A New Method for Automatic Vehicle License Plate Detection

G. L. Corneto, F. A. Silva, D. R. Pereira, L. L. Almeida, A. O. Artero, J. P. Papa,
V. H. C. de Albuquerque and H. M. Sapia

***Abstract*— License plate recognition has been widely studied, and the advance in image capture technology helps enhance or create new methods to achieve this objective. In this work is presented a method for real time detection and segmentation of car license plates based on image analyzing and processing techniques. The results show that the computational cost and accuracy rate considering the proposed approach are acceptable to real time applications, with an execution time under 1 second. The proposed method was validated using two datasets (A and B). It was obtained over 92% detection success for dataset A, 88% in digit segmentation for datasets A and B, and 95% digits classification accuracy rate for dataset B.**

***Keywords*— License plate detection, Digits segmentation, Vehicle License Plate, OCR, Image Processing.**

I. INTRODUÇÃO

SISTEMAS inteligentes aplicados no reconhecimento de placa de licenciamento veicular está em crescente expansão, podendo ser considerado em diferentes situações como, por exemplo, em sistemas de vigilância, rastreamento e identificação de veículos [1]. Segundo Kodwani e Meher [2], os métodos apresentados na literatura são eficientes, mas necessitam de elevado custo financeiro, tornando, na maioria dos casos, inviáveis.

Existem muitas variáveis a serem consideradas ao realizar o reconhecimento de uma placa veicular como, por exemplo, iluminação, sujeiras, desgaste, placas com cores diferentes, dígitos com grafias diferentes, local da placa no veículo, ângulo da câmera, distância da placa em relação à câmera, poluição visual no cenário que contém placa (muitos objetos no cenário podem tornar a detecção da placa difícil), qualidade da resolução da imagem, entre outros [2]. Para resolver esses problemas, diversas técnicas foram criadas para detecção de placas, segmentação e classificação dos dígitos.

O primeiro passo no processo de reconhecimento de placas de licenciamento veicular é o reconhecimento e identificação da placa. Entre as ferramentas computacionais capazes de realizar esta tarefa, destacam-se a morfologia matemática [3][4] e o método de Cascatas de Haar [5]. Após o reconhecimento, é necessário segmentar os dígitos da placa para posterior classificação. Nesta etapa do processamento, diversos métodos são utilizados, tais como transformada de Gabor e quantização vetorial [6], combinações de morfologia matemática [7][8].

Diante do exposto, este trabalho propõe uma metodologia para realizar a detecção de placas de licenciamento veicular, segmentação e classificação dos dígitos contidos nas placas. Para a detecção foi usado o método Cascatas de Haar [9], usando um classificador treinado com o algoritmo de aprendizagem AdaBoost (Adaptive Boosting) [10]. Para segmentação dos dígitos, foi utilizada uma combinação de limiarização [11], projeção vertical e horizontal [12], detecção de contornos [13], e combinações de morfologia matemática. Finalmente, para a classificação dos dígitos das placas, foi utilizada a técnica proposta por Silva et al. [14] [15], em que são analisadas as transições entre os *pixels* da imagem de um dígito segmentado, criando regras sobre essas transições, que utiliza regras criadas para classificar dígitos na imagem. A classe de dígitos com menos regras violadas é escolhida.

II. DESCRIÇÃO DOS MÉTODOS UTILIZADOS

A. Detecção

A detecção é realizada utilizando um classificador robusto previamente treinado. Um classificador robusto refere-se a um conjunto de classificadores fracos, ou seja, classificadores que só precisam acertar sua decisão 51% das vezes, que são combinados entre si para a formação de um novo classificador. Esse método de combinação é chamado *boosting*. Neste trabalho, foi utilizado o algoritmo de *boosting* [10], comumente denominado de AdaBoost (Adaptive Boosting), capaz de realizar uma série de rodadas de treinamento reorganizando os pesos dos classificadores fracos.

B. Classificadores Haar

O classificador utilizado é do tipo Haar (Haar *classifier*) [16], formado por conjuntos de características do tipo Haar (Haar-*like features*), conforme apresentado na Fig. 1.

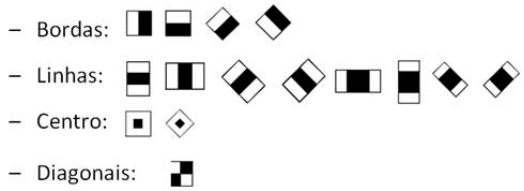


Figura 1. Coleção estendida de características.

G. L. Corneto, Universidade do Oeste Paulista (Unoeste), Presidente Prudente, São Paulo, Brasil, gcorneto@hotmail.com
F. A. Silva, Universidade do Oeste Paulista (Unoeste), Presidente Prudente, São Paulo, Brasil, chico@unoeste.br
D. R. Pereira, Universidade do Oeste Paulista (Unoeste), Presidente Prudente, São Paulo, Brasil, danilopereira@unoeste.br
L. L. Almeida, Universidade do Oeste Paulista (Unoeste), Presidente Prudente, São Paulo, Brasil, llalmeida@uoneste.br
A. O. Artero, Universidade Estadual Paulista (Unesp), Presidente Prudente, São Paulo, Brasil, almir@fct.unesp.br
J. P. Papa, Universidade Estadual Paulista (Unesp), Bauru, São Paulo, Brasil, papa@fc.unesp.br
V. H. C. de Albuquerque, Universidade de Fortaleza (Unifor), Fortaleza, Cerár, Brasil, victor.albuquerque@unifor.br
H. M. Sapia, Universidade do Oeste Paulista (Unoeste), Presidente Prudente, São Paulo, Brasil, helton@unoeste.br

Para obter o valor da imagem referente a uma característica, é utilizada a equação:

$$f(img) = \left(\sum_{i=0}^{h}\sum_{j=0}^{w} ab(i,j)\right) - \left(\sum_{i=0}^{h}\sum_{j=0}^{w} ap(i,j)\right) \quad (1)$$

em que *f*(*img*) representa a imagem, *ab*(*i*, *j*) os *pixels* referentes à região branca e *ap*(*i*, *j*) os *pixels* referentes à região preta. Entretanto, fazer a soma dos valores *pixel* a *pixel* pode demandar elevado tempo de processamento quando são utilizadas imagens de alta resolução. Para resolver tal problema, uma alternativa é utilizar a integral da imagem, ou seja, uma tabela de valores capaz de possibilitar o cálculo de qualquer somatória de um retângulo de *pixels* considerando somente quatro acessos, conforme ilustrado na Fig. 2.

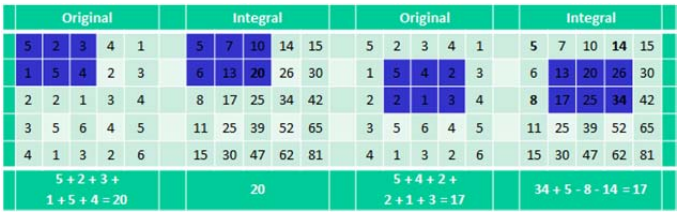


Figura 2. Ilustração do funcionamento da integral de imagem.

C. Cascata de Haar

A estrutura de classificadores em cascata visa aumentar o seu número durante várias iterações de treinamento, desta forma, o classificador treinado não testa todas as características possíveis de uma só vez, mas distribui essas características por vários estágios, ou seja, um classificador possui poucas características em seu primeiro estágio e todas as características extraídas em seu último estágio [10]. Por esta razão, a quantidade de novos estágios de treinamento são responsáveis pelo elevado custo computacional. Cada estágio da cascata pode ser considerado um classificador, e, cada um desses, possuem alto índice de detecção, porém, uma taxa moderada de detecção de falsos positivos (em média 50%). Apesar de demorada, a estrutura em cascata melhora a eficiência da detecção, uma vez que um objeto que não passa por um dos estágios de classificação é imediatamente descartado.

D. AdaBoost

O algoritmo de treinamento do AdaBoost foi apresentado por Freund e Schapire [10], considerando um conjunto de treinamento de tamanho *n*, contendo duas classes de objetos a serem classificadas, o algoritmo inicia criando uma distribuição de pesos D_k, conforme equação:

$$D_k = \frac{1}{n}. \quad (2)$$

Após a distribuição, o algoritmo realiza a classificação do conjunto e considera a quantidade de acertos e erros para gerar o erro de classificação, e então determina a importância γ associada ao erro segundo as seguintes equações:

$$e_i = \sum De \quad (3)$$

$$\gamma_i = \frac{1}{2} ln\left(\frac{1-e_i}{e_i}\right), \quad (4)$$

em que *i* indica a iteração de treinamento atual. Com o valor de γ definido, a distribuição de pesos é reavaliada segundo a equação:

$$nD_k = \begin{cases} D_k \times e^{-\gamma_i} \\ D_k \times e^{\gamma_i} \end{cases}, \quad (5)$$

em que $-\gamma$ representa os elementos classificados corretamente, γ os elementos classificados erroneamente e nD_k os novos valores de peso. Após a reavaliação, os pesos são normalizados a partir de:

$$Z_i = \sum nD_k \quad (6)$$

$$nD_k = \frac{nD_k}{Z_i}, \quad (7)$$

em que Z_i é o termo normalizador. Com os novos pesos definidos e normalizados, o algoritmo segue para a próxima iteração de distribuição e reavaliação de pesos até que o erro mínimo estipulado ou o número máximo de iterações (o que acontecer primeiro) seja atingido. Cada γ gerado nas iterações do treinamento define um classificador fraco, e o conjunto de todos eles define o classificador robusto/forte.

III. MÉTODO PROPOSTO

Nesta seção é descrito o funcionamento da ferramenta proposta, sendo divida em detecção, segmentação, e reconhecimento/classificação, conforme apresentado no fluxograma das etapas, Fig. 3.

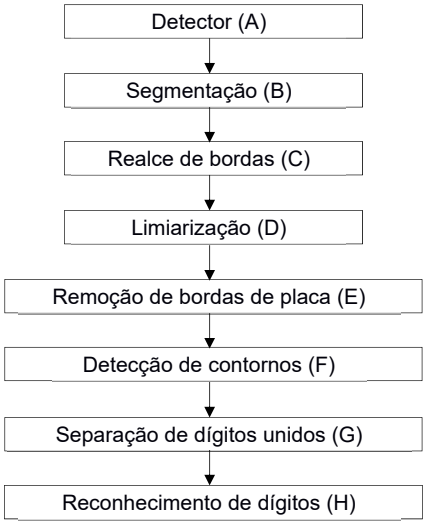


Figura 3. Etapas do método proposto.

A. Detector

Após a fase de treinamento, aplica-se o classificador forte gerado para detectar o objeto de interesse. O detector implementado no OpenCV define um retângulo com tamanho mínimo *h* x *w* e tamanho máximo *H* x *W*. Utilizando a área do retângulo como parâmetro, é realizada uma varredura completa da imagem, e, em cada etapa da varredura, as características Haar usadas no treinamento são consideradas para classificar a região de interesse. Se esta região for classificada como correta em todas os estágios do classificador, ela é definida como uma possível área de interesse. Se o detector encontrar, no mínimo, dois retângulos

de interesse que se sobrepõem (número definido como parâmetro do método), a região é considerada como escolhida. Finalmente, todas as placas são redimensionadas utilizando a escala S, definida como:

$$S = 350/W, \quad (8)$$

na qual W representa a largura da imagem da placa. Dessa forma, todas as placas possuem 350 *pixels* de largura e altura proporcional a esse redimensionamento. Esta padronização é importante, uma vez que a etapa seguinte, segmentação, envolve contagem de *pixels*, podendo interferir em resultados pouco satisfatórios.

B. Segmentação

Com a região da placa definida, é necessário segmentar os dígitos para serem reconhecidos. Os métodos de realce de bordas, limiarização e detecção de contornos utilizados são os implementados na biblioteca OpenCV.

C. Realce de bordas

O primeiro passo da segmentação é realçar as bordas da placa. Esse realce é realizado aplicando o filtro Gaussiano na imagem, usando uma semente (matriz usada para percorrer a imagem) de 9x9 *pixels*, para criar um efeito de "borramento" na imagem, diminuindo a nitidez das bordas. Em seguida, é realizada uma soma ponderada entre a imagem com o filtro Gaussiano e a imagem original, com os pesos definidos em 2 para a imagem original, e -1 para a imagem com o filtro. Essa soma é baseada em:

$$Ir = 2 \cdot Io + (-1 \cdot Ig), \quad (9)$$

em que Io representa a imagem original, Ig a imagem com filtro gaussiano, e Ir a imagem resultante. O objetivo é definir os detalhes da borda. Após o realce das bordas, os resultados obtidos na limiarização vão ter mais destaque nos dígitos da placa, e menor preservação de ruídos.

D. Limiarização

Esta fase trata da definição de um ou mais limiares entre os tons de cinza da imagem. Para este trabalho, foi utilizado o limiar obtido de modo automático pelo método de Otsu [11]. Após a limiarização, é aplicada a operação morfológica de fechamento utilizando elemento estruturante circular de tamanho 7x7 para remover ruídos na imagem, conforme Fig. 4.

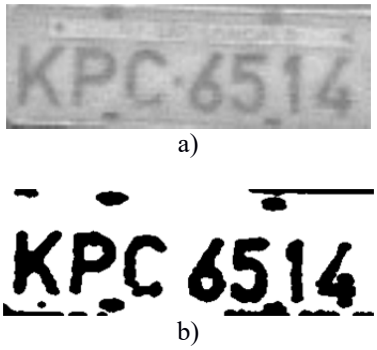


Figura 4. Placa veicular: a) detecção e b) após remoção de ruídos.

E. Remoção de bordas de placa

Na maioria dos casos, as bordas ficam aparentes na imagem limiarizada, sendo necessário ser removidas para não interferir na detecção de contornos externos. Este processo é realizado a partir das projeções vertical e horizontal da imagem, Fig. 5.

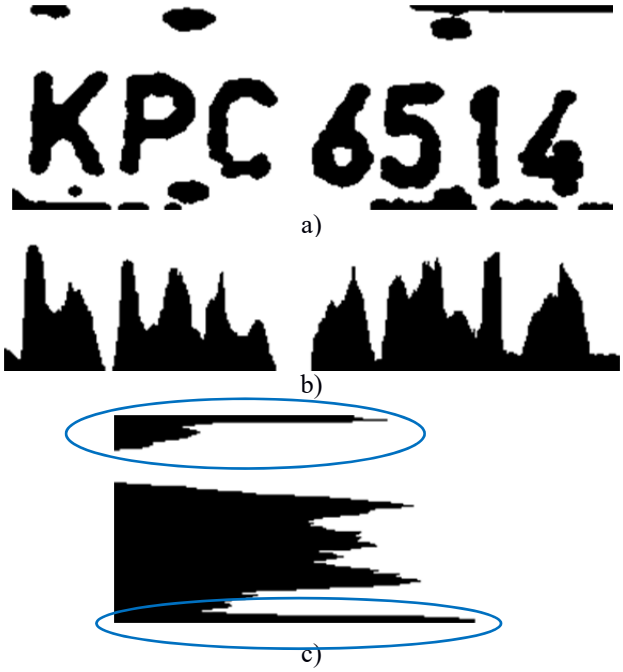


Figura 5. Placa a) limiarizada, b) projeção vertical e c) projeção horizontal. A área destacada representa a região de corte para remoção das bordas.

As projeções indicam a quantidade de *pixels* pretos em cada linha ou coluna da imagem. Essa informação é usada na remoção das bordas, na qual existe uma grande concentração de *pixels*, seja na forma horizontal ou vertical, portanto, se uma linha da projeção possui mais da metade da largura total da imagem, essa linha é removida.

O mesmo raciocínio é aplicado para bordas verticais, porém, uma coluna da projeção precisa ter mais de 90% da altura da imagem, para que dígitos não sejam confundidos com bordas.

A projeção horizontal pode ser usada para remover áreas "vazias" na placa, com o objetivo de separar ao máximo somente os dígitos. Este processo é realizado eliminando as linhas que possuam menos de um terço da largura da imagem, conforme ilustrado na Fig. 6.

F. Detecção de contornos

Para detecção de contornos, foi considerado o método proposto por Suzuki e Abe [13], que é baseado no algoritmo de *border-following*, que cria uma estrutura hierárquica entre os contornos encontrados, definindo os mais externos de uma imagem. Estes contornos são analisados, uma vez que ainda podem existir ruídos que poderão ser detectados como contornos.

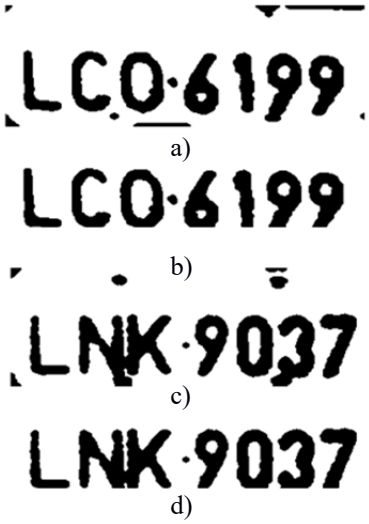


Figura 6. Ilustração das remoções de bordas.

Nesta etapa, são estabelecidos retângulos capaz de comportar cada contorno e excluir os retângulos que não possuírem altura e largura suficientes ou possuírem altura excessiva (largura excessiva não é considerada, pois pode ser um sinal de dígitos unidos). Na Fig. 7 é ilustrado o resultado da detecção de contornos, bem como após a remoção de contornos indesejados.

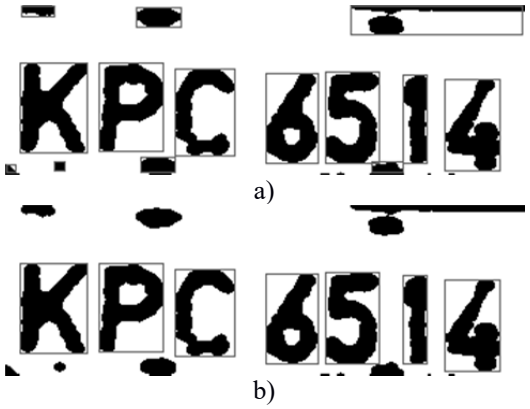


Figura 7. Eliminação de contornos desejados.

G. Separação de dígitos unidos

Esta fase é considerada somente quando é necessário separar dígitos unidos por conta de distorções na imagem da placa. Todo dígito com mais 60 *pixels* de largura é tratado como dígitos unidos.

Para separar dígitos unidos, é realizada uma projeção vertical da área de interesse e, em seguida, é calculado o ponto de mínimo global dessa projeção. Então, o retângulo é divido na coluna referente ao ponto mínimo e as imagens são reanalisadas, quando mais de dois dígitos estarem unidos.

Ao fim de todos os passos, a imagem utilizada na segmentação está distorcida e o reconhecimento pode ser prejudicado. Por isso, é realizada mais uma limiarização na imagem original, utilizando a área dos retângulos definidos no final da segmentação. A limiarização é aplicada em cada retângulo individualmente, dessa forma, o valor de limiar estabelecido não é influenciado por ruídos remanescentes na placa.

H. Reconhecimento de dígitos

O processo de reconhecimento só é iniciado caso o método de segmentação retornar exatamente sete dígitos, caso contrário, o erro é automaticamente assumido. O método utilizado foi implementado por Silva et al. [14]. Esse algoritmo percorre a imagem de um dígito conhecido, registrando as transições entre os valores 0 e 1 dos *pixels*. A imagem é analisada conforme ilustração apresentada na Fig. 8.

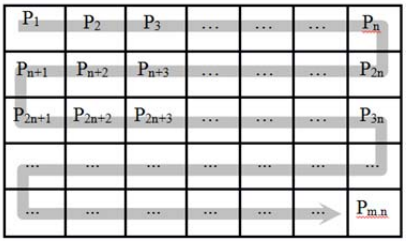
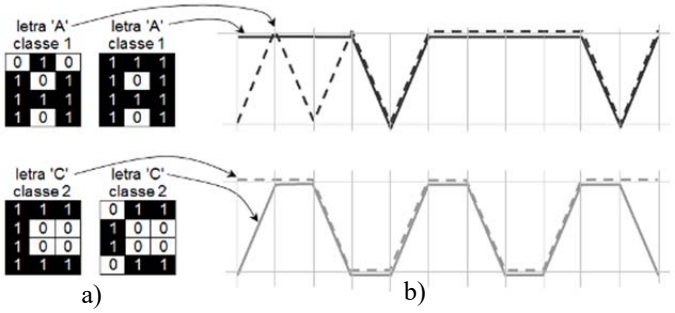


Figura 8. Caminho de varredura dos *pixels* pelo algoritmo de reconhecimento.

Para cada classe de dígito, podem ser registradas mais de uma lista de transição como, por exemplo, dígitos de fontes diferentes. Uma vez que todas as transições foram registradas, algumas regras são estabelecidas para realizar o reconhecimento de outros dígitos. É gerada uma lista que contém todas não-ocorrências de transições para cada classe de dígito. Em imagens preto e branco, as transições possíveis são 00, 01, 10 e 11. Na Fig. 9 a) são representadas as transições de *pixel* de duas classes de letra 'A' e duas classes de letra 'C'. Na Fig. 9 b) um gráfico de coordenadas paralelas ilustra as transições das diferentes classes em a), e, finalmente, Fig. 9 c), as não-ocorrências de transições nas classes.



	Transições que não ocorrem entre pixels adjacentes										
	a1-a2	a2-a3	a3-a4	a4-a5	a5-a6	a6-a7	a7-a8	a8-a9	a9-a10	a10-a11	a11-a12
classe 1	00 10	00 01	00 10	00 01 11	00 10 11	00 01 10	00 01 10	00 01 10	00 01 10	00 01 11	00 10 11
classe 2	00 10	00 01 10	00 01 11	01 10 11	00 10 11	00 01 10	00 01 11	01 10 11	00 10 11	00 01 10	00 01

c)

Figura 9. Transições de *pixels*.

Com as não-ocorrências registradas, o reconhecimento é realizado criando uma lista de transições a ser reconhecido e comparando com as não-ocorrências das classes criadas. A partir disso, três regras são estipuladas: (i) reconhecer o dígito da classe que não for violada pelo dígito a ser reconhecido, (ii) caso mais de uma classe se encaixar na primeira regra, a classe que for mais restritiva entre elas é escolhida, e (iii) se as

transições não se encaixarem nas restrições de nenhuma classe, a classe menos violada é escolhida.

IV. EXPERIMENTOS

Os experimentos realizados foram executados usando um laptop com processador i5 M430 2.27GHz com 4 GB de memória RAM.

A. Treinamento dos classificadores

Foram considerados dois treinamentos em bases de imagens diferentes. O primeiro treinamento foi realizado com a Base "A" do Laboratório de Processamento Digital de Sinais e Imagens (LPDSI) [17], utilizando 470 imagens (apenas placas cinza não reflexivas) com variação de intensidade (HSI) entre +20 e -20, totalizando 1410 imagens. Entretanto, em 130 casos, a imagem da placa ficou escura demais ou clara demais, e essas imagens foram removidas. No segundo treinamento, foi realizada uma combinação de imagens da Base "A" e da Base "B" (criado pelos autores a partir de imagens obtidas de quadros de um vídeo de 30 minutos gravado em uma avenida bem movimentada na cidade de Presidente Prudente – SP). Os treinamentos duraram aproximadamente 16 e 18 dias para o primeiro e segundo, respectivamente. Todo o processo de treinamento foi realizado com programas disponibilizados pela biblioteca de Visão Computacional OpenCV.

B. Resultados

O método proposto arquivou resultados bastante satisfatórios de detecção e segmentação, sendo a maioria das falhas oriundas de distorções na imagem, mostrando que o método é viável e eficiente para imagens nítidas. Na Tabela 1 são apresentados os resultados obtidos com os dois classificadores treinados, sendo o primeiro utilizando somente imagens da Base "A" e o segundo utilizando a combinação de imagens da Base "A" e da Base "B". O primeiro classificador apresentou resultados superiores em relação ao segundo no que diz respeito à detecção das placas. Isto pode ter ocorrido pelo fato de que o segundo classificador utilizou imagens de duas bases diferentes (imagens da Base "A" com pouca resolução, mas com certa padronização e imagens da Base "B" com uma melhor resolução), o que prejudicou o resultado final.

TABELA I
RESULTADOS PARA CADA UM DOS MÓDULOS DESENVOLVIDOS E O TEMPO DE EXECUÇÃO DE CADA UM

	Primeiro Classificador Base "A"		**Segundo Classificador Base "A" combinada com a Base "B"**	
	Acertos [%]	**Tempo [s]**	**Acertos [%]**	**Tempo [s]**
Detecção da placa	92,55	0,09	74,33	0,12
Segmentação dos dígitos	88,86	0,01	88,71	0,01
Classificação dos dígitos	88,40	Não se aplica	95,57	Não se aplica

De modo a realizar uma comparação com outros trabalhos encontrados na literatura, foi encontrado o trabalho de Abreu et al. [18] que faz uso da Base "A" [17], cuja metodologia combina operadores morfológicos e busca por *template*. Os resultados da metodologia proposta neste trabalho demonstram ser promissores, haja visto, que foram superiores na detecção da placa e na classificação dos dígitos se comparado ao trabalho referido, sob as mesmas condições. Os resultados de detecção e classificação dos dígitos no trabalho de Abreu et al. [18] foram respectivamente 89,00% e 88,00%, enquanto no presente trabalho, os resultados foram 92,55% e 88,40%. A análise relativa ao tempo de processamento não foram analisadas, devido ao fato de que o trabalho comparado desconsidera essa informação, realçando apenas a acurácia.

C. Falhas

Na Figura 10 são mostradas algumas falhas de detecção. Fig 10 a), o fundo da placa prejudicou a extração de características, falhas de segmentação, em b) e c), ambos prejudicados pelo excesso de ruídos na imagem e falhas de reconhecimento, em d) e e), ilustrando a recorrência de erros. Os valores presentes no canto inferior esquerdo das imagens d) e e) representam o tempo de execução do método.

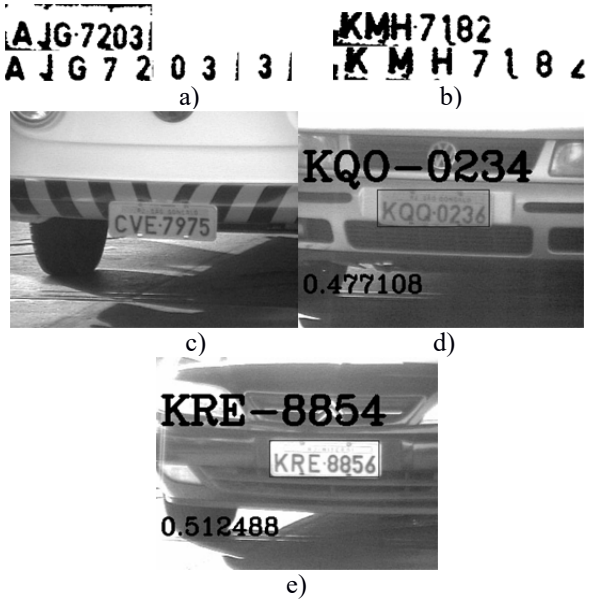


Figura 10. Exemplos de falhas ocorridas.

V. CONCLUSÕES

Os resultados apresentados com a utilização da metodologia desenvolvida neste trabalho mostram que esta pode ser aplicada em situações que necessitam de um tempo de resposta rápido, isto porque os tempos de processamento foram abaixo de 0.5 segundos, com taxa de acerto de 92,55% na detecção de placas e 88,40 na classificação de dígitos, mostrando ser uma ferramenta promissora para análise em tempo real.

REFERÊNCIAS

[1] S.G. Patel, "Vehicle License Plate Recognition Using Morphology and Neural Network", International Journal on Cybernetics & Informatics (IJCI). Gujarat, Índia, Vol. 2, No. 1, Fev. 2013, pp. 1-7.

[2] L. Kodwani, S. Meher, "Automatic License Plate Recognition in Real Time Videos using Visual Surveillance Techniques", ITSI Transactions on Electrical and Electronics Engineering (ITSI-TEEE), Raipur, Índia, Vol. 1, No. 6, Abr. 2013, pp. 60-66.

[3] F.M. Rodríguez, X. F. Hermida, "New Advances in Automatic Reading of VLP's (Vehicle License Plates)", Proc. SPC-2000 (Signal Processing and Communications), Espanha, Marbella, Set. 2000.

[4] B. Hongliang, L. Changping, "A hybrid License Plate Extraction Method Based On Edge Statistics and Morphology", Proc. of the 17th International Conference on Pattern Recognition (ICPR 2004). Vol. 2, Ago. 2004, pp. 831-834.

[5] L. Dlagnekov, "License Plate Detection Using AdaBoost". http://cseweb.ucsd.edu/classes/fa04/cse252c/projects/louka.pdf.

[6] F. Kahraman, B. Kurt, M. Gökmen, "License Plate Character Segmentation Based on the Gabor Transform and Vector Quantization", Computer and Information Sciences-ISCIS 2003. Alemanha, Berlim, 2003, pp. 381-388.

[7] D. Wazalwar, O. Erdal, J. Saniie, "A Design Flow for Robust License Plate Localization and Recognition in Complex Scenes", Journal of Transportation Technologies, Chicago, Vol. 2, No. 1, Jan. 2012, pp. 13-21.

[8] S. Nomura, K. Yamanaka, O. Katai, H. Kawakami, T. Shiose. A novel adaptive morphological approach for degraded character image segmentation. Pattern Recognition. v. 38, n. 11, p. 1961-1975, Nov. 2005.

[9] P. Viola, M. Jones, "Rapid object detection using a boosted cascade of simple features", Proc. of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001, Cambridge, Vol. 2, No. 1, Jan. 2012, pp. 13-21.

[10] Y. Freund, R.E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting", Journal of computer and system sciences. Murray Hill, EUA. Vol. 55, No. 1, Set. 1995, pp. 119-139.

[11] N. Otsu, "A Treshold Selection Method From Gray-Level Histograms", IEEE Transactions on Systems, Man, and Cybernetics, Vol. 9, No.1, Jan. 1979, pp. 62-66.

[12] Y. Lu, "Machine printed character segmentation—; An overview", Pattern Recognition, Vol. 28, No. 1, Jan. 1995, pp. 67-80.

[13] S. Suzuki, K. Abe, "Topological Structural Analysis of Digitized Binary Images by Border Following", Computer Vision, Graphics, and Image Processing. Vol. 30, No. 1, 1985, pp. 32-46.

[14] F.A. Silva, A.O. Artero, M.S.V. Paiva, R.L. Barbosa. "Um Algoritmo Rápido para o Reconhecimento de Caracteres", VII Workshop de Visão Computacional – WVC 2011. Curitiba, Paraná, 2011, pp. 149-154.

[15] F.A. Silva, A.O. Artero, M.S.V. Paiva, R.L. Barbosa, "ALPRs – A New Approach for License Plate Recogniton using the SIFT Algorithm", Signal & Image Processing: An International Journal (SIPIJ), Vol. 4, No. 1, Fev. 2013, pp. 17-33.

[16] R. Lienhart, J. Maydt, "An extended set of Haar-like features for rapid object detection", Proc. of International Conference on Image Processing, Vol. 1, 2002, pp. 900-903.

[17] LPDSI Laboratório de Processamento Digital de Sinais e Imagens. http://www.cbpf.br/cat/lpdsi/lpr/index.html.

[18] D.S. Abreu, W.A.L. Alves, S.A. Araújo, A.F.H. Librantz, "Uma Abordagem para a Localização e o Reconhecimento de Placas de Licenciamento Veicular por meio de Operadores Morfológicos e Busca por Template", VII Workshop de Visão Computacional – WVC 2011. Curitiba, Paraná, 2011, pp. 79-84.

Guilherme Lofrano Corneto graduado em Ciência da Computação na Universidade do Oeste Paulista (Unoeste), Brasil (2015).

Francisco Assis da Silva graduado em Ciência da Computação na Universidade do Oeste Paulista (Unoeste), Brasil (1998), Mestre em Ciência da Computação na Universidade Federal do Rio Grande do Sul (UFRGS), Brasil (2002) e Doutor em Ciências, Programa de Engenharia Elétrica da Universidade de São Paulo (USP), Brasil (2012). Atualmente é pós-doutorando na Universidade Estadual Paulista (Unesp) e professor na Universidade do Oeste Paulista (Unoeste), Brasil.

Danillo Roberto Pereira graduado em Ciência da Computação na Universidade Estadual de São Paulo (Unesp), SP, Brasil em 2006, Mestre em Ciência da Computação na Universidade de Campinas (Unicamp), Brasil (2009), Doutor em Ciência da Computação (Unicamp), Brasil (2013) e Pós-Doutor na Universidade Estadual de São Paulo em 2016. Atualmente é pós-doutorando na Universidade Federal de São Carlos (UFSCar) e professor na Universidade do Oeste Paulista (Unoeste), Brasil.

Leandro Luiz de Almeida graduado em Ciência da Computação na Universidade do Oeste Paulista (Unoeste), Brasil (1997), Mestre em Ciências Cartográficas na Universidade Estadual de São Paulo (Unesp), Brasil (2001) e Doutor em Ciências, Programa de Engenharia Elétrica da Universidade de São Paulo (USP), Brasil (2013). Atualmente é professor na Universidade do Oeste Paulista (Unoeste), Brasil.

Almir Olivette Artero graduado em Matemática na Universidade Estadual de São Paulo (Unesp), Brasil (1990), Mestre em Ciências Cartográficas na Universidade Estadual de São Paulo (Unesp), Brasil (1999) e Doutor em Ciência da Computação na Universidade de São Paulo (USP) (2005). Atualmente é professor na Universidade do Estado de São Paulo (Unesp), Brasil.

João Paulo Papa graduado em Sistemas de Informação na Universidade Estadual de São Paulo (Unesp), Mestre em Ciência da Computação na Universidade Federal de São Carlos (UFSCAr), Brasil (2005), Doutor em Ciência da Computação na Universidade de Campinas (Unicamp), Brasil (2008), Pós-Doutor na Universidade de Campinas (Unicamp), Brasil, 2009 e na Universidade de Harvard em 2015. É professor no Departamento de Computação na Universidade Estadual de São Paulo (Unesp), desde 2009.

Victor Hugo C. de Albuquerque graduado em Tecnologia Mecatrônica no Centro Tecnológico Federal de Educação do Ceará, Brasil (2006), Mestre em Engenharia Teleinformática na Universidade Federal do Ceará, Brasil (2007) e Doutor em Engenharia Mecânica com ênfase em Materiais na Universidade Federal da Paraíba, Brasil (2010). Atualmente é professor do Programa de Graduação em Informática da Universidade de Fortaleza (Unifor).

Helton Molina Sapia graduado em Ciência da Computação na Universidade do Oeste Paulista (Unoeste), Brasil (1995), Mestre em Ciência da Computação na Universidade Estadual de São Paulo (Unesp), SP, Brasil em 2016. Atualmente é professor na Universidade do Oeste Paulista (Unoeste), Brasil e na Universidade Estadual de São Paulo (Unesp), Brasil.