AOCR – Adaptive Optical Character Recognition (08 Janeiro 2010)

T. M. D. Bruno, F. S. Douglas, G. J. Rafael

***Resumen*— O trabalho desenvolvido consistiu na elaboração e implementação de um software OCR (Optical Character Recognition) utilizando tecnologia adaptativa. O objetivo do trabalho foi a melhoria do desempenho geral do sistema através da utilização desta tecnologia, elevando sua taxa de acerto e possibilitando a inserção gradual de novas regras adaptativas que possibilitassem a inclusão de diferentes alfabetos/fontes. A primeira parte do projeto consistiu em uma análise dos OCR's e dos processadores de imagem disponíveis no mercado e o modo de funcionamento dos mesmos. Na segunda etapa, realizou-se a elaboração da especificação do software, em que todos os requisitos funcionais e não-funcionais necessários para a elaboração do programa foram listados. Na terceira etapa, houve o estudo da teoria adaptativa e as possíveis formas de inserção desta tecnologia no software, visando à melhoria do desempenho e o aumento do número de funcionalidades. A última fase foi composta pelo desenvolvimento do software e dos algoritmos adaptativos, realização dos testes de cada funcionalidade, assim como a elaboração de toda documentação.**

I. Introdução

O reconhecimento óptico de caracteres é uma dos ramos mais antigos e pesquisados na área de reconhecimento de padrões. Os primeiros OCR's foram desenvolvidos na década de 50 por empresas e centros de pesquisa no exterior.

Atualmente, existe um potencial muito grande a ser explorado no mercado dos OCR's, tendo em vista que apenas algumas grandes e médias empresas de desenvolvimento de softwares desenvolvem tais aplicativos. Esses aplicativos são na maioria das vezes restritos a um alfabeto e possuem um preço muito elevado.

Inserido neste contexto, este trabalho teve como objetivo o desenvolvimento o primeiro OCR que utiliza tecnologia adaptativa. O AOCR é um software que tem como objetivo realizar a tradução de imagens de diversos formatos (PNG, JPEG, TIFF e BMP) em arquivos de texto editáveis (TXT), utilizando técnicas de inteligência artificial e tecnologia adaptativa. Por usar essas tecnologias é possível incluir novas funcionalidades além da tradicional conversão, o que torna este programa único. Essas principais funcionalidades são: aprendizado de novos alfabetos/fontes, aprendizado a partir dos erros cometidos, dentre outras que serão descritas neste artigo.

II. Desenvolvimento

Este projeto teve início em Março de 2009 e foi concluído em Dezembro de 2009. Todas as etapas que fazem parte do projeto (Concepção, Documentação, Desenvolvimento e Testes) foram realizadas durante o último ano do curso de engenharia de computação (ênfase semestral) da Escola Politécnica da USP dos formandos de 2009.

Durante a elaboração e a construção do programa, tentou-se criar um sistema de fácil utilização e de desempenho maior ou próximo aos encontrados nos melhores programas. Nos melhores softwares disponíveis no mercado, a taxa de acerto fica em torno de 97% (utilização de métodos estatísticos e de corretores ortográficos) e o tempo de processamento de uma página é de 1-3 minutos. Vale ressaltar que todos os OCR's realizam tais processamentos com apenas um alfabeto e uma fonte, o que os torna muito eficientes quando utilizados textos que contenham tal alfabeto e fonte.

A. Funcionamento geral do programa

Antes da etapa de desenvolvimento do programa AOCR, houve uma etapa de pesquisa e teste dos programas OCR's de código aberto disponíveis no mercado, como OCRAD [7], OCRopus [8], Tesseract [9] e GOCR [6]. Após diversos testes realizados, verificou-se que o melhor OCR disponível gratuitamente e com o código aberto era o aplicativo GOCR.

Este programa apresenta seu código fonte na linguagem C e está dividido em módulos que facilitam o entendimento de seu funcionamento. Portanto, este sistema foi escolhido como o software base, em que foram realizadas todas as alterações necessárias para a inclusão do algoritmo adaptativo, assim como o tratamento das novas fontes e novos alfabetos. A partir do GOCR foram extraídas somente as rotinas de tratamento de imagem. Os outros módulos do programa não foram utilizados, uma vez que os mesmos eram incompatíveis com os objetivos inicialmente planejados neste trabalho.

O OCR é um sistema responsável pelo processamento de textos no formato de imagem em texto em formatos editáveis. Primeiramente, a imagem é tratada pelo processador de imagem. Em seguida, os parâmetros de cada um dos símbolos são extraídos da figura. Por último, esses parâmetros são utilizados pelos algoritmos de decisão do OCR. Um diagrama do funcionamento básico do programa pode ser visto na Figura 1, em que os principais módulos do software podem ser visualizados.

Fig. 1. Entradas e saídas do algoritmo de decisão.

A funcionalidade básica do software permite o acesso às imagens e a visualização do texto processado através da tela principal do programa. Além da funcionalidade principal, o AOCR possui uma interface gráfica que permite o usuário utilizar todas as principais funcionalidades.

Inclusão/remoção de um novo alfabeto: diferentemente dos OCR's convencionais presentes no mercado, o AOCR permite a inserção e remoção de alfabetos/fontes através de sua interface, ou seja, o usuário é capaz de treinar o programa com os alfabetos/fontes que desejar. Diversos testes foram realizados com sucesso utilizando os alfabetos/fontes latino, grego, cirílico, texto escrito e símbolos matemáticos.

Correção de parâmetros: esta funcionalidade permite a correção dos parâmetros de um determinado caractere. Caso tenha ocorrido uma tradução errada devido ao fato de um parâmetro da letra estar errado, o usuário pode corrigir este parâmetro através da interface.

Escolha dos alfabetos: o usuário pode escolher um alfabeto/fonte ou um conjunto de alfabetos que serão utilizados como fonte de dados para a realização das traduções. Dessa forma, há uma otimização no tempo de processamento, não significando necessariamente uma melhora na taxa de acerto.

Testes automatizados: o usuário pode realizar a tradução de diversos arquivos através desta funcionalidade, diminuindo o tempo de processamento quando são submetidos diversos arquivos.

Foram utilizadas tecnologias de diversas áreas da computação durante o desenvolvimento do projeto. A seguir, as principais tecnologias serão descritas em mais detalhes, assim como a utilização das mesmas no programa.

B. Reconhecimento de padrões

Todos os 37 parâmetros utilizados para determinar um caractere foram criados durante a elaboração do programa. Inicialmente, cria-se um retângulo ao redor do símbolo que será processado. Então o todo o processamento necessário para a obtenção dos parâmetros é realizado. Alguns dos parâmetros resultantes de tal processamento são:

Número de buracos: quantidade de buracos da letra, sendo que um buraco é caracterizado como pixels brancos delimitados por pixels pretos.

Número de linhas horizontais: quantidades de linhas horizontais presentes na letra, sendo que linhas horizontais são caracterizadas como pixels pretos sem interrupção na direção horizontal.

Número de linhas verticais: quantidades de linhas verticais presentes na letra, sendo que linhas verticais são caracterizadas como pixels pretos sem interrupção na direção vertical.

Relação altura/largura: esta é a relação da largura pela altura do retângulo que está ao redor da letra.

Centro de massa proporcional: o centro de massa é calculado de acordo com a seguinte fórmula:

$$CM = \frac{\sum (PixelPreto) * (PosiçãodoPixel)}{TotaldePixelsPretos}$$

Densidade: a densidade é calculada pela quantidade de pixels pretos em relação ao total de pixels da letra, conforme a fórmula a seguir:

$$CM = \frac{\sum (PixelsPreto)}{TotaldePixels}$$

Número de entradas: este parâmetro verifica quantas entradas a letra possui em cada aresta do quadrado que a circunscreve, isto é, verificam-se quantos pixels brancos contínuos existem entre os pixels pretos no limiar da caixa.

Número de linhas cruzadas: este parâmetro verifica o número de linhas cruzadas por um feixe passando pelo centro (horizontal e vertical) da caixa ao redor da letra.

A partir desses parâmetros, a maioria dos caracteres pode ser classificada sem a necessidade de intervenção das técnicas mais avançadas, como adaptativas. Na Figura 2, pode-se observar uma representação dos caracteres latinos de mesma classe que podem na maioria das vezes causam problemas no momento do processamento.

Fig. 2. Classes de objetos semelhantes.

C. *Processamento de imagens*

Antes que os parâmetros de cada símbolo sejam adquiridos, faz-se necessário um tratamento da imagem. Primeiramente, realiza-se a delimitação da região que contém o símbolo. Em seguida, algumas imperfeições presentes no arquivo são removidas, como buracos não fechados, pontos extras ao redor do caractere e ruídos presentes nos contornos. Por último, os caracteres são processados e os atributos enviados para os algoritmos de decisão;

D. *Tecnologia adaptativa*

A utilização desta tecnologia visa melhorar o desempenho do programa, tendo em vista que diversas ações corretivas podem ser aplicadas para eliminar incoerências já conhecidas e comumente encontradas em OCR's atuais. Foram desenvolvidos dois algoritmos que utilizavam árvores de decisão adaptativas para realizarem as escolhas do caractere que mais se aproxima com as características extraídas da figura. Além disso, o usuário pode realizar correções dos parâmetros de cada caractere através da tela de correção de parâmetros. Assim, pode-se interagir com os dados presentes no banco de dados, aumentando a taxa de acerto dos algoritmos do programa.

E. *Algoritmos de decisão*

No que diz respeito ao processo de decisão dos símbolos, realizaram-se algumas estratégias utilizando técnicas adaptativas para superar problemas conhecidos de árvore de decisões, como overfitting[1]. Além disso, visou-se dar maior flexibilidade em relação aos parâmetros, ou seja, os parâmetros da letra sempre vão variar. Caso os valores variem mais que um determinado limite, o algoritmo resultará em um ramo errado da árvore, fornecendo um caractere errado na saída. Com a utilização da adaptatividade é possível tratar este problema, uma vez que mais de um resultado pode ser obtido, como poderá ser visto nos próximos itens.

Todas as técnicas de decisão utilizam o método de ordenação dos parâmetros. Este método utiliza conhecimentos prévios dos parâmetros de desvio padrão e a média, além de uma nota dada pelos conhecimentos prévios dos desenvolvedores, para decidir qual o melhor parâmetro. Define-se o melhor parâmetro como aquele que possuir melhores notas nas três classes a seguir:

1. Confiabilidade: quanto menos este parâmetro variar dentre o que já foi observado para o alfabeto/fonte que está sendo utilizado, melhor será sua confiabilidade. Exemplo: o número de buracos de uma letra é um parâmetro que possui uma alta confiabilidade, porque na maioria dos casos os caracteres possuem uma quantidade de buracos em uma faixa pequena de possíveis valores.
2. Dispersão: quanto mais distantes os valores das letras estão, melhor será para realizar a decisão de qual é a letra, ou seja, o quão bem o parâmetro em questão consegue distinguir qual é a letra correta. Exemplo: o número de buracos não possui uma boa dispersão, visto que a maioria das letras possui zero ou um buraco(s).
3. Classe: uma nota prévia fornecida, que pode ser alterada pelo usuário, para cada parâmetro.

$$PesoParâmetro = Classe + (Pontos_{dispersão} + Pontos_{confiabilidade})$$

As técnicas de decisão disponíveis no programa são quatro. Cada uma delas serão descritas detalhadamente a seguir:

1. Método Estatístico;
2. Árvore de decisão simples;
3. Árvore de decisão adaptativa não-determinística;
4. Árvore de decisão adaptativa "pura".

No primeiro método, são comparados todos os parâmetros de todas as letras dos alfabetos/fontes escolhidos, com os parâmetros encontrados para a letra em questão. Verifica-se a distância entre tais letras, diminuindo a probabilidade de ser a letra conforme esta distância aumenta. Cada parâmetro possui um peso diferente, que influencia na diminuição de probabilidade, baseado na qualidade deste parâmetro (utilizando o método demonstrado anteriormente). Desta forma, a perda da probabilidade segue a seguinte fórmula (para cada parâmetro):

$$Perda = \|Distância\| * CONSTANTE * Peso\_Parâmetro$$

Todos os métodos que utilizam uma árvore de decisão preparam a árvore de forma idêntica, visando deixar os parâmetros com maior qualidade perto da raiz da árvore, e procurando dividir o alfabeto/fonte de forma inteligente, visando dar a maior robustez possível a esta árvore. Desta forma, a montagem da árvore funciona da seguinte forma:

1. Escolhe-se o melhor parâmetro dentre os ainda disponíveis, a partir do alfabeto/fonte disponível no nó atual;
2. Descobre-se o valor mínimo e máximo. Divide-se esse conjunto em 10 partes iguais;
3. Verifica-se a densidade das letras do alfabeto/fonte em cada um dos conjuntos;
4. Verifica-se qual o grupo com a menor densidade;
5. Unem-se esses grupos de menor densidade caso sejam adjacentes;
6. Dividem-se os ramos no centro dessas áreas de menor densidade;
7. Caso o novo nó possua apenas uma letra, o processo acaba. Caso contrário, o processo volta ao passo 1 para cada novo ramo, removendo o parâmetro utilizado nesse passo.

Considerando que no primeiro passo foi escolhido o parâmetro número de buracos, para o alfabeto/fonte latino-arial, a Figura 3 demonstra os passos 2 ao 6:

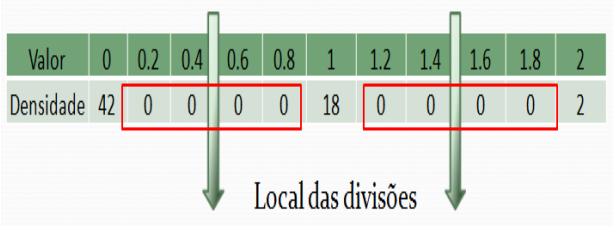

Fig. 3. Distribuição do parâmetro número de buracos das letras da alfabeto/fonte latino-arial.

A árvore de decisão simples percorre a árvore. Ao encontrar uma folha, verifica se existe apenas uma letra, que será a resposta. Caso possua mais que uma letra, utiliza o método estatístico para decidir qual é a mais adequada.

Por fim, ambas as árvores adaptativas utilizam métodos de consulta idênticos, variando-se apenas na ação de adição e remoção adaptativas utilizada.

A ação de consulta consiste em verificar se o valor do parâmetro sendo utilizado está em uma das áreas de incerteza.

Define-se área de incerteza os 10% iniciais e finais de cada ramo. No primeiro ramo a área de incerteza é ao final do ramo e anterior ao início do ramo. Isto também ocorre no último ramo, em que a área de incerteza é o início e após o fim, ou seja, valores nunca vistos anteriormente para aquele valor.

A Figura 4 demonstra as áreas de incerteza para o parâmetro número de buracos, com alfabeto/fonte latino-arial, em que o verde é a divisão dos ramos e em vermelho estão definidas as áreas de incerteza:

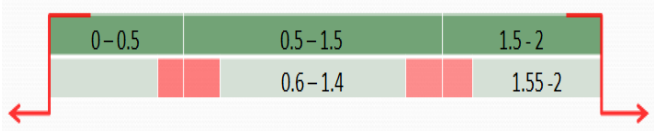

Fig. 4. Incerteza do parâmetro número de buracos das letras da alfabeto/fonte latino-arial.

Logo, quando algum parâmetro está na área de incerteza, aciona-se a ação adaptativa correspondente, que varia de acordo com o tipo de árvore que foi escolhido:

1. Não-determinística: segue todos os ramos possíveis, recolhendo o resultado de cada ramo. Escolhe-se a partir do método estatístico o melhor no final do processo;
2. "Puro": desconsidera o parâmetro atual, podando a árvore. Remonte-se a mesma sem este parâmetro (a partir do nó atual).

Para exemplificarmos a diferença existente entre cada um dos métodos, utilizou-se a Figura 5 como entrada no programa. Foram habilitados alguns alfabetos/fontes previamente existentes no banco de dados do programa (Arial, Símbolos e Grego).

Optical character recognition, usually abbreviated to OCR, is the mechanical or electronic translation of images of handwritten, typewritten or printed text (usually captured by a scanner) into machine-editable text. It is used to convert paper books and documents into electronic files, for instance, to computerize an old record-keeping system in an office, or to serve on a website.

Fig. 5. Arquivo utilizado para realizar o teste.

Optical character recognition, usually abbreviated to OCR, is
the mechan ical o r electron ic translation of images of handwritten , typewritten o r
printed text (usually captured by a scanner] into machine-editable text- It is used
to convert paper books and documents into electronic files, for instance, to
computerize an old record-keeping system in an office, or to serve on a website-

Fig. 6. Arquivo de saída do teste.

O resultado pode ser visto na Figura 6 e os parâmetros relativos a esses testes podem ser observados na Tabela 1, em que estão presentes os valores de tempo de execução e a taxa de acerto para cada um dos algoritmos. Esses são os valores finais da primeira versão software, concluído em dezembro de 2009. Pode-se observar a diferença existente nos resultados obtidos para cada um dos métodos.

TABELA I
COMPARAÇÃO ENTRE OS 4 MÉTODOS DISPONÍVEIS NO PROGRAMA

Método	Tempo (s)	Taxa de acerto (%)
Estatístico	98	84,7
Simples	9	19,5
Adaptativo 1	34	94,5
Adaptativo 2	162	93,6

III. CONCLUSÕES

Podemos observar através dos resultados obtidos pela execução do arquivo da Figura 5 que cada um dos métodos obteve um tempo de resposta e uma taxa de acerto diferente. Levando-se em consideração a taxa de acerto, temos que os métodos adaptativos proporcionaram uma melhor taxa de acerto que o método de árvore de decisão simples e que o método estatístico. Entretanto, pode-se observar que o algoritmo de árvore de decisão simples apresenta um menor tempo de processamento que todos os outros métodos. Um fato importante de ser ressaltado é o fato de que apenas as comparações entre os métodos que utilizam a mesma tecnologia são válidas. Deste modo, as comparações devem ser feitas entre os métodos Adaptativo 1, Adaptativo 2 e Simples. O método estatístico foi inserido devido ao fato de ser o mais utilizado pelos OCR's.

VI. Este trabalho proporcionou aos alunos e à comunidade científica uma maior aproximação com os verdadeiros problemas concernentes ao reconhecimento óptico de caracteres. Este é um dos temas mais antigos e mais estudados dentro da área de reconhecimento de padrões. Acreditamos que a utilização dos conceitos de adaptatividade adquiridos ao

longo do desenvolvimento do projeto ajudou de forma significativa na criação de um software único no mercado, uma vez que o mesmo permite ao usuário o controle da maioria das funcionalidades mais importantes do programa, como a inserção de novos alfabetos/fontes de modo dinâmico, a correção dos parâmetros dos caracteres e o treinamento do OCR.

Este foi o primeiro trabalho desenvolvido no Laboratório de Técnicas Adaptativas na área de reconhecimento óptico de caracteres com a utilização de técnicas adaptativas. Portanto, no que diz respeito ao futuro deste projeto, pode-se afirmar que diversas funcionalidades adicionais poderiam se desenvolvidas, assim como melhorias em processos específicos do programa. Podemos citar as seguintes áreas que poderiam receber contribuições e avanços futuramente:

1. Acredita-se que poderiam ser desenvolvidos novos parâmetros para realizar a escolha dos caracteres. Alguns desses parâmetros haviam sido planejados, mas não foram implementados, como o ângulo de inclinação e o número de pontas presentes nos símbolos;

2. Novos métodos para melhorar o processamento de imagem poderiam ser implementados, tendo em vista que o maior problema encontrado nas imagens durante o desenvolvimento deste trabalho foi a presença de imperfeições e falhas nos caracteres, principalmente quando as imagens eram geradas por scanners;

3. Modificações nos algoritmos adaptativos podem ser realizadas, melhorando a montagem das árvores de decisão, os tempos de processamento e as taxas de acerto.

IV. AGRADECIMENTOS

Os autores reconhecem as contribuições de J. N. João para a o desenvolvimento deste trabalho.

V. REFERENCIAS

[1] N. Morton, P. S. Eric, Pattern Recognition Engineering, New York.

[2] P, Hemerson, N. J. João, "Tecnologia Adaptativa em Engenharia de Computação: Estado da Arte e Aplicações.", Tese de doutorado, Universidade de São Paulo, Dec. 2003.

[3] P, Hemerson, N. J. João, "AdapTree - Proposta de um Algoritmo para Indução de Árvores de Decisão Baseado em Técnicas Adaptativas", In: Conferência Latino Americana de Informática, 2002, Montevideo.

[4] T.F. William, "Computer Vision for Interactive Computer Graphics", , IEEE Computer Graphics and Applications, Vol. 18, Issue 3, pp. 42-53, May-June 1998.

[5] J. N. João, "Adaptatividade e Tecnologia Adaptativa", Tutorial baseado no WTA 2007. Revista IEEE América Latina. Vol. 5, Num. 7, ISSN: 1548-0992, Novembro 2007. (p. 495).

[6] J. SCHULENBURG. Gocr. Disponível em: <http://jocr.sourceforge.net/>. Acesso em: 10 de abril de 2009, 2009.

[7] OCRAD. Ocrad. Disponível em: <http://www.gnu.org/software/ocrad/ocrad.html/>. Acesso em: 10 abril 2009, 2009.

[8] OCROPUS. Ocropus. Disponível em: <http://code.google.com/p/ocropus/>. Acesso em: 10 abril 2009, 2009.

[9] T.; MEZHIROV I. SMITH, R.; BREUEL. Tesseract. Disponível em: <http://code.google.com/p/tesseract-ocr/>. Acesso em: 10 de abril de 2009, 2009.

Bruno Tadayoshi Mendes Doy nasceu no Brasil, na cidade de São Paulo (SP) no dia 03 de setembro de 1985. Cursa atualmente o curso de Engenharia Elétrica com Ênfase em Computação na Escola Politécnica da Universidade de São Paulo situada na cidade de São Paulo (SP), Brasil.

Realizou durante 3 anos pesquisas na área de eficiência energética e otimização de consume residenciais e empresariais. Trabalha atualmente em uma empresa de desenvolvimento de softwares.

Douglas Fernandes de Souza nasceu no Brasil, na cidade de São Paulo (SP) no dia 13 de outubro de 1986. Cursa atualmente o curso de Engenharia Elétrica com Ênfase em Computação na Escola Politécnica da Universidade de São Paulo.

Artigos Publicados:

Software de simulação e visualização 3D da corrosão anisotrópica do Silício - D.F. de Souza e M.N.P. Carreño; XVI Simpósio Internacional de Iniciação Científica da USP; Engenharias e Exatas (2008) .

3D Simulation Software for Visualization of MEMS Microfabrication Processes - F.B. Colombo, P.M. Hokama, F. Tsuda, R.G. Jankauskas, D.F. de Souza, P. Kayatt and M.N.P. Carreño; ECS Transactions : Microelectonics Technology and Devices - SBMicro 2008; 14 No 1 (2008) 99-108.

Software CAD em C++ para Projeto de MEMS e MOEMS - D.F. de Souza, R.G. Jankauskas e M.N.P. Carreño; XV Simpósio Internacional de Iniciação Científica da USP - Engenharias e Exatas; Engenharias e Exatas (2007).

Rafael Garib Jankauskas nasceu no Brasil, na cidade de São Paulo (SP) no dia 28 de julho de 1987. Cursa atualmente o curso de Engenharia Elétrica com Ênfase em Computação na Escola Politécnica da Universidade de São Paulo situada na cidade de São Paulo (SP), Brasil.

Artigos Publicados:

Ferramentas de visualização dinâmica para software de visualização e simulação de processos de microfabricação - R.G. Jankauskas e M.N.P. Carreño; XVI Simpósio Internacional de Iniciação Científica da USP; Engenharias e Exatas (2008).

3D Simulation Software for Visualization of MEMS Microfabrication Processes - F.B. Colombo, P.M. Hokama, F. Tsuda, R.G. Jankauskas, D.F. de Souza, P. Kayatt and M.N.P. Carreño; ECS Transactions : Microelectonics Technology and Devices - SBMicro 2008; 14 No 1 (2008) 99-108.

Software CAD em C++ para Projeto de MEMS e MOEMS - D.F. de Souza, R.G. Jankauskas e M.N.P. Carreño; XV Simpósio Internacional de Iniciação Científica da USP - Engenharias e Exatas; Engenharias e Exatas (2007).