Capítulo 14

Reconhecimento de Caracteres Baseado em Regras de Transições entre *Pixels* Vizinhos

Francisco Assis da Silva*, Almir Olivette Artero,
Maria Stela Veludo de Paiva e Ricardo Luís Barbosa

Resumo: Este capítulo trata do reconhecimento de caracteres impressos e manuscritos, apresentando um algoritmo totalmente baseado na análise do comportamento das transições entre os *pixels* vizinhos nas imagens dos caracteres. A partir desta análise, são definidas regras que determinam em qual classe cada caractere deve ser colocado, caracterizando uma classificação supervisionada. A baixa complexidade deste algoritmo tem tornado possível o seu uso em aplicações onde o tempo de reconhecimento é bastante crítico, como é o caso de sistemas de reconhecimento em tempo real, usados em sistemas de visão computacional, como robôs e veículos não tripulados.

Palavras-chave: Reconhecimento de caracteres, Classificação supervisionada, Análise de transições entre *pixels*, Processamento de vídeo.

Abstract: *This chapter deals with the recognition of printed and handwritten characters, and presents an algorithm based exclusively on the analysis of the behavior of transitions between neighboring pixels in the images of the characters. From this analysis, rules are defined to determine in which class each character should be placed, corresponding to a supervised classification scheme. The low complexity of this algorithm has made its use possible in applications where the recognition time is quite critical, such as real-time recognition systems used in computer vision systems for robots and autonomous vehicles.*

Keywords: *Character recognition, Supervised classification, Pixel transition analysis, Video processing.*

*Autor para contato: chico@unoeste.br

1. Introdução

O problema do reconhecimento de caracteres têm chamado atenção há bastante tempo, com os trabalhos pioneiros de Tauschek, que obteve a patente do OCR (*Optical Character Recognition*) na Alemanha em 1929 e, posteriormente, nos Estados Unidos (Tauschek, 1935), e de Handel (1933), que também registrou uma patente nos Estados Unidos. A partir da década de 50, com o impulso gerado pelos computadores (Dimond, 1957; Neisser & Weene, 1960; Eden, 1961; Eden & Halle, 1961; Frishkopf & Harmon, 1961), a área se tornou ainda mais atrativa, surgindo uma grande variedade de propostas de algoritmos para resolver este problema. Atualmente, o reconhecimento de caracteres continua sendo uma área de intensa pesquisa, que ainda apresenta vários problemas, por causa da grande diversidade de formas que os caracteres podem assumir, principalmente, no caso de caracteres manuscritos (ICR, *Intelligent Character Recognition*) (Gonzalez & Woods, 2001; Montaña, 2007; Jain & Ko, 2008; Pereira et al., 2010; Shrivastava & Gharde, 2010; Trentini et al., 2010; Lin et al., 2011). Mesmo no caso do reconhecimento de caracteres impressos (OCR), que é uma tarefa mais simples, ainda persistem diversas dificuldades, por causa da grande diversidade de fontes que podem ser usadas nos documentos.

Apesar do grande número de propostas apresentadas para o reconhecimento dos caracteres, existem duas tarefas básicas nesta área, que são a abordagem estrutural (Pavlidis, 1980; Schalkoff, 1992) e a abordagem estatística (Schalkoff, 1992; Duda et al., 2001; Jain et al., 2000), que inclui a extração de atributos e a classificação dos objetos a partir de informações obtidas a partir de objetos conhecidos (Zhu et al., 2000). Entretanto, para realizar uma classificação satisfatória dos caracteres, é preciso usar uma grande quantidade de atributos que, muitas vezes, compromete a execução da tarefa. Na segunda abordagem, o reconhecimento dos caracteres pode ser feito através de uma classificação supervisionada, quando se busca obter informações que permitam predizer, com precisão, a classe de cada amostra a partir de medições realizadas em caracteres cujas classes são conhecidas. Uma alternativa a este processo é a classificação não supervisionada, também chamada de agrupamento e, neste caso, procura-se, simplesmente colocar os caracteres em classes, onde eles apresentam grande similaridade entre seus integrantes e baixa similaridade entre os elementos de classes distintas.

Além de realizar a classificação correta dos caracteres, um dos principais desafios da área é o tempo de processamento, que precisa ser muito baixo, para que a tarefa possa ser feita em tempo aceitável. Assim, neste trabalho propõe-se uma estratégia simples e rápida para modelar o comportamento dos caracteres, usando apenas as transições que ocorrem entre os níveis de *pixels* adjacentes que formam os caracteres. Por causa de sua baixa complexidade, o algoritmo consegue excelente tempo de processamento, o

que permite a sua aplicação em tarefas consideradas de tempo real, como é o caso do reconhecimento de placas em rodovias, durante o deslocamento do veículo. As demais seções deste capítulo estão organizadas da seguinte maneira: na Seção 2 são apresentados alguns trabalhos relacionados ao reconhecimento de caracteres; na Seção 3 é apresentada a estratégia proposta neste trabalho para modelar o comportamento dos caracteres cujas classes são conhecidas (treinamento) e, então, obter as transições permitidas em cada classe e usar esta informação na classificação dos caracteres; a Seção 4 apresenta alguns experimentos realizados com esta proposta em um conjunto real de dados, bem como os resultados obtidos e a análise de desempenho; por fim, na Seção 5 são apresentados os comentários finais e trabalhos futuros.

2. Trabalhos Relacionados

Embora o problema do reconhecimento de caracteres tenha atraído atenção desde os primórdios da computação, com a atual tendência de uso dos *tablets*, o reconhecimento de caracteres manuscritos continua sendo uma área de grande interesse pelos fabricantes destes dispositivos, pois a inserção eficaz de textos em dispositivos sem o tradicional teclado QWERTY é um importante diferencial entre os aparelhos. Alguns trabalhos recentes nesta área são descritos a seguir.

O trabalho de Montaña (2007) realiza o reconhecimento de padrões de dígitos empregando redes neurais. Em seu trabalho são usadas duas redes neurais diferentes para alcançar os resultados, uma rede Perceptron Multicamadas e uma rede baseada no Mapa de Kohonen.

O trabalho de Jain & Ko (2008) apresenta um algoritmo de classificação para reconhecer dígitos numéricos manuscritos (0-9), sendo utilizada a implementação da Análise de Componentes Principais (*Principal Component Analysis* – PCA), combinada com o algoritmo do primeiro vizinho mais próximo (1-*nearest neighbor*) para reconhecer os dígitos.

A contribuição do trabalho de Pereira et al. (2010) é melhorar a precisão do reconhecimento de caracteres manuscritos usando um novo método de extração de características, também baseado em Análise de Componentes Principais. Neste caso, aplica-se uma nova técnica que visa combinar os melhores aspectos de uma Análise de Componentes Principais Modular (MPCA) e uma Análise de Componentes Principais de Imagem (IMPCA).

No trabalho de Trentini et al. (2010) a partir da imagem segmentada de uma placa de automóvel, é realizada uma varredura em cada coluna da imagem e são contadas as quantidades de *pixels* pretos, representando a densidade correspondente a cada coluna. Para a segmentação dos caracteres, ou seja, separar os caracteres em relação ao fundo da placa é utilizada uma função de análise de máximos e mínimos locais. Para o reconhecimento dos caracteres é utilizado o algoritmo *Random Trees*, também

chamado de *Random Forests*, o qual é um classificador baseado em árvores de decisão e pode reconhecer os padrões de várias classes ao mesmo tempo.

O trabalho de Shrivastava & Gharde (2010) é utilizado para reconhecimento de números *Devanagari* manuscritos. *Devanagari* é um alfabeto manuscrito usado por vários idiomas na Índia. Para a realização do trabalho os autores utilizaram a técnica de aprendizagem de máquina *Support Vector Machines* (SVM).

3. Modelagem do Comportamento das Sequências de *Pixels*

A estratégia proposta neste trabalho sugere descrever os caracteres, enquadrando-os em uma malha com dimensões definidas previamente e, em seguida, observar as transições entre os níveis de cinza (0 e 1 – imagens binárias) dos *pixels* adjacentes. Deste modo, uma imagem com dimensões $m \times n$ gera um conjunto com $m.n$ atributos ($m.n - 1$ transições). Os atributos são definidos percorrendo os *pixels* na sequência indicada na Figura 1 (embora outras sequências possam ser experimentadas). Em seguida, busca-se determinar o comportamento das sequências dos *pixels* em cada classe e, então, este conhecimento pode ser usado para classificar as poligonais (coordenadas paralelas (Inselberg, 1985) – Figura 2) de outros registros, para os quais não se conhece a classe.

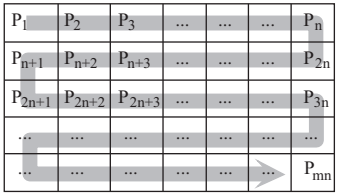
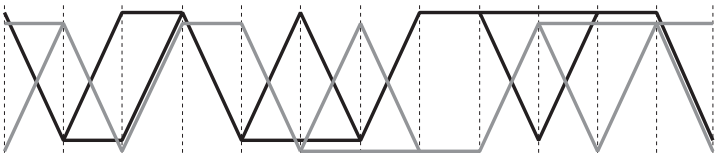


Figura 1. Sequência de *pixels* na imagem usada.

Nesta estratégia, é construída uma lista com as transições permitidas para cada caractere, anotando as transições que ocorrem entre os atributos adjacentes em cada uma das classes ou, então, as transições que não ocorrem em cada classe. Como as imagens usadas são binárias, as transições possíveis entre dois *pixels* que formam um caractere são: 00, 01, 10 e 11. A Figura 2 apresenta um exemplo em que os comportamentos das poligonais de quatro caracteres em duas classes (dois em cada classe) são comparados. Em (a) tem-se a visualização em coordenadas paralelas do conjunto de dados apresentado em (b), destacando as transições. Em (c) observa-se que, entre os atributos a_1 e a_2, os caracteres da classe 1 possuem apenas a transição 10, não ocorrendo as transições 01, 11 e 00. Quanto à classe 2, em (d), nota-se que não ocorrem as transições 10 e 00, para esses atributos. Assim, para cada classe são determinadas todas as transições que

(a)



(b)

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}
classe 1	1	0	0	1	0	1	0	1	1	1	1	1	0
classe 1	1	0	1	1	0	0	0	1	1	0	1	1	0
classe 2	0	1	0	1	1	0	1	0	0	1	1	1	0
classe 2	1	1	0	1	1	0	0	0	0	1	0	1	1

(c)

(d)

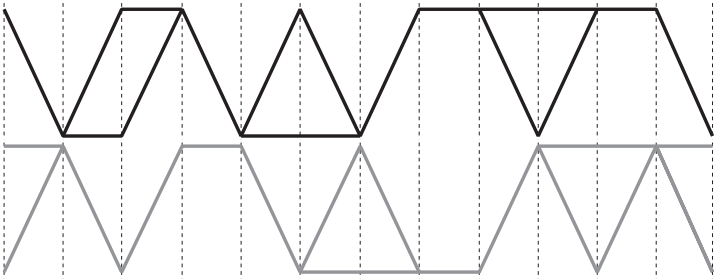


Figura 2. a) Exibição dos registros da classe 1 em preto e da classe 2 em cinza; b) conjunto de dados contendo 4 registros com treze atributos em duas classes; c) Exibição dos registros da classe 1; d) classe 2.

não ocorrem. Em seguida, a partir desta informação, os caracteres a serem reconhecidos são incluídos na classe que apresenta o menor número de inconsistências em relação às transições características anotadas em cada classe.

Em seguida, a classificação dos caracteres pode então ser conduzida usando as três regras de classificação propostas:

R_1**:** Reconhecer o caractere na classe cujas transições não são violadas pelas transições do registro;

R_2**:** Havendo mais de uma classe que satisfaz esta condição, a classe que possuir mais restrições (classe mais restritiva) deverá ser a escolhida;

R_3**:** Quando o registro não atende todas as restrições de nenhuma classe, deverá ser inserido na classe menos violada, ou ser classificado como ruído em casos extremos.

A última regra (R_3) prevê um limiar definido pelo usuário para determinar quando o registro deve ser classificado como ruído. Por exemplo, classificando-o como ruído quando ele não atende ao menos 30% das restrições de regra alguma.

As restrições das duas classes do conjunto de dados ilustrado na Figura 2 (b) são apresentadas na Tabela 1. No caso, as duas classes possuem

Tabela 1. Transições que não ocorrem entre atributos adjacentes no conjunto de dados ilustrado na Figura 2 (b).

	Atributos adjacentes											
	a_1-a_2	a_2-a_3	a_3-a_4	a_4-a_5	a_5-a_6	a_6-a_7	a_7-a_8	a_8-a_9	a_9-a_{10}	a_{10}-a_{11}	a_{11}-a_{12}	a_{12}-a_{13}
Classe 1	00	10	00	00	10	01	00	00	00	00	00	00
	01	11	10	01	11	11	10	01	01	10	01	01
	11			11			11	10			10	11
Classe 2	00	00	00	00	00	10	01	01	00	00	00	00
	10	01	10	01	01	11	11	10	10	01	10	01
		11	11	10	11			11	11			

quantidades iguais de restrições, ou seja, não ocorrem as trinta transições na classe 1 e as trinta transições na classe 2, indicadas na Tabela 1. Embora esta abordagem tenha sido proposta para operar com dados binários (com cardinalidade igual a dois), dados de maior cardinalidade também podem ser processados de duas formas diferentes. Na primeira os valores no conjunto de dados devem ser convertidos para binário, o que conduz a um aumento no número de atributos, melhorando o processo, pois aumenta o número de transições. A segunda possibilidade consiste em discretizar os valores dos atributos em número finito de níveis e então definir as transições para todos os níveis. Assim, dados dois atributos a_i e a_j, com cardinalidade c_i e c_j, respectivamente, o número de transições entre eles é dado pelo produto $c_i.c_j$.

O número total de transições T para um conjunto de dados contendo n atributos é dado pela soma de todas as transições entre os atributos adjacentes c_i e c_{i+1} indicada pela Equação 1.

$$T = \sum_{i=1}^{n-1} c_i \cdot c_{i+1} \tag{1}$$

3.1 Alternativas para melhorias no processo

Duas alternativas que podem ser usadas para melhorar a qualidade das classificações usando esta estratégia são: 1) aumentar as dimensões da malha usada para representar os caracteres; 2) Anotar as transições entre três ou mais *pixels* no lugar das transições entre dois *pixels*, propostas inicialmente. Assim, para três vizinhanças, as transições a serem verificadas seriam: 000, 001, 010, 011, 100, 101, 110 e 111. Com estas duas estratégias, aumenta-se a quantidade de informações/restrições usadas e, consequentemente, aumenta-se as chances de se classificar os caracteres corretamente. De fato, usando uma quantidade j de vizinhanças, se resolve ambiguidades que ocorrem quando se usa uma quantidade $j-1$ de vizinhanças. Isto é ilustrado na Figura 3, que mostra as ambiguidades que ocorrem usando transições entre dois *pixels* sendo resolvidas, usando três *pixels*. Nesta figura é possível observar que usando apenas as transições entre dois *pixels*,

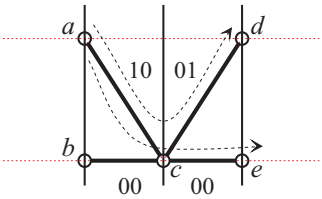


Figura 3. Ambiguidades que ocorrem usando transições entre dois *pixels*, resolvidas usando três *pixels*.

não se sabe se a poligonal que começa em a, vai para c e depois para d, ou se começa em a, vai para c e depois para e. O mesmo vale para a poligonal que começa em b (b,c,e ou b,c,d ?).

Nesta figura, a ambiguidade surge porque existem quatro possibilidades para as duas poligonais que iniciam em a e b, e terminam em d e e. Conforme se aumenta o número de vizinhanças nas transições, o classificador elimina tais ambiguidades, porém, se torna menos flexível. Assim, quando se considera as transições entre apenas dois *pixels*, o classificador pode inserir em uma classe, registros apenas parecidos com os usados no seu treinamento. Porém, se forem usadas as transições entre todas as vizinhanças possíveis, o classificador somente será capaz de classificar os objetos idênticos aos usados no seu treinamento.

4. Experimentos

Esta seção apresenta dois experimentos aplicando a técnica proposta neste trabalho. No primeiro é usado um conjunto de caracteres manuscritos, que tem sido amplamente utilizado para o teste de classificação, por causa da sua complexidade. O segundo experimento utiliza um conjunto contendo caracteres impressos, usando diferentes fontes.

4.1 Experimento 1

Nesse experimento é apresentada uma análise do conjunto de dados *binaryalphadigs* (Frank & Asuncion, 2010), usando a estratégia proposta. Este conjunto de dados é formado por 390 registros, obtidos a partir das imagens de 39 exemplos dos algarismos 0, 1, ..., 9, escritos à mão, conforme ilustra a Figura 4.

No experimento realizado com este conjunto, cada caractere foi reamostrado usando uma grade com uma resolução de 16×20 *pixels*, usando apenas as cores preto e branco (imagens binárias). Assim, cada caractere é representado neste conjunto através de 320 atributos (*pixels*) que podem assumir os valores zero (preto) ou um (branco).

A Figura 5 mostra as transições entre os 320 atributos usando coordenadas paralelas. A visualização dos registros em algumas de suas classes

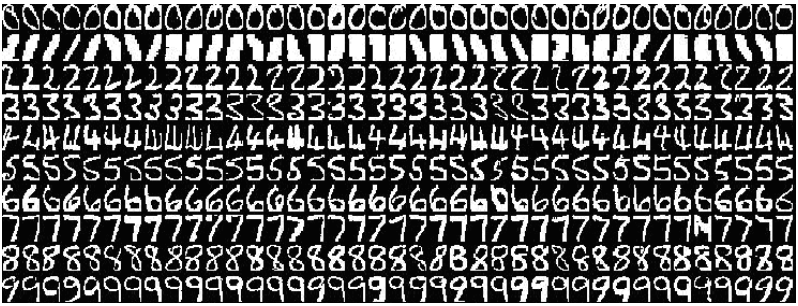


Figura 4. Conjunto de dados *binaryalphadigs* (Frank & Asuncion, 2010), contendo trinta e nove caracteres (números: 0, 1, ..., 9) escritos à mão.

revela um comportamento particular para cada classe, evidenciando diferentes conjuntos de restrições para cada classe.

As quantidades de restrições nas classes zero até nove são, respectivamente, 228, 315, 155, 197, 117, 188, 196, 277, 88 e 272. Assim, a classe dos caracteres "1" é a mais exigente (possui um número maior de restrições), enquanto que a classe dos caracteres "8" possui o menor número de restrições. Como a cardinalidade de todos os atributos é igual a dois, e o conjunto tem 320 atributos, o número total de transições entre os atributos do conjunto, obtido usando a Equação 1, é dado por 1.276.

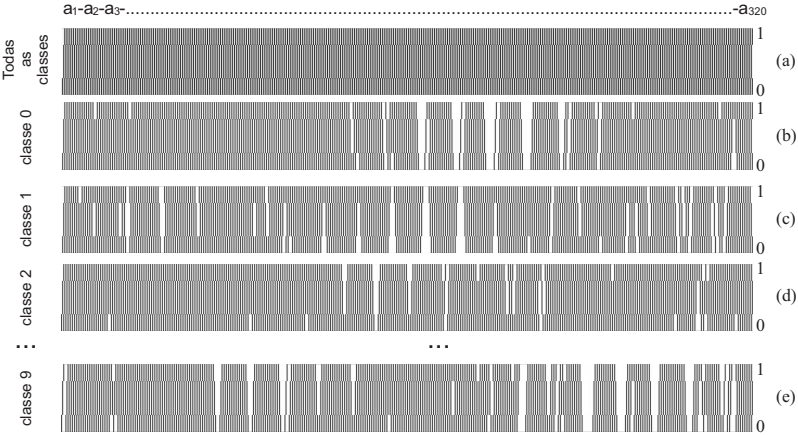


Figura 5. Visualização obtida usando: a) todas as transições de todos os caracteres nas dez classes; b) apenas os registros da classe do caractere "0"; c) apenas os registros da classe do caractere "1"; d) apenas os registros da classe do caractere "2"; e) apenas os registros da classe do caractere "9".

A matriz de confusão apresentada na Figura 6 ilustra a eficácia da estratégia para classificar os caracteres de acordo com as regras de classificação propostas, ou seja, o caractere é inserido na classe cujas transições não são violadas pelas transições impostas pela classe. Nos casos em que se encontra mais de uma classe satisfazendo esta condição, o caractere é inserido na classe menos violada que possui mais restrições.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	39	0	0	0	0	0	0	0	0	0
1	0	39	0	0	0	0	0	0	0	0
2	0	0	39	0	0	0	0	0	0	0
3	0	0	0	39	0	0	0	0	0	0
4	0	0	0	0	39	0	0	0	0	0
5	0	0	0	1	0	38	0	0	0	0
6	0	0	0	0	0	0	39	0	0	0
7	0	0	0	0	0	0	0	39	0	0
8	0	0	0	0	0	0	0	0	39	0
9	0	0	0	0	0	0	0	0	0	39

Figura 6. Matriz de confusão obtida com a classificação usando as transições entre os atributos adjacentes.

Apesar da dificuldade com este conjunto de caracteres (Figura 4), a matriz de confusão mostra que a maior parte dos registros foi colocada em suas devidas classes, exceto um registro da classe 5 (caractere “5”), que foi colocado na classe 3 (caractere “3”). Trata-se do 23^o caractere “5” na Figura 4, que atende todas as restrições das classes 3 e 5 e, pela regra R2 (regras de classificação propostas), foi reconhecido como da classe 3, porque ela possui um número maior de restrições que a classe 5.

Conforme apontado anteriormente, uma solução imediata para este problema é aumentar o tamanho da malha em que a imagem é amostrada, consequentemente, gerando uma quantidade maior de transições. Este efeito pode ser confirmado nas Figuras 7, 8 e 9 que mostram os resultados obtidos com diferentes tamanhos de malha. Na Figura 7 tem-se um resultado ruim usando a malha 10x8 (reduzida). Nos experimentos realizados, os caracteres classificados erroneamente são delimitados por retângulos.

Na Figura 8, tem-se o resultado já analisado na matriz de confusão da Figura 6, que usa a malha 20×16, e classifica um caractere “5”, na classe dos caracteres “3”.

Na Figura 9, amostrando os caracteres sobre uma malha com resolução 40×32, tem-se um resultado ótimo, com todos os caracteres classificados em suas devidas classes. A explicação para a melhoria nos resultados é que quando se aumenta a malha, aumenta-se também a quantidade de transições, o que diminui a chance de se classificar os caracteres em classes erradas.

Em seguida, utilizando transições entre três *pixels*, tem-se o seguinte resultado para a malha 10×8, ilustrado na Figura 10, onde se verifica

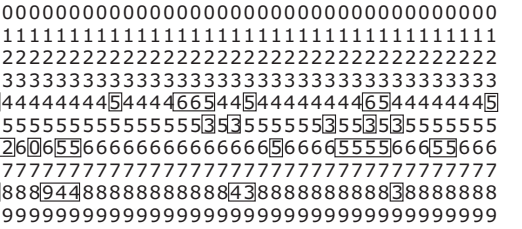


Figura 7. Resultado ruim da classificação usando uma malha 10×8 com transições entre 2 *pixels*.

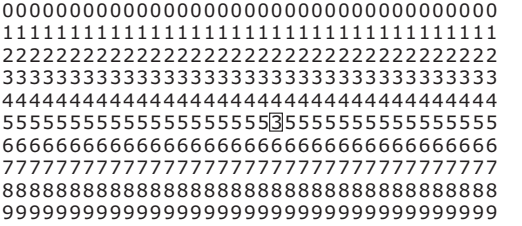


Figura 8. Resultado razoável da classificação usando uma malha 20×16 com transições entre 2 *pixels*.

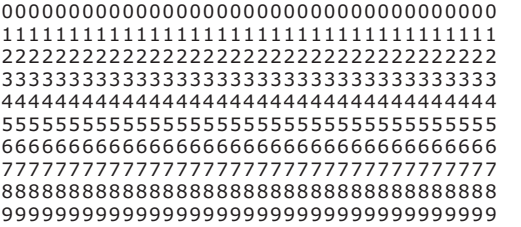


Figura 9. Resultado ótimo da classificação usando uma malha 40×32 com transições entre 2 *pixels*.

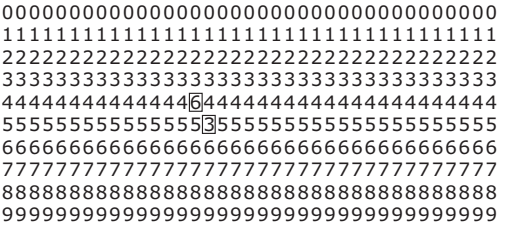


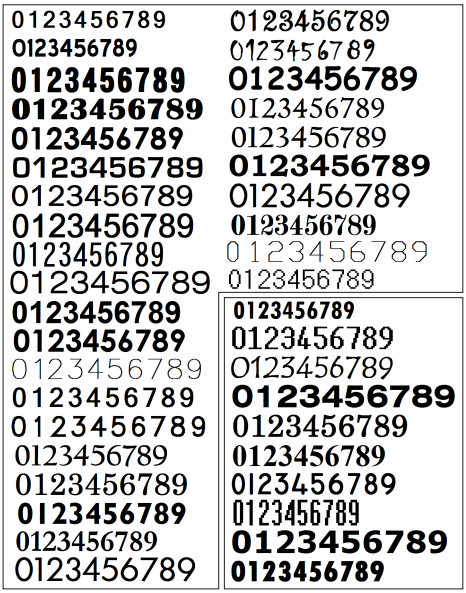
Figura 10. Resultado melhor para a classificação usando uma malha 10×8 com transições entre 3 *pixels*.

uma quantidade de erros bem menor, se comparada com o resultado da Figura 7, que também usa a malha 10×8, porém, usando transições entre dois *pixels*.

Para as malhas 20×16 (Figura 8), o uso das transições entre três *pixels* elimina completamente os erros de classificação. Do mesmo modo, o uso de uma vizinhança de quatro *pixels* para definir as transições, também elimina todos os erros, mesmo com a resolução baixa de 10×8.

4.2 Experimento 2

Nesse experimento é apresentada uma análise da classificação usando o conjunto de imagens de caracteres exibido na Figura 11. Neste experimento, o treinamento do classificador foi realizado usando apenas os caracteres em (a), enquanto que o teste foi feito usando apenas os caracteres em (b), observando que se trata de um conjunto de fontes diferentes.



(a) (b)

Figura 11. a) Conjunto de caracteres usados no treinamento do classificador (30 fontes diferentes); b) Conjunto usado no teste (outras 10 fontes diferentes).

Os resultados deste experimento são apresentados na Figura 12, sendo que em (a), são utilizadas as transições entre dois *pixels*, gerando um total de quatro erros de classificação, em (b), usando as transições entre três *pixels*, foi obtida a classificação correta para todos os caracteres, enquanto que em (c), usando as transições entre quatro *pixels*, também foi obtida a classificação sem erros para todos os caracteres.

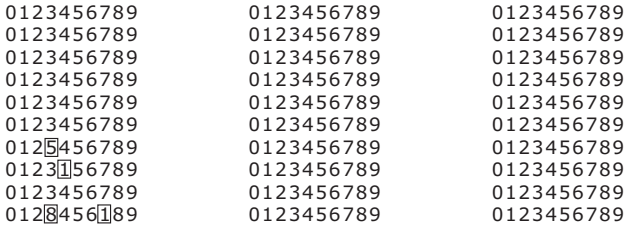


Figura 12. Resultado da classificação usando uma malha 40×32 com transições a) dois *pixels*; b) três pixels; c) quatro *pixels*.

O experimento a seguir ilustra o uso da proposta apresentada neste trabalho, com caracteres alfabéticos (a..z e A..Z), além dos dígitos (0..9). A Figura 13(a) mostra o conjunto de caracteres usados no treinamento deste classificador, que no caso corresponde aos caracteres alfanuméricos usando a fonte *Times New Roman*.

Neste caso, como existe uma diferença entre os tamanhos dos caracteres maiúsculos e minúsculos, deve-se realizar o recorte dos caracteres com um tamanho adequado, obedecendo a sua altura, tanto na etapa de treinamento, quanto na etapa de classificação. Isto é ilustrado na Figura 14, que mostra como os recortes dos caracteres "a" e "A" devem ser feitos. De fato, não existe dificuldade alguma neste processo, pois todos os caracteres são enquadrados automaticamente dentro de retângulos, todos com a mesma altura, independentemente de se tratar de um caractere maiúsculo ou minúsculo.

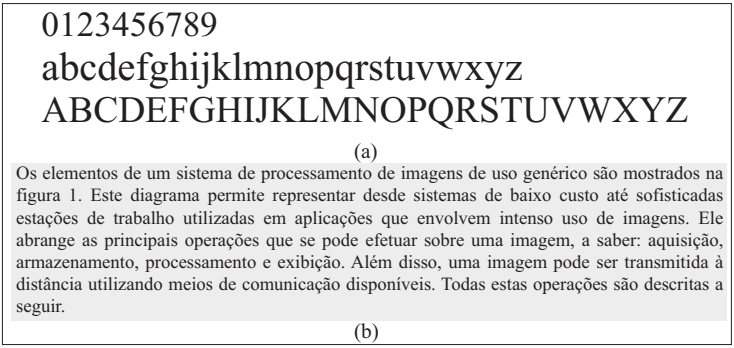


Figura 13. a) Caracteres alfanuméricos usando a fonte *Times New Roman*, usados no treinamento do classificador; b) Imagem de um parágrafo digitalizado do livro de (Marques Filho & Vieira Neto, 1999, página 2).

Figura 14. Enquadramento diferenciado para caracteres maiúsculos e minúsculos.

A Figura 15 apresenta o texto extraído da imagem na Figura 13(b), com esta classificação. Neste caso, pode-se observar apenas alguns erros: o caractere "1" (um) foi reconhecido como a letra "l" (L), por causa da grande semelhança entre estes caracteres usando a fonte *Times New Roman*; caracteres acentuados, devido a sua não inclusão no conjunto de treinamento do classificador (Figura 13(a)); pontuações (vígulas, pontos finais) também não foram reconhecidos pelo mesmo motivo.

Os elementos de um sistema de processamento de imagens de uso gen□rico s□o mostrados na figura 1 Este diagrama permite representar desde sistemas de baixo custo at□ sofisticadas esta□□es de trabalho utilizadas em aplica□□es que envolvem intenso uso de imagens Ele abrange as principais opera□□es que se pode efetuar sobre uma imagem a saber aquisi□□o armazenamento processamento e exibi□□o Al□m disso uma imagem pode ser transmitida a dist□ncia utilizando meios de comunica□□o dispon□veis Todas estas opera□□es s□o descritas a seguir

Figura 15. Texto extraído da imagem apresentada na Figura 14(b).

Em seguida é apresentada uma comparação entre alguns programas de OCR, que podem ser encontrados na Internet. Neste estudo são comparados a qualidade da classificação, o que é feito através de uma contagem dos erros de classificação dos caracteres. Também é feita uma comparação entre os tempos de execução. Os programas usados são:

- FreeOCR.net - Trata-se de um programa OCR que inclui o núcleo *Tesseract free OCR*, que pode ser usado com os *drivers* Twain e WIA. Este núcleo foi desenvolvido pela Hewlett Packard entre 1985 e 1995. Atualmente está disponível sob a forma *open-source*, mantido pela Google Inc. Este programa pode ser encontrado em: http://www.freeocr.net;
- SimpleOCR 3.1 - Este programa foi desenvolvido pela Simple-Software e é distribuído na modalidade *freeware*, podendo ser usado

por usuários domésticos, instituições educacionais e também usuários corporativos. Este programa pode ser encontrado no *website* http://www.simpleocr.com;

- A-PDF-OCR - Trata-se de um programa comercial, desenvolvido pela APDF, que pode ser encontrado em http://www.a-pdf.com;
- Cuneiform Pro OCR 6.0 - Este programa foi desenvolvido pela Give Me Freeware e usa alfabetos para vinte idiomas diferentes, incluindo o Português. Pode ser encontrado no *website* http://freeware.odlican.net;
- Image2pdf OCR 3.2 - Trata-se de um programa *freeware*, desenvolvido pela SoftSolutions, que pode ser obtido em http://products.softsolutionslimited.com;
- ABBYY FineReader 11 Professional - Trata-se de um programa comercial, desenvolvido pela ABBYY USA Software House, que pode ser encontrado em http://www.abbyy.com.

A Figura 16 mostra o resultado da comparação da qualidade destes seis programas e também do algoritmo apresentado neste trabalho. A Figura 17 mostra o resultado da comparação dos tempos de execução destes seis programas e também do algoritmo apresentado neste trabalho. A imagem utilizada para a classificação contém 3718 caracteres.

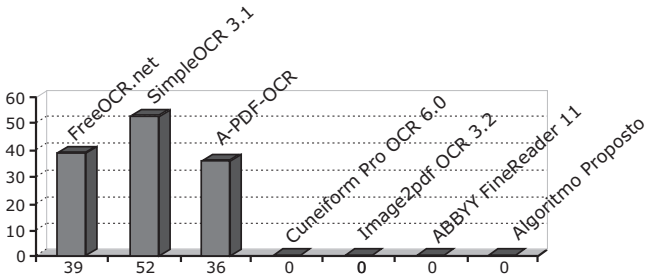


Figura 16. Contagem de erro de classificação.

4.3 Análise de desempenho

Os tempos médios de processamento, em segundos, do algoritmo proposto, usando o conjunto de dados *binaryalphadigs* (Figura 4) é apresentado na Tabela 2, usando transições entre dois, três e quatro *pixels*, com diferentes resoluções de malha. A máquina usada possui um processador Intel Core i3 M330 de 2.13GHz e 4GB de RAM.

A Tabela 3 apresenta os tempos médios, em segundos, de processamento, usando os caracteres impressos da Figura 11. Neste caso, são usadas transições entre dois, três *pixels* e quatro *pixels*, com a resolução de malha de 40×32.

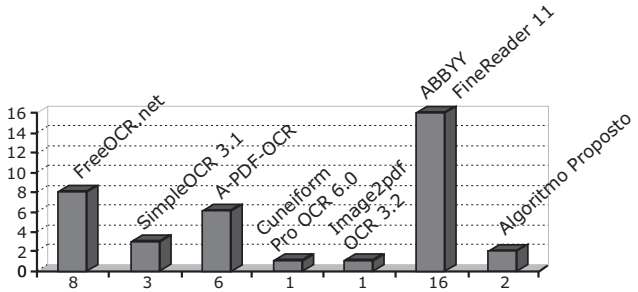


Figura 17. Tempos de excecução (segundos).

Estes resultados mostram que o algoritmo proposto consegue identificar caracteres em uma taxa média de 316,19 caracteres por segundo, usando a transição de 4 *pixels* e 923,36 caracteres por segundo, usando a transição de 3 *pixels*, com a malha 40×32 (melhores resultados). Usando a transição de 2 *pixels*, com uma malha 10×8, o algoritmo consegue identificar caracteres em uma taxa média de 28.994,75 caracteres por segundo, o que é bastante razoável para tarefas que precisam realizar identificações de caracteres em tempo real.

A Figura 18 mostra como o tempo de processamento e a quantidade de erros obtidos estão relacionados com as quantidades de *pixels* usadas nas transições. Foram utilizados os dados obtidos com o experimento usando o conjunto de dados *binaryalphadigs* (malha 10×8) para construir os gráficos.

5. Conclusões

A classificação de caracteres usando as transições entre os níveis dos *pixels* adjacentes se mostrou bastante eficiente, mesmo com um conjunto de caracteres tão difícil como os caracteres manuscritos, pois como se observa, o caractere "1" (Figura 4), por exemplo, apresenta uma grande variação, que tem desafiado a maioria dos algoritmos conhecidos de reconhecimento de caracteres. A descrição dos caracteres a partir do comportamento das tran-

Tabela 2. Tempos médios de processamento (segundos) para um caractere usando o algoritmo proposto (conjunto de dados *binaryalphadigs* – Figura 4).

	Resoluções		
Transições	10×8	20×16	40×32
2 *pixels*	0,0000344	0,0001511	0,0006261
3 *pixels*	0,0000782	0,0002851	0,0010766
4 *pixels*	0,0001614	0,0005958	0,0031556

Tabela 3. Tempos médios de processamento (em segundos) para um caractere usando o algoritmo proposto (caracteres da Figura 11).

	Resolução
Transições	40×32
2 *pixels*	0,000683
3 *pixels*	0,001083
4 *pixels*	0,001849

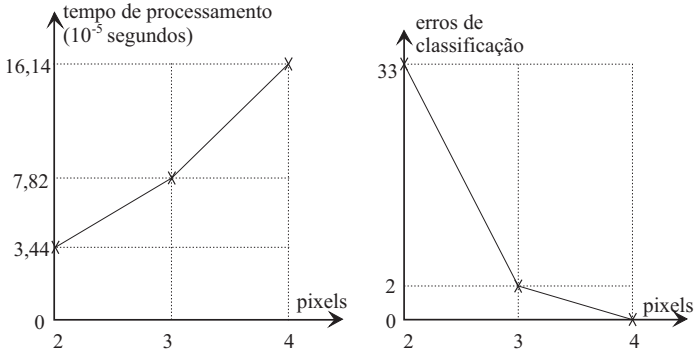


Figura 18. a) Tempos de processamento × quantidade de *pixels* usadas nas transições; b) Erros de classificação × quantidade de *pixels* usadas nas transições.

sições de níveis entre os *pixels* vizinhos se mostrou uma estratégia muito simples, rápida e eficiente, que pode ser facilmente implementada em hardware e usada em aplicações em tempo real. Este é o caso dos sistemas de visão computacional que obtêm suas imagens a partir de câmeras de vídeo, como robôs autônomos e veículos rápidos não tripulados. O uso de malhas de maiores dimensões resulta em um aumento no número de transições entre *pixels*, contribuindo para a melhoria dos resultados. Do mesmo modo, o uso de transições entre três *pixels* também contribui para aumentar o número de combinações e transições, descrevendo melhor cada classe de caracteres. Observando que um aumento excessivo na quantidade de transições gera uma perda da capacidade de generalização da classificação. Este algoritmo também está sendo aplicado em reconhecimento automático de placas de sinalização de velocidade, para fins de georreferenciamento automático das mesmas, com resultados bem promissores.

Referências

Dimond, T.L., Devices for reading handwritten characters. In: *Proceedings of Eastern Joint Computer Conference*. p. 232–237, 1957.

Duda, R.O.; Hart, P.E. & Stork, D.G., *Pattern Classification*. 2a edição. New York, USA: John Wiley & Sons, 2001.

Eden, M., On the formalization of handwriting. In: *Proceedings of the Fourth London Symposium on Information Theory*. 1961.

Eden, M. & Halle, M., The characterization of cursive writing. In: *Proceedings of the Fourth London Symposium on Information Theory*. p. 287–299, 1961.

Frank, A. & Asuncion, A., UCI machine learning repository. 2010. http://archive.ics.uci.edu/ml/.

Frishkopf, L.S. & Harmon, L.D., Machine reading of cursive script. In: *Proceedings of the Fourth London Symposium on Information Theory*. p. 300–316, 1961.

Gonzalez, R.C. & Woods, R.E., *Digital Image Processing*. 2a edição. Reading, USA: Addison-Wesley, 2001.

Handel, P.W., *Statistical Machine*. 1933. U.S. Patent 1 915 993.

Inselberg, A., The plane with parallel coordinates. *The Visual Computer*, 1(2):69–91, 1985.

Jain, A.K.; Duin, R.P.W. & Mao, J., Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

Jain, G. & Ko, J., *Handwritten Digits Recognition*. Multimedia Systems Project Report, University of Toronto, 2008.

Lin, H.; Ou, W. & Zhu, T., The research of algorithm for handwritten character recognition in correcting assignment system. In: *Proceedings of the Sixth International Conference on Image and Graphics*. p. 456–460, 2011.

Marques Filho, O. & Vieira Neto, H., *Processamento Digital de Imagens*. Rio de Janeiro, RJ: Brasport, 1999.

Montaña, E.G., Digits recognition via neural networks. 2007. http://ociotec.com/wp-content/uploads/2007/07/DigitsRecognition.pdf.

Neisser, U. & Weene, P., A note on human recognition of hand-print characters. *Information and Control*, 3(2):191–196, 1960.

Pavlidis, T., *Structural Pattern Recognition*. Heidelberg, Germany: Springer-Verlag, 1980.

Pereira, J.F.; Alves, V.M.O.; Cavalcanti, G.D.C. & Ren, T.I., Modular image principal component analysis for handwritten digits recognition. In: *Proceedings of the Seventeenth International Conference on Systems, Signals and Image Processing*. p. 356–359, 2010.

Schalkoff, R.J., *Pattern Recognition: Statistical, Structural and Neural Approaches*. New York, USA: John Wiley & Sons, 1992.

Shrivastava, S.K. & Gharde, S.S., Support vector machine for handwritten devanagari numeral recognition. *International Journal of Computer Applications*, 7(11):9–14, 2010.

Tauschek, G., *Reading Machine*. 1935. U.S. Patent 2 026 329.

Trentini, V.B.; Godoy, L.A.T. & Marana, A.N., Reconhecimento automático de placas de veículos. In: *Anais do VI Workshop de Visão Computacional*. p. 267–272, 2010.

Zhu, X.; Shi, Y. & MA, S., Research on handwritten character recognition. *Pattern Recognition and Artificial Intelligence*, 13(2):172–180, 2000.

Notas Biográficas

Francisco Assis da Silva possui graduação em Ciência da Computação (Universidade do Oeste Paulista – UNOESTE, 1998), mestrado em Ciência da Computação (Universidade Federal do Rio Grande do Sul – UFRGS, 2002) e doutorado em Engenharia Elétrica na área de Visão Computacional (Universidade de São Paulo – USP/São Carlos, 2012). Atualmente é docente da UNOESTE/Presidente Prudente.

Almir Olivette Artero possui graduação em Matemática (Universidade Estadual Paulista – UNESP, 1990), especialização em Sistemas de Informação (Universidade do Oeste Paulista – UNOESTE, 1995), mestrado em Ciências Cartográficas (UNESP, 1999) e doutorado em Ciência da Computação (Universidade de São Paulo – USP, 2005). Atualmente é docente da UNESP/Presidente Prudente).

Maria Stela Veludo de Paiva possui graduação em Engenharia Elétrica/Eletrônica (Universidade de São Paulo – USP, 1979), mestrado e doutorado em Física Aplicada (USP/São Carlos, 1984 e 1990), tendo realizado Pós-Doutorado na University of Southampton (1992). Atualmente é docente no Departamento de Engenharia Elétrica da Escola de Engenharia de São Carlos (USP) e desenvolve pesquisas na área de Visão Computacional.

Ricardo Luís Barbosa possui graduação e especialização em Matemática (Universidade Estadual Paulista – UNESP, 1990 e 1993), mestrado e doutorado em Ciências Cartográficas (UNESP, 1999 e 2006). Atualmente é docente da UNESP/Sorocaba e desenvolve pesquisas na empresa Cartovias Engenharia Cartográfica.