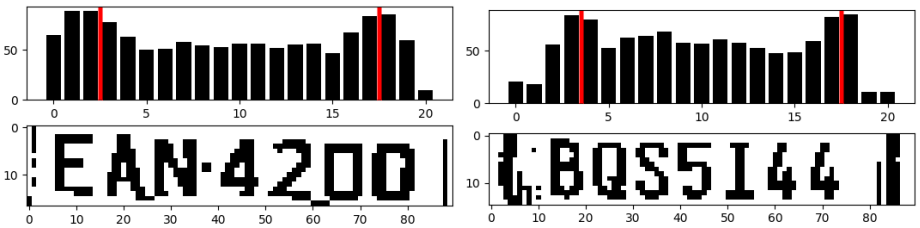

Figura 3: Projeção horizontal de duas placas.

Após os recortes, é feita a projeção vertical, onde são encontradas as coordenadas em x para se recortar cada caractere. Nessa projeção, essas coordenadas são definidas pelas posições das colunas nas quais a quantidade de pixels é mínima, pois identificam um espaço em branco, que é o limite lateral de um caractere. Pode-se observar na Figura 4 a projeção vertical das duas placas utilizadas

anteriormente, e as barras vermelhas que representam os limites que separam cada caractere.

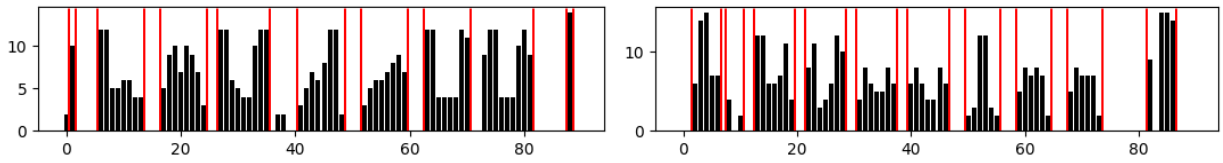

Figura 4: Projeção vertical de duas placas.

Analisando as projeções verticais, identifica-se a necessidade de tratar as segmentações, para eliminar recortes que não são caracteres de fato, mas sim bordas ou ruídos, assim como separar caracteres unidos. É utilizado então um algoritmo que calcula a largura de cada segmento e, caso algum seja maior que 15% da largura da imagem, o mesmo é divido no meio, a fim de separar os caracteres unidos, assim como um algoritmo que conta a quantidade de recortes feitos e, caso haja mais que 7 segmentos, são eliminados os excedentes, obtendo um padrão de 3 segmentos na primeira metade da imagem, e 4 na segunda metade.

A Figura 5 apresenta a localização dos caracteres antes e depois do tratamento da projeção vertical, no qual são eliminados falsos positivos de caracteres, incluindo o hífen, para padronizar as placas que possuem e que não possuem o hífen.

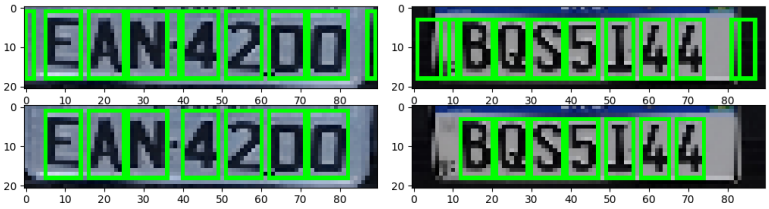

Figura 5: Localização dos caracteres em duas placas.

Por fim, cada digito é segmentado em uma lista de imagens, de forma com que possam ser reconhecidos individualmente e fornecer a saída de texto.

RESULTADOS E DISCUSSÃO

Os experimentos foram executados em um notebook com processador i5-7200U 2.5GHz, GPU HD Graphics e 8GB de RAM, utilizando todas as imagens obtidas. A Tabela 1 apresenta os resultados obtidos, com a porcentagem de acertos e tempo médio de execução de cada etapa.

Tabela 1: Resultados dos experimentos por etapa e o tempo de execução.

Etapa	Acertos (%)	Tempo de execução (s)
Detecção da placa	98,5	1,40
Segmentação dos caracteres	82,4	0,01

O algoritmo de detecção da placa utilizando YOLOv3 foi muito eficiente nos testes, até mesmo em placas com algumas inclinações, tendo maior dificuldade com veículos distantes da câmera. No entanto, não foram incluídas placas de motos, ou placas com cores diferentes nos experimentos.

A etapa de segmentação dos caracteres obteve ótimos resultados na maioria das placas extraídas, errando em apenas 36 das mesmas. Os erros mais comuns foram segmentação de um caractere em duas

partes, como por exemplo as letra 'H' e 'U', que possuem poucos pixels no centro, ou então recortar em excesso a parte de cima ou de baixo, principalmente causado por inclinação na placa. A Figura 6 apresenta um erro de segmentação, onde o 'H' é cortado no meio, e então o algoritmo elimina a letra 'C', a fim de eliminar os excedentes de 7 segmentos.

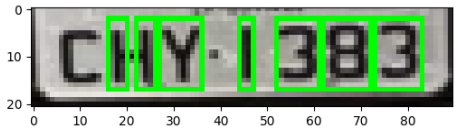

Figura 6: Localização incorreta dos caracteres.

A Figura 7 demonstra o resultado final da segmentação correta da placa de um veículo, na qual deve ser aplicado o reconhecimento de cada caractere.

Figura 7: Segmentação correta dos caracteres.

CONCLUSÕES

De acordo com os resultados obtidos neste trabalho, a partir das metodologias utilizadas, foi possível desenvolver um sistema de reconhecimento e segmentação dos caracteres de placas de veículos brasileiros com um tempo médio de execução em um notebook de 1,40 segundo e acurácia de 82,4%. Foram encontrados problemas como falha na localização de placas distantes da câmera, assim como falha na segmentação de placas inclinadas ou com letras 'H' e 'U'.

As próximas etapas do trabalho consistirão em tratar desses problemas e desenvolver a etapa de reconhecimento dos caracteres para obtê-los no padrão ASCII, de forma com que o sistema possa ser implantado no controle ou monitoramento de veículos.

AGRADECIMENTOS

Agradeço aos amigos, colegas e namorada que colaboraram com a aquisição de imagens de veículos e pelo incentivo e motivação dados. Ao Programa Institucional Voluntário de Iniciação Científica e/ou Tecnológica (PIVICT) do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP).

REFERÊNCIAS

DU, S. et al. Automatic license plate recognition (ALPR): A state-of-the-art review. *IEEE Transactions on circuits and systems for video technology*, IEEE, v. 23, n. 2, p. 311–325, 2012.

FERNANDES, L. d. S. Reconhecimento automático de placas veiculares brasileiras incorporando max-margin object detection e redes neurais convolucionais. 2019.

GONZALEZ, R. C.; WOODS, R. C. *Processamento digital de imagens* . [S.l.]: Pearson Educación, 2009.