Journal of Zhejiang University SCIENCE
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

The Million Book Project at Bibliotheca Alexandrina

ELDAKAR Youssef[1], EL-GAZZAR Khalid[1], ADLY Noha[†1,2], NAGI Magdy[1,2]

(*[1]Bibliotheca Alexandrina, El Shatby 21526, Alexandria, Egypt*)

(*[2]Computer and Systems Engineering Department, Alexandria University, Alexandria, Egypt*)

[†]E-mail: Noha.Adly@bibalex.org

Received Aug. 5, 2005; revision accepted Sept. 10, 2005

Abstract: The Bibliotheca Alexandrina (BA) has been developing and putting to use a workflow for turning printed books into digital books as its contribution to the building of a Universal Digital Library. This workflow is a process consisting of multiple phases, namely, scanning, image processing, OCR, digital archiving, document encoding, and publishing. Over the past couple of years, the BA has defined procedures and special techniques for the scanning, processing, OCR and publishing, especially of Arabic books. This workflow has been automated, allowing the governance of the different phases and making possible the production of 18000 books so far. The BA has also designed and implemented a framework for the encoding of digital books that allows publishing as well as a software system for managing the creation, maintenance, and publishing of the overall digital repository.

Key words: Million Book Project (MBP), Digital books workflow, Digitization, Universal Digital Library, Scanning, Multi-lingual OCR, Digital publishing, Image-on-text, DjVu, PDF
doi:10.1631/jzus.2005.A1327 **Document code:** A **CLC number:** TP391

INTRODUCTION

The Bibliotheca Alexandrina (BA) has developed a workflow for turning printed books into digital books. The process starts with selection of books to be digitized, which is done mainly by BA's Library Service Department. Books metadata is entered into the Digital Lab database. Metadata entry is followed by three core phases: scanning phase, in which the digital copy is generated; processing phase, in which image enhancement is performed; and OCR phase, in which text is recognized from the processed images. At the BA, the procedures for scanning, processing, and OCR have been laid out with Arabic content in mind. After the OCR output is generated, the workflow splits into two branches: one for encoding digital books and the other for archiving on CDROM, tape, and hard drives. BA has developed a Universal Digital Book Encoder (UDBE) that generates image-on-text versions of digital books, namely, PDF and DjVu files, with support for Arabic as well as other languages. Fig.1 shows the digital books workflow.

SCANNING

The Minolta PS7000 scanner is used. Scanning settings can be set either from the control panel or by scanning software (ACDSee 4.0). The settings and parameters used in the scanning phase are described below.

Resolution

Books are scanned at 300 dpi. For Arabic books, it was found that 300 dpi is the resolution that gives the best OCR results. Actually, the specification of the Arabic OCR software used, Sakhr's Automatic Reader v6.0 Professional or v7.1 Gold, recommends using 300 dpi images. Arabic recognition engines for the Arabic OCR software are built based on 300 dpi fonts and hence images scanned at 300 dpi give better results for the Arabic OCR than images scanned at lower or higher resolutions, such as 600 dpi. Also, downgrading 600 dpi images by software to 300 dpi does not give good OCR results. In fact, the OCR quality of downgraded images is lower quality than

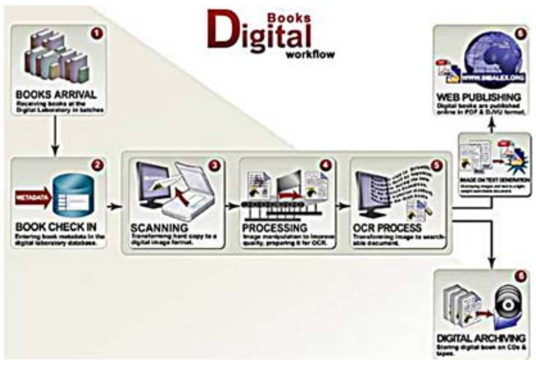


Fig.1 Digital books workflow

that of those scanned at 300 dpi. As for Latin books, the 300 dpi resolution was found to give good image quality and acceptable OCR results. Actually, 300 dpi is the resolution recommended by ABBYY's Fine-Reader for optimal performance.

As a leader of digitization projects in the Middle East, BA focuses on digitizing Arabic books. Of the 18000-book currently digitized collection, approximately 15000 books (83%) are Arabic books, and the rest of the collection encompasses a variety of Latin languages such as English, French, Italian, German, Russian, and so on.

Pages are scanned in either single or split page mode, and the selection of "text" or "photo" depends on the type of the original page.

Image format and compression

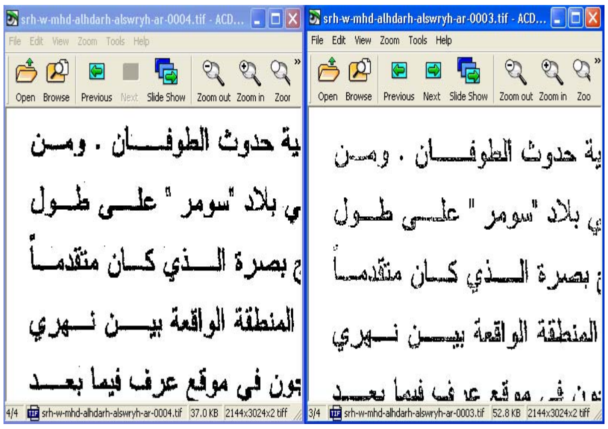
Tagged Image File Format (TIFF) and CCITT G4 compression are used. This combination ensures good compression rates for bitonal images without loss of detail.

Masking and center erase

The masking feature contains two sub-features: finger masking and centering. Center erase can be used with most of the books except with books having very close text to the center line. Setting masking and center erase features gives a neater look to the scanned pages and helps in reducing the amount of processing needed to be performed on the scanned pages.

Contrast

Contrast makes the characters look lighter or darker than how they appear in the original book. Adjusting contrast helps to fix the book printing simple errors, such as pixelized or dark characters. There are two types of contrast: black (shadow) contrast and white (highlight) contrast. When the black contrast is increased, letters appear darker. When the white contrast is increased, the light area of the paper appears whiter. Fig.2 shows a typical case in which black contrast is useful. However, the side effect of this increased black contrast is having more obvious speckles. The left image is scanned with higher shadow contrast than the right one causing filling of characters and some speckles. The contrast settings significantly affect the OCR performance.


srh-w-mhd-alhdarh-alswryh-ar-0004.tif - ACD...
File Edit View Zoom Tools Help
Open Browse Previous Next Slide Show Zoom out Zoom in Zoor
srh-w-mhd-alhdarh-alswryh-ar-0003.tif - ACD...
File Edit View Zoom Tools Help
Open Browse Previous Next Slide Show Zoom out Zoom in Zoo

ية حدوث الطوفـــان . ومـن
ي بلاد "سومر " علـى طـول
ج بصرة الــذي كــان متقدمــاً
المنطقة الواقعة بيــن نــهري
ون في موقع عرف فيما بعــد

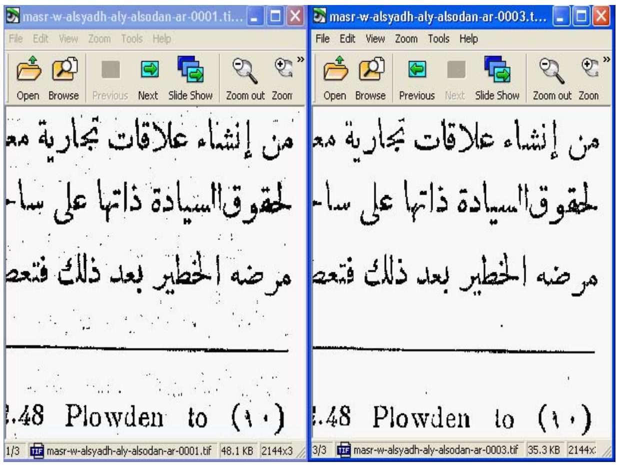
4/4 srh-w-mhd-alhdarh-alswryh-ar-0004.tif 37.0 KB 2144x3024x2 tiff

3/4 srh-w-mhd-alhdarh-alswryh-ar-0003.tif 52.8 KB 2144x3024x2 tiff

Fig.2 Effect of increasing shadow contrast

Higher values of highlight contrast are used if the book pages are yellow or if the page has a lot of speckles and noise. Fig.3 shows a yellow-paged book. Scanning this book with low values of white contrast shows a lot of speckles as shown in the left image of the figure. Increasing the value of white contrast helps in reducing the speckles as shown in the right image. Highlight contrast should be used carefully especially for Arabic books, otherwise the characters look very light and are not suitable for OCR.

Higher values of shadow contrast are used if the printing of the book characters is light or for old Arabic books with little gaps between characters. In such case, shadow contrast should be increased to make the characters look darker. The effect of increasing the shadow contrast is shown in Fig.4, where the left image is scanned with a low value of black contrast, whereas the shadow contrast is increased in the right image.


masr-w-alsyadh-aly-alsodan-ar-0001.ti...
File Edit View Zoom Tools Help
Open Browse Previous Next Slide Show Zoom out Zoom
1/3 masr-w-alsyadh-aly-alsodan-ar-0001.tif 48.1 KB 2144x3
masr-w-alsyadh-aly-alsodan-ar-0003.t...
File Edit View Zoom Tools Help
Open Browse Previous Next Slide Show Zoom out Zoom
3/3 masr-w-alsyadh-aly-alsodan-ar-0003.tif 35.3 KB 2144x

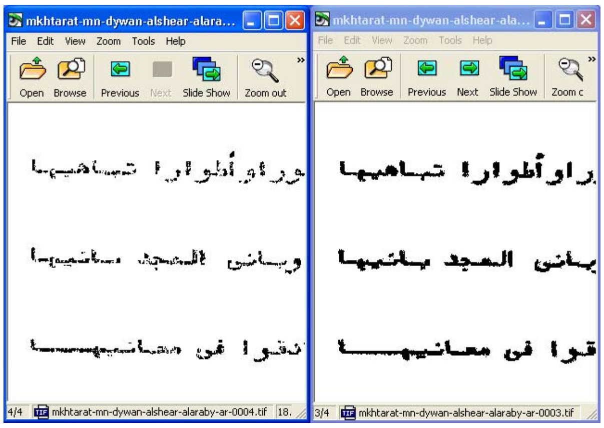
من إنشاء علاقات تجارية مع
لحقوق السيادة ذاتها على سا
مرضه الخطير بعد ذلك فتعط

.48 Plowden to (١٠)

من إنشاء علاقات تجارية مع
لحقوق السيادة ذاتها على سا
مرضه الخطير بعد ذلك فتعط

.48 Plowden to (١٠)

Fig.3 Effect of increasing highlight contrast


mkhtarat-mn-dywan-alshear-alara...
File Edit View Zoom Tools Help
Open Browse Previous Next Slide Show Zoom out
4/4 mkhtarat-mn-dywan-alshear-alaraby-ar-0004.tif 18.
mkhtarat-mn-dywan-alshear-ala...
File Edit View Zoom Tools Help
Open Browse Previous Next Slide Show Zoom c
3/4 mkhtarat-mn-dywan-alshear-alaraby-ar-0003.tif

وراو أطوارا تباهيها
وبانى المجد بانيها
تقوا فى معانيها

راو أطوارا تباهيها
بانى المجد بانيها
قوا فى معانيها

Fig.4 Effect of increasing shadow contrast on old books

Page count check

A simple check on the number of scanned pages is performed against the total number of pages in the printed book. If all the pages are scanned, the following formula must be valid: $T=L+U$, where T stands for TIFF file count, L stands for last page number, and U stands for count of unnumbered pages in the book. In addition, T must be an even number, since the front and back of each page are scanned.

PROCESSING

The purpose of the processing phase is to clean up and enhance the original images that come out from the scanner. Specifically, the processing phase has four objectives: black edge removal, centering, noise removal, and deskewing. Black edge removal and centering are performed in manual processing using Photoshop, while noise removal and deskewing are performed in automatic processing using ScanFix. A fully automatic processing procedure is desirable. However, experiment has shown that automatic cleanup features in ScanFix, such as margin, i.e. black edge, removal and intelligent cropping, are generally not quite reliable. For Arabic, in particular, some of these features can be detrimental, such as line removal, which can potentially erase the baseline of characters in this cursive script.

The first step to take place is noise removal. ScanFix's despeck parameter is experimented with using a couple of test images to determine the optimal setting for the book that removes as much noise as possible without damaging sensitive components of the text, such as dots and diacritics. Taking precaution not to damage sensitive components of the text is particularly important in Arabic, which makes extensive use of dots. The despeck parameter is typically a value ranging from 3 to 5 pixels. A batch is then run, whereby despecked copies of original files are transferred to an intermediate directory.

Images in the intermediate directory are then manually edited in Photoshop. Here, a selection rectangle is drawn tightly around the actual content, cut into the clipboard, the page is blanked out, and then the content is pasted from the clipboard back into the page. This procedure, where the steps following the drawing of the selection rectangle are recorded in an action, accomplishes black edge removal and centering in a rapid and efficient way. This manual treatment of individual pages is an opportunity for taking note of each image and observing at an early stage whether noise removal was too aggressive and returning back to work on ScanFix and the original images if necessary. This is also an opportunity for restoring photos and other non-textual content that is often damaged by despecking.

Following the manual processing, files in the intermediate directory are deskewed in batch using ScanFix, transferring the deskewed copies to the final directory of the processed images. Automatic deskewing is not performed until the pages are cleaned up in Photoshop, because the cleaner the input, the better is ScanFix able to detect and correct skew. The book is then revised generally. In case of damage to images during the ScanFix batch, files are restored from the intermediate directory. Pages that are still skewed after automatic deskewing are manually corrected, where the SF_SKEWTO string option in ScanFix is used. Deskewing bitonal images is not achievable in Photoshop, and converting to grayscale, editing, and then converting back to bitonal causes alterations in the shape of characters, which has a damaging effect on OCR.

Deskewing in ScanFix alters the dimensions of images slightly, so, Photoshop is used in automation mode to resize the canvas back to the original dimensions. Finally, because Photoshop does not support CCITT G4 compression, images are recompressed in batch using ACDSee. Upon completion, the intermediate directory is discarded.

It is highly desirable to generate skew-free pages in the scanning phase. Scanning a single page typically takes 6 s. It is always very helpful for the scanning operator to spend more time in adjusting the scanned page rather than making this correction in the processing phase.

OCR

The OCR process is language-specific, and, therefore, the OCR phase is split into two branches: a Latin branch and an Arabic branch.

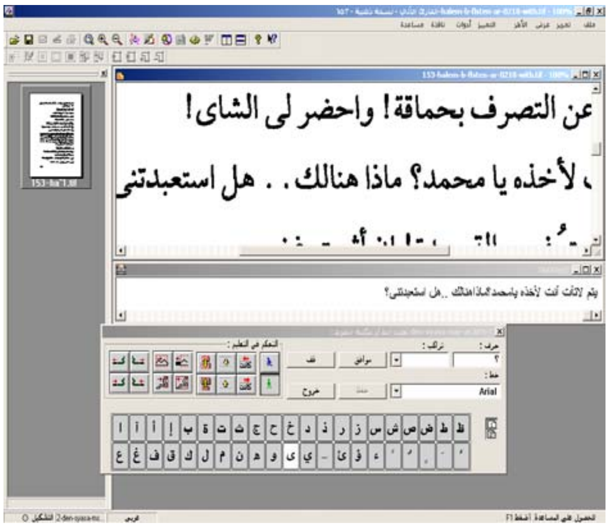
In the Latin branch, ABBYY's FineReader is the system used to produce text and eventually PDF from page images. Other than selecting the appropriate Latin language, the software's settings are kept at their defaults. However, automatic orientation correction is turned off, because it could mistakenly rotate pages that are already oriented correctly. The batch function is used to run the entire PTIFF directory into the OCR. A single "text-under-image" PDF is saved at 150 dpi containing all pages of the book. The outcome of the recognition is kept in FineReader's native format, the FRF files, under the TXT directory, which makes it possible to export the results to any of the supported formats without repeating the OCR process itself later.

Unlike Latin-based scripts, Arabic is written cursively, where characters are joined together. As such, an Arabic character assumes different forms according to its position in the word: initial, medial, final, or isolated. Arabic characters also rely extensively on non-connected components, such as dots and diacritics. In addition, there exist many variants of Arabic fonts. Certain fonts are rather intricate, featuring complex ligatures, where characters overlap and do not share a common baseline. Furthermore, suppliers of Arabic fonts do not all follow a specific standard. This all makes segmenting and recognizing Arabic text a difficult problem for a computer system. The software used in the OCR of Arabic books is Sakhr's Automatic Reader. Additionally, though, a text enhancement process is applied before the actual OCR.

The condition of Arabic characters varies greatly between different kinds of printings, such as old and new, light and heavy, and solid and dot matrix printings. Text enhancement, therefore, has the potential to significantly improve recognition accuracy during the actual OCR. This pre-OCR text enhancement is carried out using ScanFix, where smoothing and completion features are particularly employed. Because of the difficulties associated with this complex script in OCR, enhancement of Arabic text can be a delicate business, where the effects and defects of each action must be weighed out. Although text enhancement is in fact image processing, it is not performed in the actual processing phase and is delayed until the OCR phase, because this irreversible process must be tested through the OCR software, and the specialists working in the OCR phase are likely to have a better sense of which kind of text works better for the OCR engine. At the end of the OCR phase, the final text-enhanced images overwrite the images in the PTIFF directory.

During the actual OCR process, Automatic Reader works on the text-enhanced pages to convert the images into computer text. Here, special techniques are applied in order to maximize the recognition accuracy. In particular, OCR engine settings are adjusted and a "learning" process is employed.

The learning process consists of using two representational pages of the book to determine which patterns the OCR engine has trouble with and build a "font file" that associates such patterns with the right characters (Fig.5). This font file is in fact built on an initial font file selected from a set of pre-built font libraries, which helps recycle work and improves performance. Using a different page, accuracy is manually calculated before and after the learning in order to have a record of the real OCR performance. In the end, all page images are fed into Automatic Reader in a batch


عن التصرف بحماقة! واحضر لى الشاى!

، لأخذه يا محمد؟ ماذا هنالك . . هل استعبدتنى

Fig.5 Font learning

Font	Low bound	High point
AR-H1	97.70%	99.50%
AR-H2	97.60%	99.50%
AR-H3	97.04%	99.10%
AR-H4		Under construction
AR-L1	92.70%	96.70%
DT-M1		Under construction
DT-L2	88.40%	96.80%
TA-H1	97.30%	99.10%
TA-H2	97.60%	99.20%
TA-H3		Under construction
TA-H4	96.50%	97.74%
TA-L1	94.00%	97.70%
TA-L4	94.00%	97.90%
TA-M2	95.80%	98.80%
TA-M4	94.50%	97.50%
X		

Fig.6 OCR font libraries

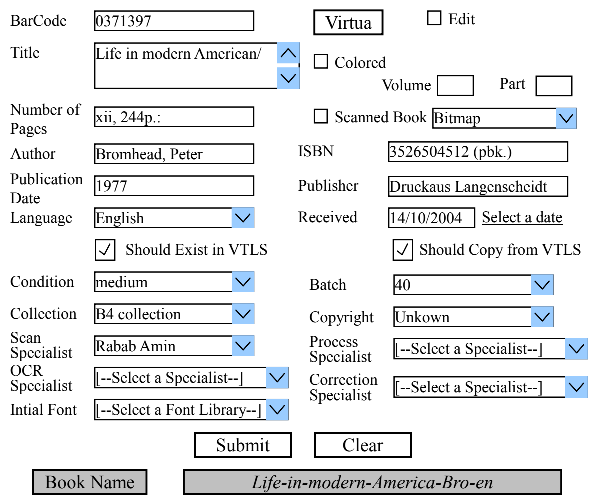
and results written to a merged ART file, the Automatic Reader's native format, which is then converted to XML to be integrated into the process that leads to publishing.

In order to improve OCR accuracy for the Arabic language, a set of custom recognition font libraries have been built. Each of these recognition font libraries is a database of character glyphs that together describe a particular type of script. Using such libraries, the OCR engine is able to better recognize text on images of printings that belong to one of these groupings. These font libraries were built by "training" Automatic Reader's OCR engine on carefully selected and classified sets of scanned pages spanning a wide variety of printings. Classification is based on three criteria: script type, printing quality, and font size. At present, within this scheme of classification, there exists 16 different font libraries, three of which are under construction, and one is a virtual group known as "Group X" used to tag unclassifiable printings and handwritings. Fig.6 lists the 16 font libraries. Based on testing results while compiling these font libraries, each library is assigned a minimum accuracy to be achieved when applying the library to images that belong in this font group. The typesetting fonts Traditional Arabic (TA), Arabic Transparent (AR), and DecoType (DT) have been used as a guide in the naming of the OCR font libraries, while letters have been used to denote high (H), medium (M), and low (L) quality, and numerals from 1 (largest) to 5 (smallest) have been used to indicate the size.

METADATA

Books are selected and brought in batches from BA's Library Service Department. Books are categorized into collections usually denoted by the source of the books. The batch and collection information is maintained per book. For productivity tracking, the condition of the printed book is also recorded in the database. Thus, the time spent on scanning a book is evaluated with respect to the page count and the physical condition of the book, as books in bad physical condition often take longer to scan than books in good condition.

The printed book is checked whether it has been already scanned by entering the value of its BA barcode into the database system if the book exists in BA's book collection. In such case, metadata can be directly retrieved from the Integrated Library System (ILS). For BA, Virtua VTLS is used as ILS. For books that do not belong to BA's collection and do not have a BA barcode, minimal metadata entry is performed, in which a combination of book title, author, and publisher is entered and used to check for book duplications. If the book has not been previously scanned, then the scanning phase can proceed. Fig.7 shows insertion of the metadata information of a printed book into the book database using a Web tool.

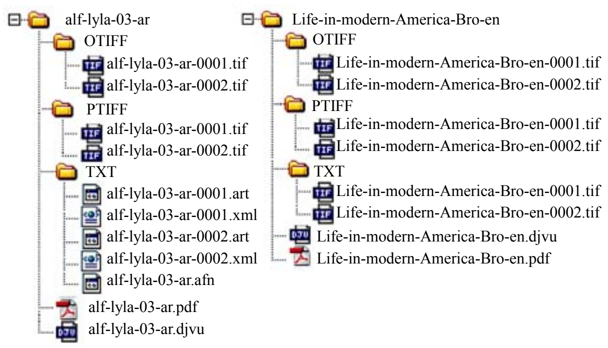


BarCode	0371397	Virtua	☐ Edit
Title	Life in modern American/	☐ Colored	Volume Part
Number of Pages	xii, 244p.:	☐ Scanned Book	Bitmap
Author	Bromhead, Peter	ISBN	3526504512 (pbk.)
Publication Date	1977	Publisher	Druckaus Langenscheidt
Language	English	Received	14/10/2004 Select a date
	✓ Should Exist in VTLS		✓ Should Copy from VTLS
Condition	medium	Batch	40
Collection	B4 collection	Copyright	Unkown
Scan Specialist	Rabab Amin	Process Specialist	[--Select a Specialist--]
OCR Specialist	[--Select a Specialist--]	Correction Specialist	[--Select a Specialist--]
Intial Font	[--Select a Font Library--]		
	Submit	Clear	
Book Name	Life-in-modern-America-Bro-en		

Fig.7 Metadata

DIRECTORY STRUCTURE AND NAMING CONVENTION

As a part of the workflow, it is important to keep a standard directory structure with uniqueness, hence each book has a unique "book directory name" in the ASCII character set. In the case of Latin books (Fig.8), the book directory name is obtained by concatenating the book title with spaces replaced by dashes, author abbreviation, and the two-letter ISO-639 language code. For example, an English book titled "Life in Modern America" whose author is "Peter Bromhead" will have the book directory name of life-in-modern-America-Bro-en. If the book is multi-volume, then the volume number followed by a dash is added just before the two-letter language code. Considering the same example, if the book is the second volume, then its book directory name will be life-in-modern-America-Bro-2-en. The book directory name of Arabic books is



```
alf-lyla-03-ar
    OTIFF
        alf-lyla-03-ar-0001.tif
        alf-lyla-03-ar-0002.tif
    PTIFF
        alf-lyla-03-ar-0001.tif
        alf-lyla-03-ar-0002.tif
    TXT
        alf-lyla-03-ar-0001.art
        alf-lyla-03-ar-0001.xml
        alf-lyla-03-ar-0002.art
        alf-lyla-03-ar-0002.xml
        alf-lyla-03-ar.afn
    alf-lyla-03-ar.pdf
    alf-lyla-03-ar.djvu

Life-in-modern-America-Bro-en
    OTIFF
        Life-in-modern-America-Bro-en-0001.tif
        Life-in-modern-America-Bro-en-0002.tif
    PTIFF
        Life-in-modern-America-Bro-en-0001.tif
        Life-in-modern-America-Bro-en-0002.tif
    TXT
        Life-in-modern-America-Bro-en-0001.tif
        Life-in-modern-America-Bro-en-0002.tif
    Life-in-modern-America-Bro-en.djvu
    Life-in-modern-America-Bro-en.pdf
```

Fig.8 Directory structure of a sample Arabic book (left) and a sample Latin book (right)

obtained in a similar manner (Fig.8). First, however, the book title and author is machine transliterated into English, and then the same procedure is followed.

The book directory name is taken as the name of the directory that contains all files associated with the digital book throughout its passage through the workflow. The book directory contains the following sub-directories: OTIFF: Original scanned TIFF images; PTIFF: Processed TIFF images; TXT: OCR output.

FLOW OF DIGITAL BOOKS

BA has developed a special application, the DL-Client, for managing the flow of digital books across the different queues of the scanning, processing, and OCR phases. Each specialist has a specific DL-Client user which allows the DL-Client to set limits on the number of books in each user's queue. This ensures that digital books will reach their final destinations as quickly as possible and will not be placed in users' queues for a long period.

DL-Client ensures that the digital book exists only in one place during the book's lifecycle. Whenever the digital book completes a phase, its book directory is moved to its appropriate new location in the queues.

After successfully entering the metadata of a certain book, the DL-Client creates the book directory –using the generated book directory name–under the books.SCAN folder on the scanning PC. After the scanning is complete, books are moved from books.SCAN queue to books.SCAN.complete queue on the storage server. After scanning the digital book, the DL-Client records the page count, represented by the number of TIFF files, into the Digital Lab's database.

The processing phase starts by pulling a digital book from books.SCAN.complete on the storage server to the local directory books.PROCESS on the processing PC. Upon processing completion, the digital book is moved from books.PROCESS on the processing PC to books.PROCESS.complete on the storage server.

The same procedure takes place in the OCR phase. A digital book is pulled from books.PROCESS. complete and is placed locally in the directory books. OCR on the OCR PC. Upon OCR completion, the digital book is moved from books.OCR on the OCR

PC to books.OCR.complete on the server.

Following the OCR, books are moved to the archiving queue, where they are written to CDROM and tape. In addition, books are copied to the encoding system's queue, where image-on-text documents are generated to be collected by the publishing system. Fig.9 shows the flow of digital books.

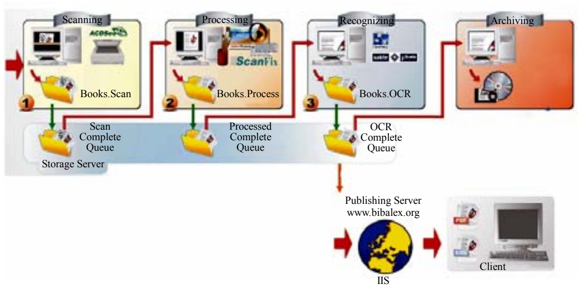


Fig.9 Flow of digital books and digital book queues

ENCODING

Publishing through image-on-text

Unlike content originally composed in digital format, digitized content presents a challenge in publishing. The presentation of digitized books should provide at least the functionality of the original documents, and, where appropriate, provide improvements that digital presentation makes possible. Ideally, users should be able to access, display, search, and navigate digitized documents as effectively as possible using familiar interfaces. Digital content produced from non-digital material contains three levels of information. The first level is the image level, where paper-based material is presented as arrays of pixels. The second level is the text level, where textual information is extracted through analysis of the images. Finally, the third level is the structure level, which represents the logical and the physical layout of the document, where elements of the images are associated with the textual information. The challenge in publishing of digitized material, therefore, arises in presenting this full reconstructed information in the most usable format. In digital publishing of originally non-digital items, the following must be addressed:

(1) Preserving the layout. The layout of the original document should be preserved, including formatting, structure, figures, tables, etc., such that the user is presented with an exact copy of the original.

(2) Possibility to search. It is desirable that the user be able to search in the text of the published document and locate the exact location in the document where the search terms occur.

(3) Efficient image compression. Because improved distributability is an important advantage, efficient image compression is necessary to render feasible the transfer of digitized work across networks. Yet, image quality must not be unacceptably compromised.

(4) Multilingual text support. Since our goal is to digitize whatever is possible of the written works of humanity, supporting all human languages becomes a necessity. The key to supporting multilingual text in digitized documents is in the publishing format to implement a robust international standard character set such as Unicode (Allen and Becker, 2003).

(5) Multipaging. A book consists of many pages that are bound together. To be able to publish digital items that are convenient to distribute and browse, the publishing format has to be designed to contain multiple pages in a single file, and, in turn, the format viewing software has to provide convenient means for browsing through multipage documents.

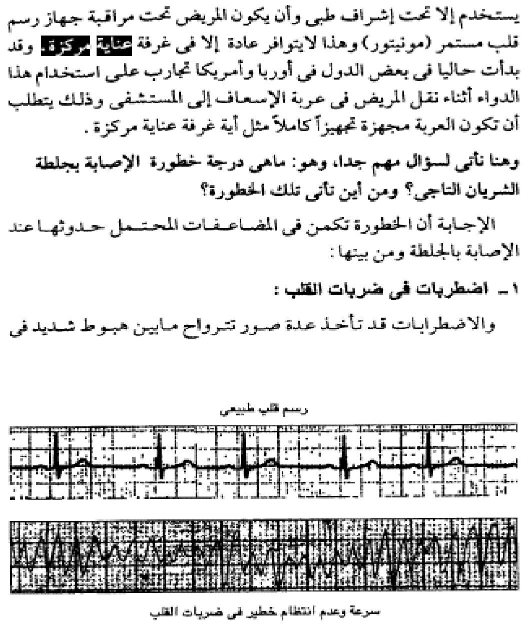
There are four different approaches in publishing digitized documents. The simplest of these approaches is to publish the material as scanned page images, such as the Gallica digital library of the BNF (http://www.gallica.bnf.fr), the Gutenberg's Bible, and the Shakespeare in Quarto of the British Library (http://www.bl.uk/treasures). This approach has the advantage of preserving the exact original look of books, but provides no means for a computer to search, copy, or otherwise process textual information.

To compensate for this shortcoming, the second approach involves publishing the actual text obtained either through manual data entry or OCR techniques. Manual data entry which is an approach adopted by endeavors such as Project Gutenberg (www.gutenberg.org), has the drawback of losing all original layout, although it ensures high text accuracy. However, this manual approach suffers from being slow and expensive. In contrast, numerous projects resort to publishing OCR output for its being a quicker alternative.

Although errors in OCR text may not cause problems in tasks such as text categorization, they may have a negative effect if the text is read by a human being. Therefore, a third approach to publishing digitized documents attempts to bring together the advantages of the first and second approaches by pairing up the image with its textual information. For instance, in (Hong and Srihari, 1997; Lesk, 1996) OCR documents are represented in HTML, where words, which are recognized with high confidence recognition results, are used; otherwise, original word images are substituted. A different method for pairing text and images, which is used in Carnegie Mellon's UDL system (UDL, 2004), is to provide an interface that allows the reader to easily switch between a page's image and its OCR plain text version while reading a digital book. The same method was used in the Making of America digital library (MOA) (Kenney and Rieger, 2000). Although the third approach is more comprehensive than any of the former two approaches considered by itself, it still treats the page image and the text as two separate items while they are merely different representations of one thing.

The fourth approach aims at benefiting from both representations of the information by pairing the image with the associated OCR text in an overlapping manner in a multilayered document. The page image is positioned as the first layer on the *Z*-axis, which is the visible part. The text tokens are positioned according to layout by specifying bounding box information in a hidden layer behind the image. A person viewing such document never sees the actual text but instead sees the original page image independent of how well OCR worked. Yet, retrieval systems can still search, highlight, and copy-and-paste the hidden text to the level of accuracy achieved by the OCR, as shown in Fig.10. Image-on-text documents are also referred to as text-under-image or hidden text documents.

Multilayering is not a new concept. Adopted by Adobe Acrobat in the PDF format (PDF Reference, 2004), image-on-text has proven extremely useful as it preserves exact layout while allowing access to text, and, therefore, has been used by many digital libraries, such as IEEE and ACM but only with Latin languages. OCR engines can produce the necessary components of the layered presentation, as a result of layout analysis and segmentation, and several commercial OCR systems for Latin languages output image-on-


يستخدم إلا تحت إشراف طبي وأن يكون المريض تحت مراقبة جهاز رسم قلب مستمر (مونيتور) وهذا لايتوافر عادة إلا في غرفة عناية مركزة. وقد بدأت حاليا في بعض الدول في أوربا وأمريكا تجارب على استخدام هذا الدواء أثناء نقل المريض في عربة الإسعاف إلى المستشفى وذلك يتطلب أن تكون العربة مجهزة تجهيزاً كاملاً مثل أية غرفة عناية مركزة .

وهنا نأتي لسؤال مهم جدا، وهو: ماهي درجة خطورة الإصابة بجلطة الشريان التاجي؟ ومن أين تأتي تلك الخطورة؟

الإجابة أن الخطورة تكمن في المضاعفات المحتمل حدوثها عند الإصابة بالجلطة ومن بينها :

١ ـ اضطربات في ضربات القلب :

والاضطرابات قد تأخذ عدة صور تتراوح مابين هبوط شديد في

Fig.10 Arabic image-on-text with highlighting

text documents. For non-Latin languages, however, image-on-text technology is often not readily available.

Ding *et al*.(2004) produced multi-level Chinese documents with hidden text under the image layer, through an application embedded in their TH-OCR engine. Documents generated by the TH-OCR engine can be published in PDF, HTML and XML. Modification for the kernel OCR technology was introduced to support Japanese and Korean languages. Although the TH-OCR extended the support for multilayered documents to non-Latin languages, it is still limited to three Asian languages and is not generalized. Further, the publishing framework is embedded in the OCR kernel and, therefore, cannot be reused for other languages, especially that the TH-OCR software is applied only in China and is licensed to limited applications.

General viewers for formats that are capable of image-on-text, such as DjVu and PDF, are available, but image-on-text-aware viewers are needed to manipulate this specific type of documents. The Multivalent project (Phelps and Wilensky, 2001), sponsored by the NSF Digital Libraries Initiative, brings about an alternative approach to browsing of digital documents that works well for image-on-text. Multivalent is based on the concept of documents consisting of layers, which are manipulated by add-in modules known as "behaviors". Switching between the image and OCR layers, seeing through the image layer and examining pieces of the OCR layer using "lenses", and annotating the content are possible in Multivalent. Currently, Multivalent's supported formats include PDF and XDOC, which is a native OCR format from ScanSoft. One disadvantage of using XDOC in publishing, however, is that page images are stored externally and do not make use of modern compression technologies, such as shape clustering and wavelet-based compression. Multivalent presently lacks support for the DjVu format.

A framework

The Universal Digital Book Encoder (UDBE) is a framework for the encoding of image-on-text documents that features a pluggable architecture for OCR engines and format encoders.

The main concept in the design of the UDBE is that it adopts a Common OCR Format (COF) that captures the necessary information for image-on-text documents. OCR Converters convert recognition results of OCR engines into the COF, and Format Handlers, then, encode the COF along with page images into image-on-text documents, as shown in Fig.11.

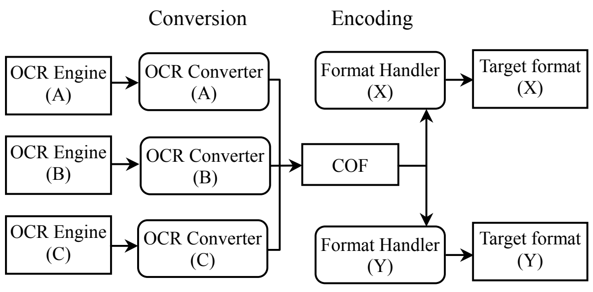


Fig.11 Concept of the UDBE

The UDBE allows for the integration of any OCR engine through OCR Converters, which convert the native OCR format into the COF, rendering the UDBE independent of the native format, which is specific to the engine. Likewise, it allows for the support of any target format through Format Handlers.

The components of the UDBE are illustrated in Fig.12. The two blocks of input are post-processed page images which are enhanced versions of original page images, and OCR text in the native format of the OCR engine that processed the page images. For each book, the UDBE launches an independent process for each of the Format Handlers to process the input blocks and write a file in the target format. Eventually, a digital repository collects these files to publish.

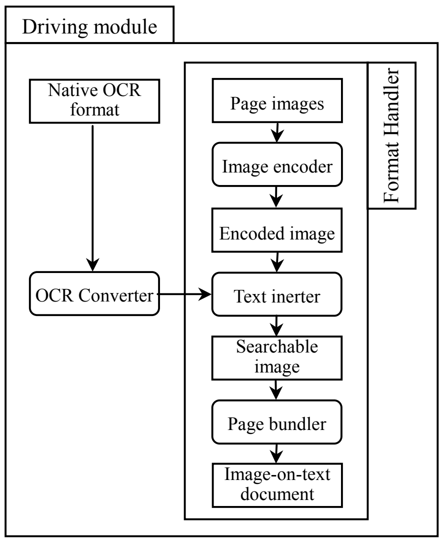


Fig.12 Components of the UDBE

Each format handling process consists of three general steps. Firstly, each page image is encoded according to the compression scheme of the target format. At this point, the encoded images are not associated with text, and, therefore, are unsearchable. Secondly, the COF is traversed and OCR text is inserted in a layer behind each encoded image, reconstructing pages into searchable documents while preserving their original layout. In the last step of format handling, all encoded images are concatenated into one document with multiple pages, which is the light-weight image-on-text bundle that is eventually published. Finally, a driving module binds together all OCR Converters and Format Handlers. This driving module invokes the appropriate OCR Converters for native OCR data and invokes the Format Handlers to produce target formats. The driving module enables the encoding to proceed in automated fashion.

All together, the UDBE is made up of two input blocks, namely, native OCR data for each supported OCR format and page images, a conversion module for each integrated OCR engine, three format handling modules and an output block for each supported target format, and a driving module that manages the overall encoding process.

1. Common OCR Format

To allow Format Handlers to encode image-on-text documents using recognition results of any OCR engine, the UDBE requires the representation of recognition results in a Common OCR Format (COF). Integration of an OCR engine, thus, consists of the implementation of an OCR Converter, which is a module that parses the engine's native OCR format and converts it to the UDBE's COF.

Encoding of image-on-text documents requires two pieces of OCR data: word strings and bounding box coordinates. A word string is a sequence of characters that make up a whole word that the OCR engine recognized while analyzing a page image. Word strings should be represented in Unicode (Allen and Becker, 2003) in order to accommodate multilingual text and must be encoded in Unicode's UTF8 format in order to facilitate portability. Bounding box coordinates are the coordinates in pixels of the enclosing rectangle that describe a word's location and dimensions on a page image.

In coming up with a model for the COF that captures word strings and bounding box coordinates, it would have been desirable that the UDBE adopts a format that is a standard or as close as possible to a standardized representation of OCR data. It was revealed that, although there is currently no standard belonging to a governing body, such as ISO or W3C, for representing OCR data, there are two applicable formats that are commonly used. One is DjVuXML (DjVuXML manual page, http://djvulibre.djvuzone.org/doc/man/djvuxml.html), which is an XML-based format modelled after HTML that provides a simple scheme for describing hidden text layers as well as other information in DjVu documents (DjVu Technology Primer, 2004). The other is the Document Attribute Format Specification (DAFS) (DAFS, 1994), which is a binary-coded format that provides a specification for document decomposition and is used in applications such as document layout analysis, OCR, and logical analysis. Although DjVuXML accommodates OCR text in a simple structure, adopting a format designed for a specific purpose, namely, describing DjVu documents, could be inadvisable in a universal document encoding application, because it could prove limited in accommodating desired features. On the other hand, although DAFS is a general purpose format for document decomposition, adopting it could be inadvisable because of its complexity and being binary-coded. Therefore, due to the lack of an enforced standard and the concerns mentioned regarding both DjVuXML and DAFS, a COF was designed for the UDBE to be specifically customized for the representation of image-on-text documents.

The UDBE's COF is an XML (Yergeau *et al.*, 2004) format inspired by both DjVuXML and DAFS. Unlike DAFS, the COF is not binary-coded but is based on XML, a widely used standard that represents information in plain text in a self-explanatory fashion, making it less complex to parse. The structure of the COF is illustrated in Fig.13.

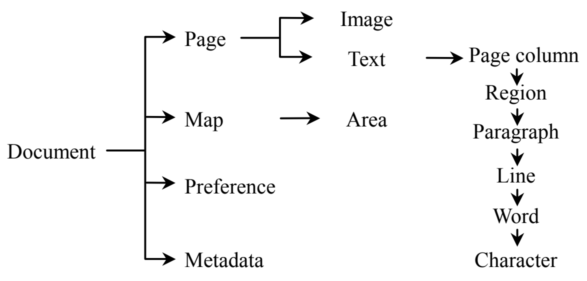


Fig.13 Structure of the COF

An image-on-text document in the COF consists of pages, maps, a preference block, and metadata. Page elements containing the two actual components of the content, namely, the image and the text. The text element is a hierarchical structure that consists of the following levels: page column, region, paragraph, line, word, and char (character). Each of these levels has a coords attribute that defines its bounding box and a CSS (Lie and Bos, 1999) style attribute that describes its visual properties. Within a document, map elements define HTML-like image maps, preference elements set viewer preferences, and metadata elements embed bibliographic metadata using the Dublin Core Metadata Element Set (DCMI, 2004) in RDF (Resource Description Framework) form.

It should be noted that it is always possible to build an OCR Converter for an OCR engine if the native output is accessible. The native output, which contains characters and bounding box data, is a by-product of layout analysis, segmentation, and recognition, which are functions intrinsic in the OCR process. Thus, in a current OCR suite, although the final output is often just the symbolic content, the detailed information on layout analysis, segmentation, and recognition is always stored as an intermediate step, and this information includes bounding box coordinates of text blocks, lines, words, and characters.

2. Format handling

Format Handlers are the part of the UDBE that processes page images and OCR text in the COF to produce files in target formats for publishing. Format handling is broken down into three general functions: Image Encoding, Text Insertion, and Page Bundling. For each target format, a separate module handles each of these functions.

Image Encoding operates on page images. It does not operate on the COF. The Image Encoding module accepts each individual page image and encodes it into an individual file in the target format. Because page images constitute the greater chunk of data relative to OCR text, and because page images are the viewable piece in image-on-text documents, the primary concern of image encoding is reducing file size and preserving quality.

Based on an image's properties, the Image Encoding module decides on the compression scheme to apply according to the target format's specification. Document format specifications often support a number of different image compression methods, each designed for a specific type of image data. For instance, JPEG compression performs well on grayscale and RGB data but is inadequate for bilevel data.

Certain compression algorithms cause a level of loss of detail in order to achieve larger compression ratios. In developing an image encoding module, therefore, it is important to judge to what degree a lossy encoding method is acceptable. In addition, image encoding could perform resolution down-sampling in order to further cut down the file size at the expense of quality by representing the image in fewer pixels. The effects of resolution downsampling are less obvious in RGB images containing simple textures than in bilevel images containing text.

Text Insertion operates on output from Image Encoding and OCR Conversion. In Text Insertion, encoded page images are associated with OCR text to produce searchable images. The Text Insertion module processes image-only documents in the target format and text in the COF then encodes image-on-text documents in the target format.

A valid target format specification could explicitly support hidden text. Alternatively, the format could support the necessary features that allow for the implementation of hidden text functionality. In essence, an image-on-text format is capable of containing images and text, such that images are displayed and text is not but is highlightable based on settable parameters. For instance, a format that is capable of displaying images, positioning text using pixel coordinates, setting text width and height independently, and transparency is a valid image-on-text format. The Text Insertion module of such format would place each word from the COF at its pixel location in the target format, set the text object's width and height to match the word's bounding box, and apply transparent rendering to the text.

It may be necessary during Text Insertion to perform coordinate system conversions in order to adapt bounding box coordinates from the COF's pixel-based top-left-origined space to the target format's. In converting from units of pixels to units of non-pixels, such as points, millimeters or inches, the page image's resolution must be taken into account. In addition, it may be necessary to perform character set conversions on the UTF8 word strings if the target format requires a different encoding.

Page Bundling is the final step in format handling. It operates on output from the Text Insertion. The purpose of Page Bundling is to group individual searchable page images produced during Text Insertion into one bundle containing all pages, making it convenient to browse. The functionality of a Page Bundling module is simple. It consists of constructing a blank document in the target format and inserting each individual searchable page image into it in the correct page sequence.

Implementation and performance

The current implementation of the UDBE supports Arabic, Persian, and Latin languages through the integration of a multilingual OCR engine, and encodes into two target formats, namely, DjVu and PDF.

An OCR Converter was implemented for the native format of Automatic Reader, which is an OCR product that features engines for the recognition of Arabic, Persian, and 18 Latin-based languages, such as English, French, and Spanish. Automatic Reader also features a learning system, which allows for the definition of custom recognition fonts to expand the types of prints the engine handles and to improve recognition accuracy. This OCR Converter, thus, enables the UDBE to handle digitized books in a wide range of languages. Automatic Reader makes its native format available through an SDK.

DjVu and PDF are two formats that were found suitable for use in publishing of digitized books according to the requirements outlined earlier. Since they are the only known formats to support image-on-text in light-weight documents suitable for Web publishing, support for these two formats was integrated into the UDBE.

Developed during the last decade of the twentieth century at AT&T Labs, DjVu is an image compression technique and a file format specifically designed for building high-visual-quality digital libraries. The compression technique uses a mixed raster content (MRC) imaging model, where advanced image analysis is used to segment the image into layers and compress each layer separately using the algorithm that best suits its content (DjVu Technology Primer, 2004). Traditional image compression models are either designed to compress natural images with few sharp edges or images containing text and mostly consisting of sharp edges. DjVu works by combining these two approaches on document images through segmentation, which involves the separation of text from background and pictures (Haffner *et al.*, 1999). In a DjVu document, text layers are typically stored at 300 dpi and compressed using the shape clustering JB2 algorithm, which takes advantage of similarities between characters, while pictures are typically stored at 100 dpi and compressed using the wavelet-based IW44 algorithm.

LizardTech acquired the DjVu technology from AT&T Labs and commercialized it in the company's Document Express product but also made a free implementation of the technology available through the DjVu Libre open source project. While bilevel encoding in DjVu Libre is competitive with LizardTech's, color image encoding remains superior in the commercial product due to its ability to segment color images into layers. The UDBE's implementation of the DjVu Format Handler is built around DjVu Libre in order to provide a purely free solution. Alternatively, however, a solution built around LizardTech's Document Express is also implemented. These implementations are wrappers around DjVu Libre and Document Express, respectively.

Each of DjVu Libre and Document Express includes a DjVuXML (DjVuXML manual page, http://djvulibre.djvuzone.org/doc/man/djvuxml.html) processor that updates DjVu documents according to the information in the DjVuXML file. Text Insertion in the DjVu format, therefore, involves the transformation of the COF into DjVuXML and applying the DjVuXML file to the encoded DjVu images. Page Bundling of DjVu documents is a straightforward task, because both DjVu Libre and Document Express include a utility to merge DjVu documents.

The Portable Document Format (PDF) (Phelps and Wilensky, 2001) is a document format from Adobe that has earned extremely wide popularity. Having many aspects in common with the PostScript printing language, the PDF format is robust enough to represent a wide variety of document types including image-on-text documents. For each input page image, the Image Encoding module for the PDF format constructs a PDF page equal in width and height to the dimensions of that image, encodes the image data using the compression method supported by the PDF specification (PDF Reference, 2004) that yields the smallest file size, then inserts the encoded data into the document such that it covers the whole page. Page images that feed into the encoding system are either bilevel or RGB images. CCITT G4 compression is designed specifically for bilevel image data and will always yield better compression ratios for such data than either JPEG or deflate. For RGB data, JPEG compression is adequate for continuous tone images, such as photographs and colorful drawings with smooth edges, while deflate compression is adequate for images consisting of large blocks of solid colors. Commonly, however, color book pages are of the former type and compress best with JPEG.

The Text Insertion module reads the OCR text data in the COF, which is the unified OCR text representation that the encoding system has adopted. Unlike the DjVu format, the PDF format does not per se specify image-on-text functionality. However, it is possible to achieve image-on-text functionality in PDF documents by utilizing the PDF text and font operators to emulate bounding box behavior.

In the UDBE, where supporting Arabic text in the target publishing formats was a primary objective, it was necessary to “ligaturize” Arabic text strings in order to display correctly in the search side pane in Adobe Reader. In other words, the PDF Text Inserter must transform each character string in its input into its appropriate shape according to its position in a word. In doing so, the Text Inserter makes use of a module to perform Arabic character shaping. An open source Arabic character shaping module was used in this implementation (iText: A free Java PDF library, http://www.lowagie.com/iText). Latin text does not require ligation, because Latin characters always assume one shape regardless of their position in a word.

The PDF Format Handler’s Page Bundling module concatenates the image-on-text PDF pages coming from the Text Insertion into a single PDF document. The module does this bundling by reading each individual page into a PDF XObject—which is an object that acts as a template, encapsulating other objects, such as text and graphics, within it—then inserting the template into the appropriate page position in the multipage PDF document.

Support for the PDF target format was written in the Java programming language based on the open source iText API (iText: A free Java PDF library, http://www.lowagie.com/iText). The PDF Format Handler’s image access and conversion functionalities in the Image Encoding module were based on the Java Advanced Imaging (JAI) API (JAIAPI, 2004).

Evaluation of the performance of this implementation of the UDBE framework showed that for either DjVu or PDF, the increase in file size after adding hidden text layers to image-only documents remains within reasonable bounds, justifying the decision to publish scanned books in image-on-text format in order to achieve the strongly desired searchability and ability to otherwise process text in applications such as machine translation. Fig.14a compares average page file sizes of image-only and image-on-text output for a set of bilevel Arabic books, and Fig.14b does a similar comparison for Latin. Performance has also been compared to alternative systems capable of producing image-on-text for Latin languages, namely, Acrobat, FineReader, and LizardTech’s Document Express, which possesses built-in OCR functionality, and performance has been found to be comparable. The UDBE has been used in the encoding of nearly 7000 books of BA’s digitized work, and the system will continue to be used on other batches of books as their OCR becomes available.

DAR

The Digital Assets Repository (DAR) (Saleh *et al.*, 2005) is a system developed at the BA to create and maintain the digital library collections. The system introduces a data model capable of associating the metadata of different types of resources with the content, such that searching and retrieval can be done efficiently. The system automates the digitization process of library collections as well as the preservation and archiving of the digitized output and provides public access to the collection through browsing and

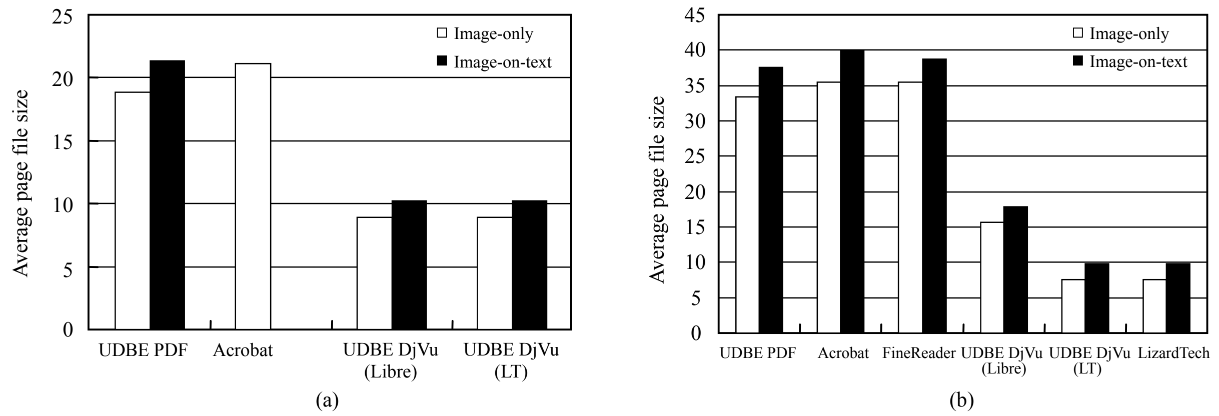


Fig.14 Bilevel Arabic (a) and Latin (b) image-only and image-on-text

searching capabilities. The goal of this project is building a digital resources repository by supporting the creation, use, and preservation of varieties of digital resources as well as the development of management tools. These tools help the library to preserve, manage, and share digital assets. The system is based on evolving standards for easy integration with Web-based interoperable digital libraries.

The system core consists of two fundamental modules: the Digital Assets Factory (DAF), which is responsible for the automation of the digitization workflow; and the Digital Assets Keeper (DAK), which acts as a repository for digital assets either produced by the DAF or directly introduced into the repository.

FUTURE WORK

Continuing to develop better scanning, image processing, and OCR techniques is critical to the production of good quality digital books. This also requires continuing to watch for emerging technology in scanning hardware and image processing and OCR software. Better Arabic OCR software is particularly desirable. In preparing books for publishing, further work is necessary in order to integrate more OCR engines into the UDBE and to research alternatives to DjVu and PDF. In addition, it is important to continue to investigate published standards that could be beneficial for formulation of the COF. The UDBE's current implementation of the PDF Image Encoder also requires further work in order to achieve better compression through JBIG2 and JPEG2000 encoding, and possibly through image segmentation and MRC. In addition, extensions to existing viewers could be desirable in order to add features specific to image-on-text, such as the ability to display the hidden text layer.

References

Allen, J., Becker, J., 2003. The Unicode Standard, Version 4.0. Addison-Wesley, Reading, MA.

DAFS (Document Attribute Format Specification), 1994. RAF Technology, Inc., Redmond, Washington.

DCMI (Dublin Core Metadata Element Set), 2004. The Dublin Core Metadata Initiative.

Ding, X., Wen, D., Peng, L., Liu, C., 2004. Document Digitization Technology and its Application for Digital Library in China. Proceedings of the First International Conference on Document Image Analysis for Libraries, p.46-53.

DjVu Technology Primer, 2004. LizardTech, Inc., Seattle, WA.

Haffner, P., Bottou, L., Howard, P., Le Cun, Y., 1999. DjVu: Analyzing and Compressing Scanned Documents for Internet Distribution. Proceedings of International Conference on Document Analysis and Recognition (ICDAR'99), p.625-628.

Hong, T., Srihari, S., 1997. Representing OCRed Documents in HTML. Proceedings of the Fourth International Conference on Document Analysis and Recognition, p.831-834.

JAIAPI (Java Advanced Imaging API), 2004. Sun Microsystems, Santa Clara, CA.

Kenney, A., Rieger, O., 2000. Moving Theory into Practice: Digital Imaging for Libraries and Archives. Research Libraries Group.

Lesk, M., 1996. Substituting Images for Books: The Economics for Libraries. Proceedings of Symposium on Document Analysis and Information Retrieval, p.1-16.

Lie, H., Bos, B., 1999. Cascading Style Sheets, Level 1. The World Wide Web Consortium.

PDF Reference, 2004. Fourth Edition. Adobe Systems, Inc., San Jose, CA.

Phelps, T., Wilensky, R., 2001. The Multivalent Browser: A Platform for New Ideas. Proceedings of the ACM Symposium on Document Engineering, Atlanta, US.

Saleh, I., Adly, N., Nagi, M., 2005. DAR: A Digital Assets Repository for Library Collections. Proceedings of ECDL'05, Vienna, Austria.

UDL (The Universal Digital Library), 2004. Carnegie Mellon University, Pittsburg, PA.

Yergeau, F., Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., 2004. Extensible Markup Language (XML), 1.0, Third Edition. The World Wide Web Consortium.