# Posting Paper on the Web

William A. Barrett and Rex Barzee
Department of Computer Science
Brigham Young University
Provo, Utah 84604
barrett@cs.byu.edu

## Abstract

We present a document processing system that accepts scanned images of paper documents as input and outputs hyperlinked electronic documents. The system segments document images, separating text from graphics, recognizes text, and creates hypertext links between document components (text, images, graphics).

By (1) limiting input to popular Times-Roman and Helvetica fonts found in first-generation scans of columnated magazines and tabloids, and using (2) gray scale attributes, (3) multiple character prototypes to recognize kerned and touching characters, (4) a lexicon to find and correct recognition errors, and (5) providing user interaction to recognize problem words, we achieve OCR accuracies up to 99.8%. This compares closely to our measurements of human proofreading accuracy (99.94%) which, however, takes six times longer.

A simple method for automated selection of important words in a document and creation of hypertext links from those words to other document components is developed to provide high-level searching and browsing of the document. Browsing granularity (i.e. the number and types of linked words) is user-selectable. The appropriateness of automatically selected link anchors is comparable to human performance.

## 1. Introduction

With growing interest in the Internet and the world wide web of information, there is a commensurate interest in technology for creation and access of digital libraries containing huge corpora of information. While much effort is focused on incorporation of electronic documents into digital libraries, much work is still needed to capture the content and make accessible the extant body of information that currently exists only in printed form. Full integration of existing paper based documents requires converting raw scanned documents to an electronically accessible and browseable format. This paper describes a document recognition system (Idoc) that automatically converts raw document images from popular, columnated magazine formats containing common Times-Roman and Helvetica fonts into electronically accessible and linked (HTML) documents.

Idoc uses a two pass algorithm to segment document images into document components. It incorporates multiple character prototypes, grayscale character attributes, lexicon searching, and word level recognition to perform text recognition (OCR). It also integrates on-line user supplied feedback to improve the recognition accuracy. Idoc uses the number of occurrences and capitalization of words to extract, from the text of a document, important words to be used as link anchors. Idoc uses full text indexing and the cosine measure to create hypertext links from the anchors to other document components.

### 1.1 Document Segmentation

Following previous work in segmentation of text from background, we make use of both global and local thresholding techniques [1,2] to provide robustness and prevent broken and touching characters. (Errors in character segmentation (broken and touching characters) account for as much as 79% of character recognition errors in commercial OCR systems [6].) Since many documents are not X-Y separable [3, 4], we make use of connected component algorithms to segment documents into textual, graphical and image components [5].

After segmentation, document components must be classified as text, graphics, or images so that appropriate recognition, linking, and/or storage techniques may be applied to each component. Classification has been done using a wide range of techniques including size metrics [7], histogram distribution [8], and texture [9]. Syntactic techniques and associated page grammars have been used to exploit publication formats in order to simultaneously segment and classify document components [4, 10, 11]. Knowledge based component merging approaches have also been applied to component segmentation and classification [10, 12]. In addition, classification and separation of text strings from mixed text/graphic images has been done using knowledge-based heuristics and collinear (Hough) grouping techniques [13].

## 1.2 Component Recognition

Document component recognition commenced with the relatively (compared with arbitrary images and graphics) civilized problem of text recognition (OCR) [14]. Early work with template matching was followed with more robust shape or feature based techniques [15, 16, 17]. More recently the importance of preserving gray scale information has been recognized [18, 19]. Lexical and contextual knowledge [20] has been shown by many to further improve the accuracy of recognition algorithms. One extensive study [21] demonstrated that the best commercial software achieved only 98.6% accuracy, which means that substantial human post editing is still required and that OCR is still not a completely solved problem.

Although recognizing the content of graphical document components (line drawings and diagrams) is a long standing problem within the field of computer vision, techniques for recognizing lines [5], symbols [22], characters [13] and shapes [23] have been developed. Within digital images, algorithms and systems exist for recognizing lines, circles, other shapes [5] , and characters [24]. However, recognition of arbitrary image content generally requires incorporation of domain-specific knowledge [25] and this is far from a solved problem.

## 1.3 Component Linking

Document component linking includes indexing and cross referencing document components, notably text, thus allowing for electronic access to a document through word and string searching and browsing. Although some commercial systems perform automatic full text indexing on whole libraries of electronic documents, hypertext links which connect related text and other components are typically created by humans, which is often tedious and expensive.

The Hypermedia Research Authoring Toolkit (HART) [26] is used to assist users in creating hypertext documents from paper based documents, thus lowering the cost for a paper based to hypermedia conversion. HART suggests likely key phrases, anchors, nodes, and links to a user.

# 2. Paper in - HTML out

The approach described in this paper (Idoc) accepts full color images of paper as input and outputs hyperlinked HTML documents. Idoc was designed to execute these tasks on articles from several basic popular magazine formats, containing Times-Roman and Helvetica fonts and to create links for important individual words. Thus it is not intended that Idoc correctly analyze *all* document formats or perform *omni-font* character recognition; rather, the intent here is to link, and make electronically accessible everyday textual tabloid information with sufficient accuracy to permit *posting paper on the web*.

To accomplish the paper-to-HTML conversion, Idoc processes document images in four main steps. In step 1, shown in figure 1, a document image is segmented and parsed as explained in section 2.1, producing graphic images and character images. In step 2, the character images are recognized (section 2.2), producing a text document. In step 3 text documents are indexed, creating a full text index (section 3). Finally in step 4, a user request for a recognized document is received by Idoc, and using the graphic images produced in step 1, the text documents produced in step 2, and the index produced in step 3, Idoc creates an HTML document and serves it to the user.

## 2.1 Segmenting and Parsing Images

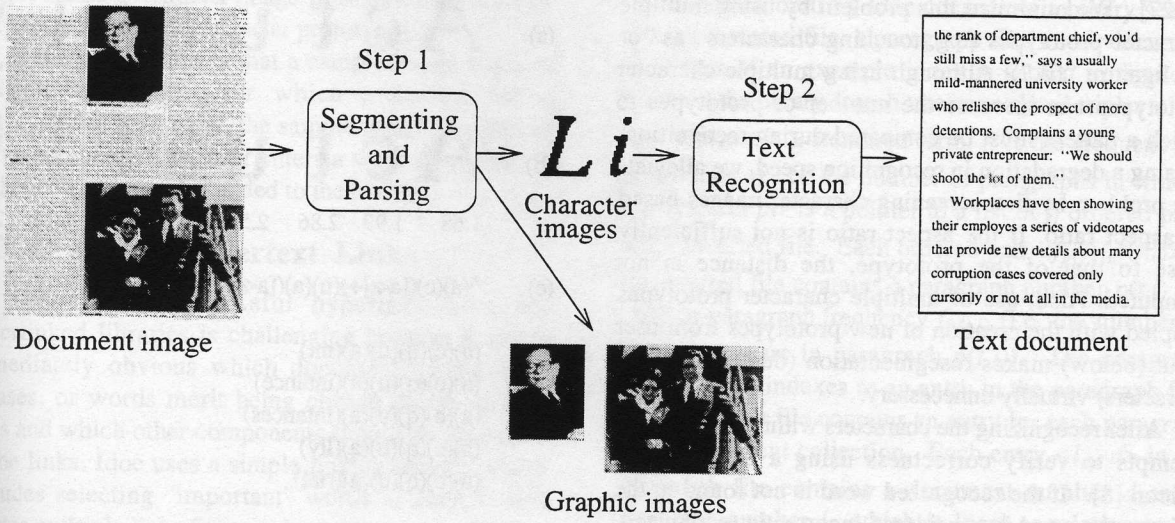
Idoc first creates a gray level version and a binary version of the original full color document image. The gray level image is obtained from the color image using the standard NTSC RGB to luma conversion (namely $Y = 0.299R + 0.587G + 0.114B$). The binary image is obtained by thresholding the grayscale image globally, labeling all connected components within the thresholded image, then locally re-thresholding the area containing and immediately surrounding each connected component using techniques described in [1].

A two-pass connected component algorithm similar to the bottom-up approach investigated by Nagy and Krishnamoorthy et al [3, 4] was developed to parse document images. We use connected component labeling instead of X-Y parsing to allow separation of kerned characters and to maintain greater consistency in the parsing order for subsequent merging. We merge connected components using ad hoc rules (instead of grammars) designed for standard columnated articles from the popular magazines.

## 2.2 Text Recognition

We make use of a minimum distance classifier in conjunction with grayscale projection profiles and binary indentation profiles to perform OCR. To enhance robustness, we also incorporate multiple character prototypes to recognize kerned and touching characters, use of a lexicon to recognize problem characters based on word-level context, and user supplied feedback to improve present and future recognition. Due to a finer level of quantization, grayscale profiles were found to match prototypes more closely than binary profiles.

The majority of OCR errors are caused by missegmented (broken, kerned, and touching) characters



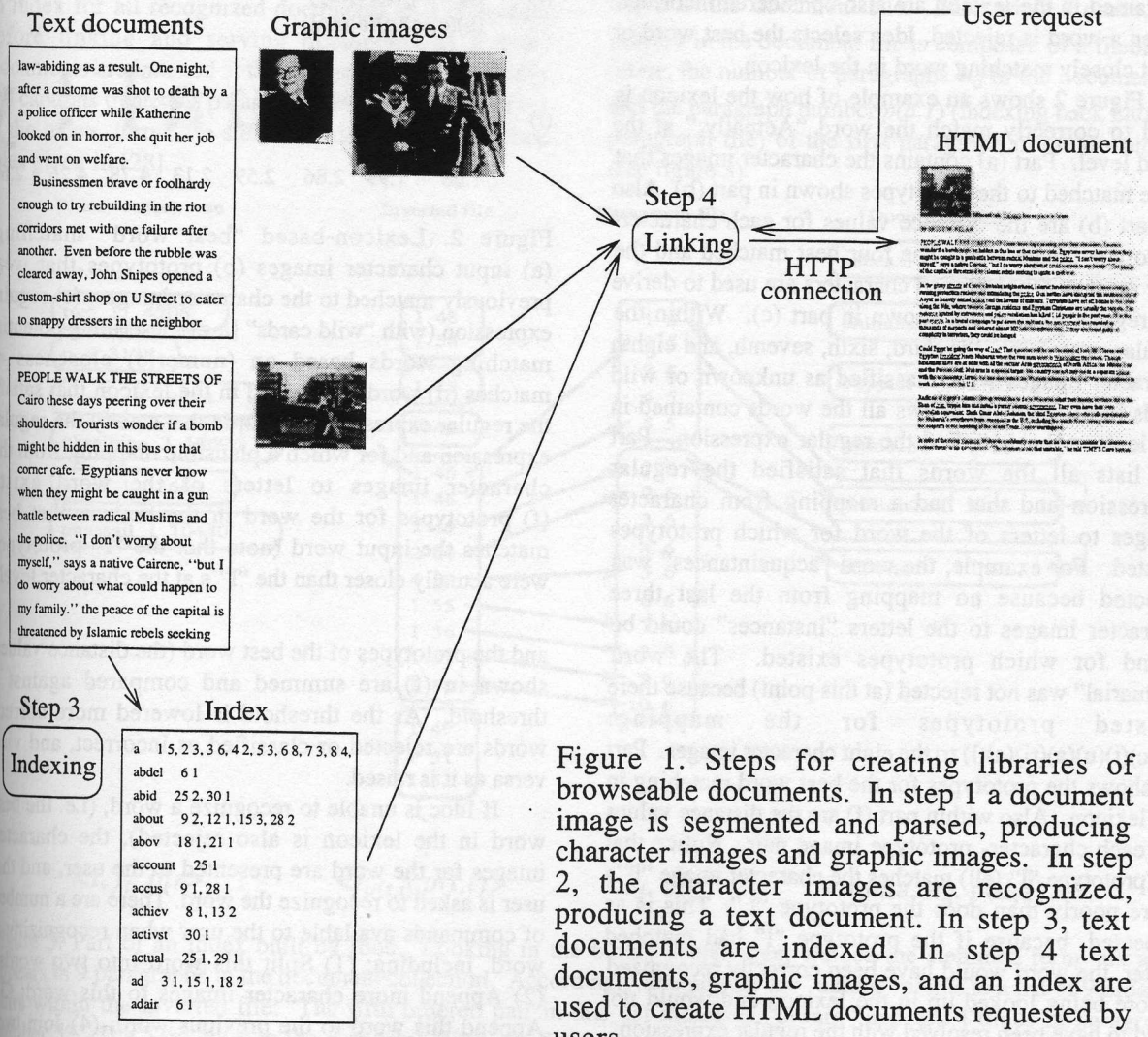




Figure 1. Steps for creating libraries of browseable documents. In step 1 a document image is segmented and parsed, producing character images and graphic images. In step 2, the character images are recognized, producing a text document. In step 3, text documents are indexed. In step 4 text documents, graphic images, and an index are used to create HTML documents requested by users.

[6, 27]. We minimize this problem by using multiple character prototypes (e.g. touching characters "as" or the ligature "ffi."). Although using multiple character prototypes can increase the number of prototypes to which a pattern must be compared during recognition, causing a degradation in recognition speed, we alleviate this problem by first screening character images based on aspect ratio. If the aspect ratio is not sufficiently close to that of the prototype, the distance is not computed. Also, use of multiple character prototypes coupled with the creation of new prototypes from user input (below) makes resegmentation (due to touching characters) virtually unnecessary.

After recognizing the characters within a word, Idoc attempts to verify correctness using a 25,000 word lexicon [8]. If the recognized word is not found in the lexicon, the word is considered incorrectly recognized. This means correctly recognized words that are not contained in the lexicon are also considered incorrect. When a word is rejected, Idoc selects the best word or most closely matching word in the lexicon.

Figure 2 shows an example of how the lexicon is used to correctly match the word "Actually," at the word level. Part (a) contains the character images that were matched to the prototypes shown in part (b). Also in part (b) are the distance values for each character-prototype image pair. The four best matched and the four most poorly matched characters are used to derive the regular expression shown in part (c). Within the regular expression, the third, sixth, seventh, and eighth character images were classified as unknown or wild cards [a-z]. Part (d) shows all the words contained in the lexicon that satisfied the regular expression. Part (e) lists all the words that satisfied the regular expression and that had a mapping from character images to letters of the word for which prototypes existed. For example, the word "acquaintances" was rejected because no mapping from the last three character images to the letters "instances" could be found for which prototypes existed. The word "actuarial" was not rejected (at this point) because there existed prototypes for the mapping: (a)(c)(t)(u)(a)(ri)(a)(l) to the eight character images. Part (f) shows the prototypes for the best word matching in the lexicon. Also within part (f) are the distance values for each character- prototype image pair. Notice that the prototype "l" (ell) matches the character image "I"'s more poorly than does the prototype "I." This is as expected, because if the prototype "l" had matched better, the word would have been correctly recognized before being looked up in the lexicon and would not need to have been resolved with the regular expression.

To determine if the word returned is indeed the correct word, the distances between the character images

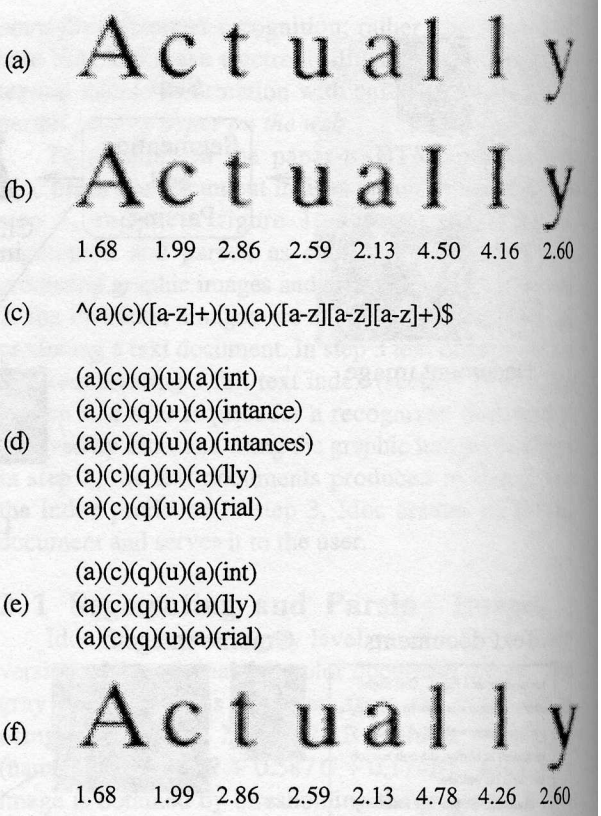



Figure 2. Lexicon-based "best word" matching: (a) input character images (b) prototypes that were previously matched to the character images. (c) regular expression (with "wild cards" [a-z]) describing potential matching words based on (numeric) closeness of matches (d) words contained in the lexicon that satisfy the regular expression (e) words that satisfy the regular expression and for which a plausible mapping from the character images to letters of the word exists (f) prototypes for the word in the lexicon that best matches the input word (note that the "I" prototypes were actually closer than the "l"'s at the character level.

and the prototypes of the best word (the distance values shown in (f) are summed and compared against a threshold. As the threshold is lowered more correct words are rejected or classified as incorrect, and vice versa as it is raised.

If Idoc is unable to recognize a word, (i.e. the best word in the lexicon is also rejected), the character images for the word are presented to the user, and the user is asked to recognize the word. There are a number of commands available to the user when recognizing a word, including: (1) Split this word into two words; (2) Append more character images to this word; (3) Append this word to the previous word; (4) Join two character images (i.e. make one character from two broken parts); (5) Delete a character image that is noise;

and (6) Name a character image with multiple characters (i.e. create a multiple character prototype).

If the user indicates that a sample image contains multiple characters, for which a corresponding prototype does not exist, the sample image is added to the prototypes. If the user enters a word that is not in the lexicon, the word is added to the lexicon.

## 2.3 Creating Hypertext Links

The creation of useful hypertext links and hyperlinked libraries is challenging because it is not immediately obvious which document components, phrases, or words merit being chosen as anchors for links and which other components should be the targets of the links. Idoc uses a simple linking scheme which includes selecting "important" words as anchors and creates multiple links from each anchor.

To facilitate creation of hypertext links, Idoc builds an index for all recognized documents in a collection before linking and serving documents to a user. Documents are indexed at the paragraph level with titles and captions treated as paragraphs. Before being inserted in the index, terms are stemmed using a set of ad hoc rules given in [28].

An index built by Idoc consists of a binary search tree, an inverted file, a paragraph file, and a document file. Part of an example index is shown in figure 3. Nodes in the binary search tree are ordered triples $<t, f_t, ptr>$ where $t$ is a stemmed term, $f_t$ is the frequency of appearance of $t$ or the number of paragraphs in which $t$ appears, and $ptr$ is a pointer to a list of $f_t$ ordered pairs in the inverted file. Each ordered pair $<f_{p(t,i)}, p(t,i)>$ in the inverted file contains a paragraph number $p(t,i)$ and a within-paragraph frequency $f_{p(t,i)}$ (i.e. the number of times $t$ appears in paragraph $p(t,i)$). The paragraph number $p(t,i)$ indexes to an entry in the paragraph file. The paragraph file contains an entry for each paragraph in the document collection. Each entry $<l_d, d>$ in the paragraph file contains a document number $d$ and a paragraph number $l_d$ which is local or relative to the document $d$. The document number $d$ indexes to a triple in the document file. Each triple $<name, n_d, p(d,1)>$ in the document file is composed of a filename *name*, the number of paragraphs $n_d$ in the document $d$, and the paragraph number $p(d,1)$ (indexing back into the paragraph file) of the first paragraph in the document (see figure 3).

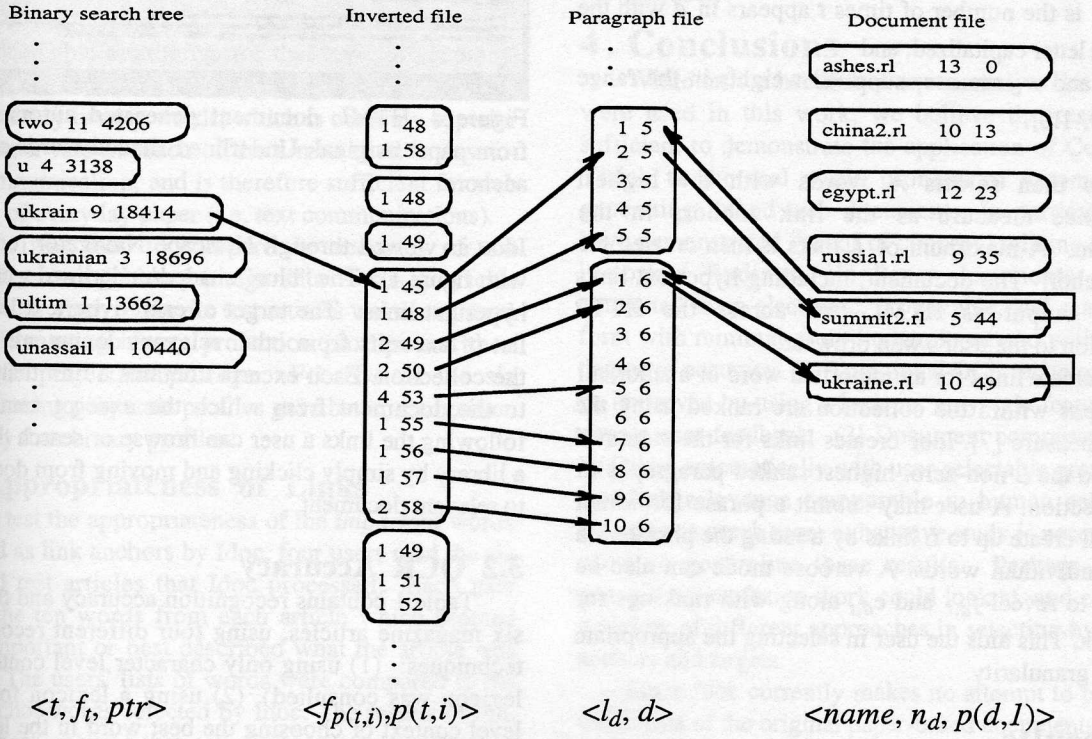



Figure 3. Part of an index built by Idoc. Looking in the binary search tree, we see the stemmed term "ukrain" appears in 9 paragraphs in the document collection. Accordingly, the pointer for "ukrain" points to a list of 9 ordered pairs within the inverted file. The fifth ordered pair in this list indicates "ukrain" appears 4 times in paragraph number 53. The 53rd entry in the paragraph file is the 5th paragraph in document number 6, and document 6 is the file *ukraine.rl*. which has 10 paragraphs, the first of which is 49th in the paragraph file.

There are several parameters that a user may modify to control the links that Idoc creates which govern the *granularity* with which the document may be browsed. A user enters the parameters in a fill-out form that Idoc places at the top of all HTML documents served to a user. The user may select verbose mode; may enter the frequency weight $w_f$, capitalized weight $w_c$, number of link anchors $A$ selected per document, and maximum number of links $L$ for each anchor; and may define phrases for which Idoc will create links.

When an HTTP request for a (recognized) document is received, Idoc loads the index for the library or collection of documents containing the requested document and an AVL (binary search) tree from the requested document which is used to calculate an importance measure for each word that is not a stop word (stop words = a, about, after, almost, . . . as . . . for, . . . etc. ).

$$importance = f_{d,t} * w_f + c_{d,t} * w_c \qquad [1]$$

where

$f_{d,t}$ is the number of times stemmed term $t$ appears in the requested document $d$

$c_{d,t}$ is the number of times $t$ appears in $d$ with the first letter capitalized; and

$w_f$ and $w_c$ are user supplied weights in the range [0.0, 1.0].

Idoc then selects $A$ words with the highest importance measure as the link anchors in the document. A maximum of $L$ links is then created for each anchor. The document, including hypertext links is then output as HTML text across the HTTP connection to the user's web browser.

To create links for an important word or anchor, all paragraphs within the collection are ranked using the cosine measure [7]. Idoc creates links for the stemmed word $t$ to the $L$ non-zero, highest ranked paragraphs in the collection. A user may submit a phrase for which Idoc will create up to $L$ links by treating the phrase as a list of individual words. A verbose mode can also be toggled to reveal $f_{d,t}$ and $c_{d,t}$ along with rankings for each link. This aids the user in selecting the appropriate level of granularity.

# 3. Results

## 3.1 HTML Out

Using Idoc, a collection of browseable documents has been built from images of magazine articles. Figure 4 shows one such HTML document created by

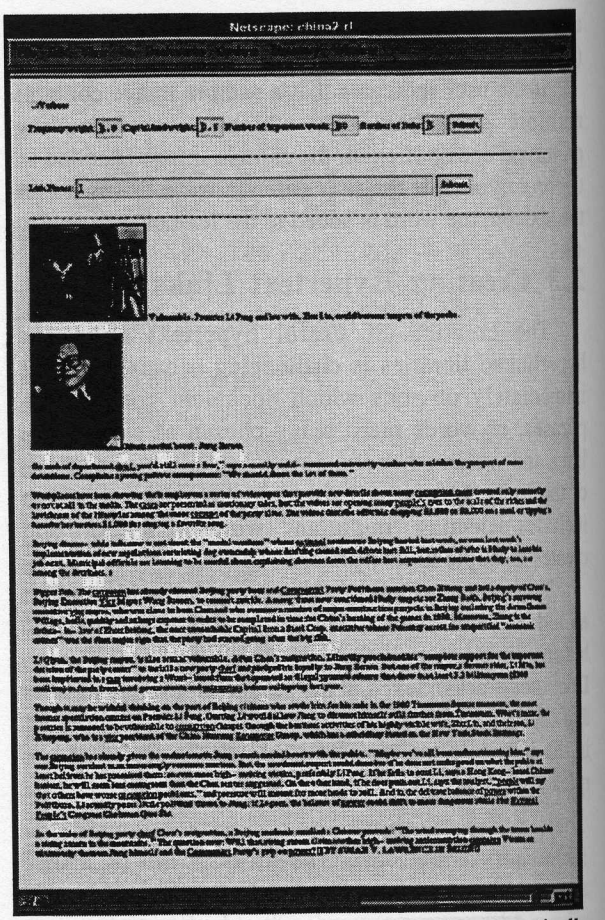



Figure 4. HTML document generated automatically from paper original. Underlined shaded words are link anchors.

Idoc as viewed through Netscape Navigator (compare with figure 1). The "blue" shaded/underlined words are hypertext links. The target of each hypertext link is a list of excerpts from other relevant documents within the collection. Each excerpt contains a link that points to the document from which the excerpt came. By following the links a user can browse or search through a library by simply clicking and moving from document to relevant document.

## 3.2 OCR Accuracy

Table 1 contains recognition accuracy and time for six magazine articles, using four different recognition techniques: (1) using only character level context (no lexicon was consulted), (2) using a lexicon for word level context or choosing the best word in the lexicon, (3) using word level context and incorporating user supplied feedback, and (4) output of Idoc edited or proofread by a human. The required human input was given by a user familiar with the recognizer and text editor but unfamiliar with the articles.

As can be seen in Table 1, using word level

| Test Document | Character Context | | | Word Context | | | User Feedback | | | Human Proofreading | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | # Errors | % Correct | Time sec | # Errors | % Correct | Time sec | # Errors | % Correct | Time sec | # Errors | % Correct | Time min:sec |
| "ashes" | 11 | 99.79 | 19 | 11 | 99.79 | 20 | 6 | 99.88 | 49 | 5 | 99.90 | 7:35 |
| "china" | 7 | 99.81 | 39 | 7 | 99.81 | 40 | 5 | 99.87 | 55 | 1 | 99.97 | 6:40 |
| "egypt" | 11 | 99.74 | 16 | 10 | 99.77 | 17 | 5 | 99.88 | 88 | 2 | 99.95 | 8:45 |
| "russia" | 18 | 99.42 | 32 | 15 | 99.52 | 35 | 9 | 99.71 | 142 | 1 | 99.97 | 6:58 |
| "summit" | 14 | 99.32 | 21 | 11 | 99.47 | 23 | 12 | 99.42 | 61 | 2 | 99.90 | 7:20 |
| "ukraine" | 10 | 99.72 | 38 | 8 | 99.78 | 39 | 7 | 99.81 | 50 | 2 | 99.94 | 7:18 |
| average | 11.8 | 99.68 | 27.5 | 10.3 | 99.72 | 29.0 | 7.3 | 99.80 | 74.2 | 2.2 | 99.94 | 7:26 |

Table 1. Recognition accuracy and time for six different magazine articles. Each article was recognized four times: (1) using only character level context, (2) using a lexicon for word level context, (3) using word level context and user supplied feedback, and (4) Idoc output after human editing. Note - for article "summit," number of errors increased from Word Context to User Feedback because user made several mistakes giving feedback.

context gave only a slight improvement over character level. The improvement is small because of the difficulty in choosing a threshold at which the best word in the lexicon is accepted or rejected. Asking a user to recognize problem text generally reduced the number of errors by half over the character and word context methods, but it also more than doubled recognition time. This could be reduced by performing other recognition tasks in the background. Having a human post edit recognized text gave the best accuracy but with over a factor of six increase in time making it an unacceptable alternative for that level of accuracy. Furthermore, 99.8% accuracy represents roughly one error in every twenty words, which is certainly superior to the error rate in much of the e-mail used in day-to-day communication, and is therefore sufficient for web posting of everyday paper (i.e. text communications).

Idoc was used to compare the accuracy of text recognition using binary image data versus grayscale image data. Each of the six test articles was recognized twice, first using binary projection profiles and then grayscale projection profiles. For five of the six articles, using grayscale profiles gave better recognition accuracy than binary profiles.

### 3.3 Appropriateness of Links

To test the appropriateness of the *important* words selected as link anchors by Idoc, four users read the six original test articles that Idoc processed. Each user chose the ten words from each article which seemed most important or best described what the article was about. The users' lists of words were compared to the link anchor words selected by Idoc. The results of this comparison showed that any given user would choose, on average, 3 of the 10 words selected by Idoc as among the 10 most important words in the article. As a baseline, the same measure was computed for each user compared to the other three users. The average measure for all users was 28.7%, and the highest measure for any user was 35.6%. Thus, based on this small set of documents, Idoc's selection of link anchors is comparable to human performance.

As follow-on work, it would be of interest to expand the number and scope of the documents included in this kind of analysis in order to see how well these results generalize. However, we believe that these preliminary results are encouraging and demonstrate that meaningful and appropriate links can be generated automatically.

## 4. Conclusions

While only a handful of representative documents were used in this work, we believe the results are sufficient to demonstrate the application of Computer Vision to the real world problem of automatically generating linked web documents. In particular, we have demonstrated "proof of concept" relating to three important findings: (1) Paper documents can be converted to an electronic, browseable, and searchable form with minimal or no human input and with a high degree of accuracy. (2) Traditional OCR techniques can be improved by using a lexicon, grayscale features and simple user feedback. (3) Document components can be linked automatically with user-selectable granularity and link relevance comparable to human selection. However, a much more exhaustive study is necessary in order to generalize these results. Perhaps in that generalization, future work could look at, and compare a variety of different approaches in selecting hyperlink anchors and targets.

Since Idoc currently makes no attempt to preserve the format of the original paper-based documents, future work could be aimed at creating hyperlinked documents that look precisely like the original in format and layout - perhaps by exploiting new web publishing languages such as XML. In connection with this, the document image parser could be trained on different document formats. Each document image could be

matched to one of the prototypes, and the parser could use a grammar or some other method specifically geared to the prototype to parse the document image. This would allow Idoc to parse a broader range of document images, and, in the process, preserve and post the document in its original published format.

Recognition could also be performed at the word level by default, with the recognizer resorting to character level recognition only in difficult cases. This could be done by collapsing all intra-word spaces, causing all letters within a word to touch, then using a measurement such as the aspect ratio of the collapsed word to choose candidate words from a lexicon, generating collapsed prototypes for each candidate word, and matching the collapsed test word to one of the candidate prototypes.

**Bibliography**


[1] N. Otsu, "A threshold selection method from gray level histograms," IEEE Transactions on Systems, Man, and Cybernetics, SMC-9(1):62--66, Jan 1979.

[2] M. Kamel and A. Zhao, "Extraction of binary character/graphics images from grayscale document images," CVGIP: Graphical Models and Image Processing, 55(3):203--217, May 1993.

[3] G. Nagy and S. Seth, "Hierarchical representation of optically scanned documents," Proceedings of the International Conference on Pattern Recognition}, 347--349. Computer Society Press, 1984.

[4] M. Krishnamoorthy, G. Nagy, S. Seth, and M. Viswanathan, "Syntactic segmentation and labeling of digitized pages from technical journals," IEEE Transactions on Pattern Analysis and Machine Intelligence, 15:737--747, August 1992.

[5] R. M. Haralick and L. G. Shapiro, "Computer and Robot Vision," chapter 11, 555--638, Addison-Wesley Publishing Company, Inc., 1992.

[6] T.A. Nartker, J. Kanai, and S.V. Rice, "A preliminary report on ocr problems in lss document conversion," November 1991. Presented at the Nuclear Waste Management Conference, Las Vegas, NV, April 1992.

[7] I.H. Witten, A. Moffat, and T.C. Bell, "Managing Gigabytes," Chapter 8, pages 295--327. Van Nostrand Reinhold, 115 Fifth Avenue, New York, NY, 1994.

[8] M. Zhuang, "Toward intelligent document understanding," MS Thesis, Brigham Young University, 1992.

[9] D. Wang and S.N. Srihari, "Classification of newspaper image blocks using texture analysis," CVGIP 47:327--352, 1989.

[10]G. Nagy, J. Kanai, M. Krishnamoorthy, M. Thomas, and M. Viswanathan, "Two complementary techniques for digitized document analysis, "ACM Conference on Document Process Systems Santa Fe, NM, Dec 1988.

[11]M. Viswanathan and M. Krishnamoorthy, "A syntactic approach to document segmentation," *IAPR* Intl Workshop on Structural and Syntactic Pattern Recognition, Pont-a-Mousson, France, Sep 1988.

[12]T. Watanabe, Q. Luo, and N. Sugie, "Structure recognition methods for various types of documents," Machine Vision and Apns, 6(2-3):163--176, 1993.

[13]L.A. Fletcher and F. Kasturi, "A robust algorithm for text string separation from mixed text/graphics images," IEEE Transactions on Pattern Analysis and Machine Intelligence, 10(6):910--918, Nov 1988.

[14]S. Mori, C.Y. Suen, and K. Yamamoto, "Historical review of ocr research and development," Proceedings of the IEEE, 80(7):1029--1058, July 1992.

[15]M.T.Y. Lai and C.Y. Suen, "Automatic recognition of characters by fourier descriptors and boundary encoding," Pattern Rec, 14(1-6):383--393, 1981.

[16]R.L. Grimsdale, F.H. Sumner, C.J. Tunis, and T. Kilburn, "A system for the automatic recognition of patterns," Proc of the IEEE, 106:210--221, 1959.

[17]G. Nagy, S. Seth, K. Einspahr, and T. Meyer, "Efficient algorithms to decode substitution ciphers with application to ocr," Proceedings of the Intl Conf on Pattern Recognition, pages 352--355, 1986.

[18]L.Wang and T. Pavlidis, "Direct gray-scale extraction of features for character recognition," PAMI, 15(10):1053--1067, October 1993.

[19]J. Rocha, B. Sakoda, J. Zhou, and T. Pavlidis, "Deferred interpretation of grayscale saddle features for recognition of touching and broken characters," Proceedings of the SPIE - Document Recognition}, pages 342--350, 1994.

[20]Rainer Hoch and Thomas Kieninger, "On virtual partitioning of large dictionaries for contextual post-processing to improve character recognition, "International Journal of Pattern Recognition and Artificial Intelligence}, 10(4):273--289, 1996.

[21]G. Nagy, "At the frontiers of ocr," Proceedings of the IEEE, 80(7):1093--1100, July 1992.

[22]C. Dachet, "Automatic reading of technical drawings: Symbol extraction by geometrical and morphological methods," Proceedings Vision Interface, pages 127--136, May 1994.

[23]R. Kasturi, R. Raman, C. Cennubhotla, and L. O'Gorman, "An overview of techniques for graphics recognition," Structured Document Image Analysis, pages 285--324, Springer-Verlag, 1992.

[24]J. Ohya, A. Shio, and S. Akamatsu, "Recognizing characters in scene images," IEEE Transactions on Pattern Analysis and Machine Intelligence, 16(2):214--219, February 1994.

[25]V. Govindaraju and R. Srihari, "Automatic face identification from news photo databases," Advanced Imaging, pages 22--26, November 1990.

[26]J. Robertson, E. Merkus, and A. Ginige, "The hypermedia authoring research tookit (hart)," Proceedings of the ACM ECHT, pages 177--185, 1994.

[27]G. Nagy, "Teaching a computer to read," Proceedings of the International Conference on Pattern Recognition, pages 225--229, 1992.

[28]M.F. Porter, "An algorithm for suffix stripping." Program, 14(3):130--137, July 1980.