Detection and recognition of car plates

*Algorithm of identification of characters

Fábio Morais
up201504257

Alberto Santos
up201708979

Rui Coutinho
up201503006

***Abstract*—This assignment was focused on the detection and recognition of cars' license plates. The first phase of this assignment intended to detect the Region of Interest (ROI) from a given image. The second one had the objective of recognising the numbers and the letters using for that the template matching technique. The last phase was focused on improving the accuracy of the previous two steps. The license plate recognition is a highly method used nowadays used, for example, by the police, to catch the offenders that drive over the speed limit.**

***Index Terms*—License plate recognition, Template matching**

I. Introdução

O objetivo deste trabalho consiste na deteção e reconhecimento de matrículas. Numa primeira fase deteta-se a Região de Interesse criando uma máscara dessa mesma região, na segunda fase usa-se uma máscara já fornecida para detetar os números e as letras através de um sistema de semelhança por template matching e no final, um algoritmo para melhorar a precisão dos métodos anteriores, tornando o trabalho, no seu global mais imune a variações de fotografias utilizadas.

II. Task 1

O objetivo desta tarefa é a identificação das matrículas numa imagem. Onde o utilizador dá uma imagem da parte traseira do carro e o programa desenha um rectângulo azul à volta da matrícula.

A. Pré Processamento da imagem

Numa primeira instância, faz-se uma equalização de histograma baseada em CLAHE para aumentar o contraste da imagem. Em seguida faz-se *sharpening* da imagem para tentar obter melhores contornos.

Fig. 1: Exemplo da equalização de histograma (imagem 6)

Em segunda instância, fazemos uma binarização da imagem baseada no método de Otsu. No entanto a binarização não é feita sobre a imagem depois da equalização. Em vez disso é dividida a imagem original no sistema de cores HSV onde todos os pixeis que não tenham uma componente S (saturation) menor de 0.65 e uma componente V (value) maior do que 0.35 terão um valor de 0 na imagem de entrada para a binarização. Isto foi feito pelo facto que o fundo da matrícula é branco e daí não interessa a cor (componente H).
Como se pode constatar na figura abaixo, embora o carro seja vermelho e com bastante luz, resultando numa imagem em escalas de cinzento clara, ao fazer isto uma boa parte do carro fica com o valor 0, pois não é branco.

Fig. 2: Exemplo deste processo (imagem 35)

B. Processamento da imagem

Depois da binarização da imagem feita, faz-se duas operações de abertura, uma com um elemento estruturante que é uma linha vertical com 5 pixeis de comprimento e, uma outra, com um elemento estrutural que é a vizinhança N4.
De seguida faz-se um *labelling* dos elementos e remove-se alguns elementos baseado no seu tamanho, forma e posição.

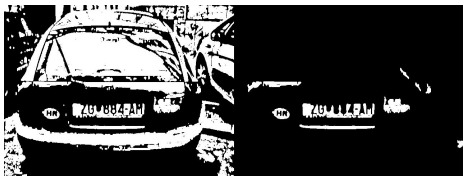

Fig. 3: Exemplo deste processo (imagem 7)

Posteriormente, faz se duas operações de fecho seguidas de duas operações de abertura, com elementos estruturais em linha primeiro horizontais e depois verticais. Este processo tem como intuito ''fechar'' as matrículas tentando manter a forma retangular inerente às matrículas.

Fig. 4: Exemplo deste processo (imagem 7)

C. Reconhecimento da matrícula

De maneira a poder-se detetar a matrícula, faz-se, primeiramente, um *labelling* da imagem, vinda do processo anterior.
De seguida, baseado nas propriedades de cada *label* , mais concretamente o tamanho, a área, a concentração, quantidade de pixeis existentes no retângulo que circunscreve a a mesma **(box)** e as suas medidas, procura-se se existe alguma matrícula.
Na existência de várias possíveis matrículas, é atribuída uma pontuação a cada *label* baseada nos pontos e mais dois critérios, a orientação e a *Solidity*, que é bastante parecida com a concentração, mas em vez de ser um retângulo é o polígono convexo mais pequeno possível. No fim a região com melhor pontuação é escolhida como a região/*label* da matrícula.

$$points = (1 - \frac{|area - 8700|}{8700}) * 0.25 \ + \ conc * 0.25 + \\ (2 - \frac{|x - 215|}{215} - \frac{|y - 43|}{43}) * 0.45 - \\ |angle| * 0.05 \ + \ sol * 0.1 \tag{1}$$

Onde a **area** é calculada pela área da box, a **conc** é calculada pelo número de pixeis da região a dividir pela **area**, **x** e **y** são o comprimento e altura, respetivamente, da box, **angle** é o ângulo da região e **sol** é a *Solidity* da região.

D. Resultados do algoritmo de deteção

O algoritmo deteta todas as matrículas e foi usado o método de Jaccard onde obteve-se os seguintes resultados:

	Max	Min	Mean
Jaccard	97.38%	70.72%	87.12%

Esta tarefa foi concluída com uma boa precisão e com ótimos resultados, podendo-se ver todas as letras e números das matrículas, no entanto demora algum tempo a ser executado e, como tal, podia ser optimizado.
Este algoritmo poderia ser optimizado usando um pós processamento apenas na zona de interesse, zona onde a matrícula foi encontrada, de maneira a aumentar ainda mais o índece de Jaccard.

III. Task 2

O objetivo desta tarefa é detetar as letras da matrícula e dar como output a matrícula num ficheiro de texto.

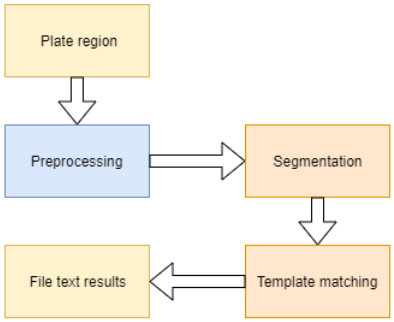

Fig. 5: Abordagem para a task 2

A. Pré Processamento

O primeiro passo após adquirir a imagem foi converter para double, pois é essencial trabalhar com uma gama alta de valores, para não perdermos informação nas operações que iremos realizar com a imagem. O próximo passo foi redimensionar a imagem para [100,600] e converter a imagem para grayscale com a função rgb2gray, convertendo a imagem RGB para grayscale.

1) Remover ruído: Para retirarmos o possível ruído que a imagem possa ter foi usado um filtro não linear, usando a técnica do Median Filter, com o auxilio da função medfilt2 do matlab, em que consiste determinar a mediana da imagem obtida, eliminando assim algum ruído.

B. Morfologia

1) Binarização: O processo de binarização converte a imagem para preto e branco, com o recurso a um thresholding que podemos definir para este classificar como branco, ou como preto. Escolher o melhor thresholding é fulcral para ter uma grande percentagem de sucesso no reconhecimento das letras, mas como as imagens têm diferentes intensidade não conseguimos definir um thresholding constante, desta forma usamos a função graythresh, que usa o método Otsu's, este método escolhe um thresholding que minimiza a variação intra classe dos pixeis pretos e brancos. [1]

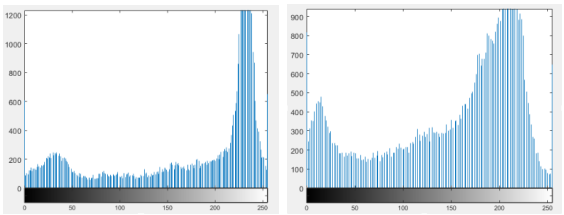

Fig. 6: Histograma de 2 imagens diferentes

Com estes histogramas podemos verificar que se escolhermos um thresholding fixo teríamos bastantes problemas, pois a gama de valores varia bastante e o que pode ser bom para uma imagem, pode não ser para outra.

2) Dilatação e erosão: Foi necessário melhorar a imagem binária, usamos por isso o close para remover os pequenos pontos pretos na imagem binária e termos as letras mais visíveis. Este processo faz a operação de dilatação e em seguida uma erosão.

C. Segmentação

1) Edge detection: Esta tecnica determina pontos de uma imagem em que a intensidade muda repentinamente, existem vários algoritmos que podemos usar, Sobel, Canny, Prewit, Roberts ... O algoritmo escolhido foi o Canny, este detector utiliza o algoritmo multi-estágios para detetar ampla margem de bordas na imagem, sendo este o metodo com menores taxas de erro neste caso, porém exige maior esforço [2] computacional.

- Redução de ruído
- Calcular gradiente
- Supressão dos não máximos
- Double thresholding
- Rastreamento de borda por histerese

Fig. 7: Imagem com deteção de bordas 'canny'

2) Morfologia: Depois de detetar as bordas das letras foi necessário recorrer mais uma vez a morfologia, para desta forma termos a certeza que as bordas estão todas ligadas. Fazendo assim uma dilatação com um retângulo como elemento estruturante de 2 por 2 e em seguida uma operação de close, ligando assim as bordas que não tenham sido ligadas.

Fig. 8: Imagem depois do fill

Como podemos verificar, esta operação ao preencher as bordas perdemos as formas interiores das letras, uma abordagem para conseguirmos recuperar isso foi fazer operações com esta imagem e a imagem binária anteriormente obtida.

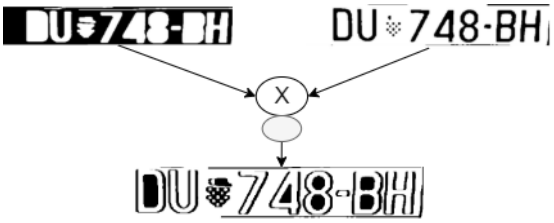

Fig. 9: Operação entre imagem binária e imagem com fill

Multiplicação entre imagem binária com a imagem da figura anterior e em seguida fazer o complemento, obtendo assim uma imagem binária com as formas interiores.

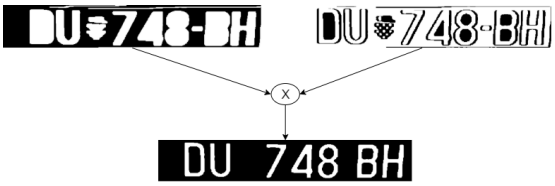

Fig. 10: Imagem depois do fill

Multiplicação entre imagem da figura anterior e a imagem com fill, obtendo assim a imagem final. Com esta abordagem foi possível apenas extrair as letras, que era o nosso grande objetivo.

3) Isolamento das letras: Para fazermos o isolamento das letras teremos de ligar os componentes de cada carácter, usando assim a função bwlabel. Com esta função iremos saber quantos objetos existem na imagem e a localização destes. Esta técnica permite detetar as regiões conectadas de uma imagem binária.

Fig. 11: regionprops

D. Reconhecimento de letras

Para fazermos o reconhecimento das letras foi usado o template matching, este método permite detetar pequenos objetos fixos na imagem. Pode ser usado em controlo de qualidade [3], entre muitas outras aplicações.

Foi usada a função corr2, em que retorna o coeficiente de correlação de 2 imagens. Desta forma percorremos os templates todos e retiramos o que tem maior índice de correlação.

Esta técnica tem algumas limitações e pode errar no seu reconhecimento [4], pois é necessário que a imagem não varia na sua rotação, mudança de escala, perspectiva...etc

A abordagem para este ponto foi criar uma biblioteca de caracteres alfanuméricos com uma resolução de 42x24 px.

Como o template matching é uma técnica que exige o mesmo tamanho nas 2 fotos a comparar foi necessário acrescentar ao programa uma ferramenta que padronize a imagem e redimensione para os 42x24 px.

Esta técnica, como era de esperar, obteve alguns erros, pois os caracteres alfabéticos que se assemelhavam a caracteres numéricos eram confundidos, como é o caso do 1 e I. Para melhorarmos o algoritmo fizemos outras abordagens para alem do template matching.

1) Matrícula padrão: Foi melhorado o nosso algoritmo, pois sabemos à prior qual o formato da matrícula a analisar, sendo que existe 4 possibilidades de matrículas:

- XX 111 X
- XX 111 XX
- XX 1111 X
- XX 1111 XX

Desta forma, conseguimos garantir que os 2 primeiros e o ultimo índice será sempre um carácter alfabético, enquanto que os os caracteres do meio serão sempre caracteres numéricos.

2) Contagem de buracos: Para conseguirmos distinguir um 6 de um G foi necessário recorrer a mais outro metodo para alem do template matching, foi usado a função bweuler que devolve o numero total de objetos numa imagem menos o numero total de buracos nessa imagem.

E. Resultados

40_plate.png	SI661BM	ST661BM	diferente
22_plate.png	ZG990NL	ZG0N	diferente

Fig. 12: Matrículas não identificadas

Na imagem 22 o pré processamento não chegou para garantir uma boa imagem para a segmentação, ficando a imagem binária com alguns cortes.

Fig. 13: Letras com imperfeições na imagem 22

Ao tentar remover pequenos objetos na imagem com a função bwareaopen tambem removemos estes caracteres, por serem menores que os 600px.

Fig. 14: Partes ambíguas de caracteres. [5]

É necessário garantir uns bons templates para estas partes ambíguas dos caracteres serem distinguidos.

Esta task foi concluída com sucesso, obtendo bons resultados, porém deveríamos de aplicar mais métodos auxiliares para alem dos usados para garantir resultados melhores.

IV. Task 3

A. Abordagem

O objetivo da task 3 está relacionado com o aumento da robustez do algoritmo de deteção da matrícula e do reconhecimento de texto. Verificou-se que existe uma enorme sensibilidade dos mesmos, pelo que o maior desafio se prende por alinhar a matrícula o máximo possível para permitir a melhor deteção possível dos caracteres.

B. Pré-processamento

Visto que as imagens recebidas se encontram, na sua maior parte, numa posição em projetiva, o processamento das mesmas é realizado pixel a pixel numa transformação global geométrica projetiva.

Fig. 15: Imagem antes do pré processamento

Primeiramente, é feito um ajuste do tamanho da imagem para 100 por 600 pixels para todas as imagens serem processadas da mesma maneira. Segue-se de igual modo a passagem para gray scale através da função rgb2gray e seguido uma série de pre-processamentos para melhorar a imagem: Aumento do contraste, aumento do sharpenning da imagem e aplicação de um filtro médio de ruído e uma binarização com base no método de Otsu.

Fig. 16: Imagem depois do pré processamento

C. Criação da máscara

Seguidamente, para a remoção do ruído que não interesa da imagem, usou-se a função bwareopen para remover todos os objetos não ligados, inferiores a 50 pixels e depois um close para ligar todos os elementos internos da matrícula. A máscara da matrícula é dada pelo AND lógico entre as duas operações mencionadas acima.

Fig. 17: Máscara obtida da matrícula

Obtida a máscara, é necessário detetar os cantos da mesma para poder aplicar a transformação geométrica.

Segundo a teoria, uma transformação geométrica de perspetiva necessita de 4 pontos:

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Usou-se a função corner para detetar pequenas variações entre pixeis a 1 e a 0 da máscara da matrícula e de seguida organizou-se em dois vetores de dados: Um com a informação das columns onde estão presentes os cantos e outro com a informação das rows. A partir do máximo da linha, máximo da coluna, mínimo da linha e mínimo da coluna é possível traçar os 4 cantos da máscara.

Tendo os 4 pontos (a alterar) da matrícula, falta saber os 4 pontos de destino. Estes correspondem a um retângulo com vértice de referência em (1,1) e com largura c e altura r, em que c é o número de colunas e r o número de linhas da matriz de dados lida a partir da função corner.

Para aplicar a transformação geométrica usou-se a função fitgeotrans que usa 3 parâmetros: 4 pontos a alterar, 4 pontos para onde se quer alterar a imagem e o terceiro parâmetro representa o tipo de projeção, a qual foi escolhida como ”projective”. Este tipo de transformação é usado ainda como parâmetro na função imwarp para concluir o processo de translação de pontos. No final, a imagem corrigida, apresenta este aspeto:

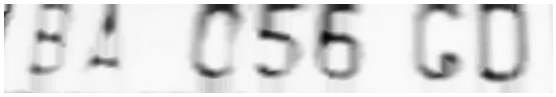

Fig. 18: Imagem final após tratamento

Exemplo número 2:

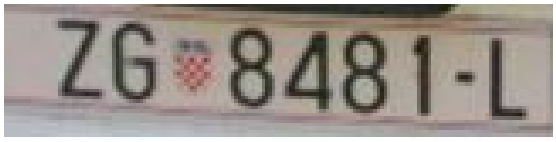

Fig. 19: Imagem original

Fig. 20: Imagem final após tratamento

V. Conclusão

Este trabalho foi concluído com sucesso, obtendo resultados satisfatórios em cada task. Para um caso real era necessário melhorar o pré processamento e introduzir métodos alternativos ao template matching para onde as condições de luz ou ângulo não são favoráveis. Nessa mesma fase de melhoria existe de facto uma dificuldade acrescida devido à grande variabilidade de fotografias (pouca luminosidade, muita luminosidade, matrículas cortadas, mais ou menos inclinadas, etc).

VI. Anexos

Fig. 21: Templates

$$r = \frac{\sum_m \sum_n (A_{mn} - \bar{A})(B_{mn} - \bar{B})}{\sqrt{\left(\sum_m \sum_n (A_{mn} - \bar{A})^2\right)\left(\sum_m \sum_n (B_{mn} - \bar{B})^2\right)}}$$

Fig. 22: Algoritmo de template mathing

References

[1] S. Bangare, A. Dubal, P. Bangare, and S. Patil, “Reviewing otsuâ€™s method for image thresholding,” *International Journal of Applied Engineering Research*, vol. 10, pp. 21 777–21 783, 01 2015.

[2] J. Canny, “A computational approach to edge detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, Nov 1986.

[3] M. S. Aksoy, O. Torkul, and I. H. Cedimoglu, “An industrial visual inspection system that uses inductive learning,” *Journal of Intelligent Manufacturing*, vol. 15, no. 4, pp. 569–574, 2004. [Online]. Available: https://doi.org/10.1023/B:JIMS.0000034120.86709.8c

[4] C. E. Anagnostopoulos, I. E. Anagnostopoulos, I. D. Psoroulas, V. Loumos, and E. Kayafas, “License plate recognition from still images and video sequences: A survey,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 3, pp. 377–391, Sep. 2008.

[5] C. Patel, D. Shah, and A. Patel, “Automatic number plate recognition system (anpr): A survey,” *International Journal of Computer Applications (IJCA)*, vol. 69, pp. 21–33, 05 2013.