A Framework for the Encoding of Multilayered Documents

Youssef Eldakar †
† Bibliotheca Alexandrina
El Shatby 21526
Alexandria, Egypt

Noha Adly †*

Magdy Nagi †*
** Computer and Systems*
Engineering Department
Alexandria University
Alexandria, Egypt

{youssef.eldakar,noha.adly,magdy.nagi}@bibalex.org

Abstract

Electronic publishing of material digitized using imaging and OCR calls for a special delivery format capable of reconstructing original documents in a well-usable electronic form. We present a framework for the universal encoding of multilingual image-on-text documents, enabling retrieval systems to text-search and highlight hits on original page images. A generalized format for representation of image-on-text allows for integration of different OCR engines and target format encoders. This framework's current implementation encodes multilingual content into DjVu and PDF. Performance has been evaluated with focus on file size and shown that overhead of adding text layers is small compared to advantages and that output is comparable to other systems.

1. Introduction

Recent years have been witnessing a growth in the digital publishing of documents. Although currently most documents are born digital, it is necessary to account for converting the massive heritage of paper-based documents into digital versions. Today's imaging and OCR technology is making it possible to digitize material at a fast pace. The presentation of such digitized material should provide at least the functionality of originals, and, where appropriate, provide improvements. Ideally, users should be able to access, display, search, and navigate through digitized documents as effectively as possible using familiar interfaces. Digitized content contains three levels of information: image, text, and structure. The challenge in publishing of digitized material, therefore, becomes in presenting this full reconstructed information in the most usable format, and, therefore, the following must be addressed:

- *Preserving the layout*: the layout of the original document must be preserved, including formatting, structure, figures, and tables, such that an exact copy of the original is presented.
- *Possibility to search*: searching in the text of the published document and locating the exact place where the search terms occur must be possible.
- *Efficient image compression*: because improved distributability is an important advantage, efficient image compression is necessary to make feasible the transfer of digitized work across networks. Yet, image quality must not be unacceptably compromised.
- *Multilingual text support*: since our goal is to digitize whatever is possible of the written works of humanity, supporting all human languages is a necessity. The key to this is to implement a robust international standard character set such as Unicode [1].
- *Multipaging*: since a book consists of many pages that are bound together, to be able to publish digital books that are convenient to distribute and browse, the publishing format has to accommodate multiple pages in a single file, and the viewing software has to provide convenient means for browsing multipage documents.

2. Related work

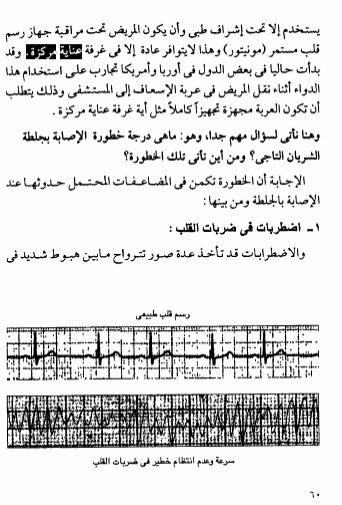
In publishing digitized documents, there exists four different approaches. The simplest of these approaches is to publish the material as scanned page images as in the Gallica digital library of the BNF (gallica.bnf.fr), and the Gutenberg Bible and the Shakespeare in Quarto of the British Library (prodigi.bl.uk). This approach

has the advantage of preserving the exact original look of digitized items but provides no means for a computer to search, copy, or otherwise process textual information. Although several research efforts were done for Document Image Retrieval in the fields of IR and CBIR [17], the retrieval is limited to document image features only, such as layout similarity or word matching [14]. Other efforts focused on the recognition and extraction of metadata, such as the title, author and citations, especially for scientific publications such as [7] [11].

To compensate for this deficiency, 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 (Gutenberg.org) – bears the drawback of losing all original layout and being slow and expensive, although it ensures high text accuracy. In contrast, numerous projects resort to publishing OCR output for being a quicker alternative. Although OCR suites extract the layout, they do not support a standardized output format that different viewers can understand. In an attempt to preserve the layout in a standardized format, Ishitani [10] proposed a method for transforming OCR output to an XML document while reconstructing the layout according to specified DTDs. Another proposal [19] converted OCR output to an SGML/XML representation, namely, Structured Document Format (SDF) based on the hierarchical document format of DAFS [5]. Similar efforts were done using SGML [6] [12] [18]. It should be noted, though, that the variety of document types that could be processed through these methods are limited to the ones with specified DTDs. Further, the quality of the published documents is also dependent on the level of accuracy achieved by the OCR as the user is exposed to reading the text with the errors produced by the OCR.

Although OCR errors might not cause problems in tasks such as text categorization, they will have a negative effect if the text is read by humans. Therefore, a third approach attempts to bring together the advantages of the first and second approaches by pairing up the image with the text. For instance, in [9] [13], OCR output is represented in HTML, where recognition results are used for words recognized with high confidence; otherwise, original word images are substituted. A different method for pairing text and images, which is used in UDL (ulib.org) and the Making of America digital library (MOA) (cdl.library.cornell.edu/moa), is to provide an interface that allows the reader to easily switch between a page's image and its OCR plain text. Although the third approach is more comprehensive than any of the former two 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 laying the image over its associated OCR text 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 Figure 1.


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

وهنا نأتى لسؤال مهم جدا، وهو: ماهى درجة خطورة الإصابة بجلطة الشريان التاجى؟ ومن أين تأتى تلك الخطورة؟

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

١ - اضطربات فى ضربات القلب :

والاضطرابات قد تأخذ عدة صور تتروح مابين هبوط شديد فى

رسم قلب طبيعى

سرعة وعدم انتظام خطير فى ضربات القلب

٦٠

Figure 1. Arabic image-on-text with highlighting

This multilayered format is not a new concept. Adopted by Adobe Acrobat in the PDF format [15], image-on-text has proven extremely useful, as it preserves exact layout while allowing access to text. Image-on-text has been used by different digital libraries, such as those of IEEE and ACM, but only with Latin languages. Recent large-scale digitization projects – such as Google Print (books.google.com) and the Open Content Alliance (openlibrary.org) have also used image-on-text with Latin languages. For non-Latin languages, image-on-text technology is often not readily available. Ding et al. in [3] produced Chinese image-on-text documents through an application embedded in TH-OCR, extended the availability of image-on-text to three Asian languages. Yet, this solution is limited within TH-OCR and cannot be extended and reused. Image-on-text-aware viewers are needed to manipulate this specific type of documents.

The Multivalent project [16], sponsored by the NSF Digital Libraries Initiative, offers a browser that brings about an alternative approach to browsing of digital documents that works well for image-on-text.

We present in this paper a framework for encoding multilingual digital books into image-on-text. We present a system supporting Arabic and Latin and applicable to any OCR engine and thus extensible to any language. The system currently supports two publishing formats, namely, DjVu and PDF, preserving the layout, and multipaging. The size of the published documents is comparable to the ones generated through other systems, and the overhead incurred by the hidden text layer is reasonable.

The paper is organized as follows. Section 3 introduces the Universal Digital Book Encoder and describes its main components. Section 4 discusses the implementation of the UDBE. This implemented system has been evaluated against existing systems supporting image-on-text in Latin and its performance evaluation is presented in Section 5 along with its performance on Arabic data sets.

3. The Universal Digital Book Encoder

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.

3.1. Overview

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* encode the COF along with page images into image-on-text documents, as shown in Figure 2.

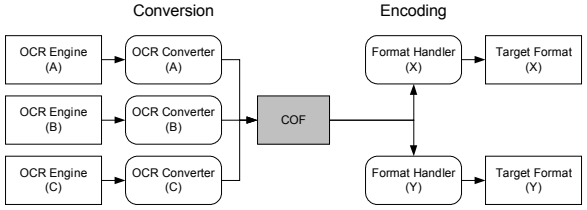


Figure 2. Concept of the UDBE

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 Figure 3. The two blocks of input are processed page images – which are enhanced versions of original page images – and OCR text in the native format of the OCR engine used. 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.

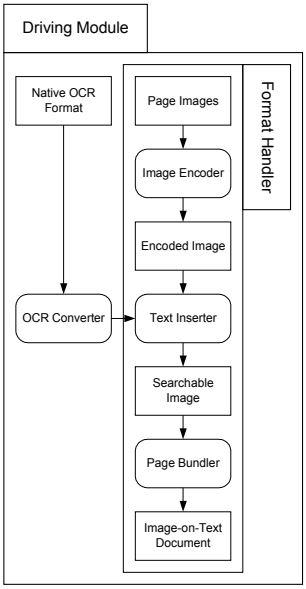


Figure 3. Components of the UDBE

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 an automated fashion.

3.2. Common OCR Format

Integration of an OCR engine consists of the implementation of an *OCR Converter* that converts the engine's native format into the Common OCR Format

(COF). In coming up with a model that captures the two necessary pieces of OCR data – word strings and bounding box coordinates – it would have been desirable to adopt a format that is a standardized representation of OCR data. It was revealed that, although there is currently no standard belonging to a governing body for representing OCR data, there are three applicable formats that are commonly used. One is DjVuXML [4], an XML-based format modeled after HTML that provides a scheme for describing DjVu documents [8]. Another is the Document Attribute Format Specification (DAFS) [5], a binary-coded format that provides a specification for document decomposition in applications such as document layout analysis, OCR, and logical analysis. The third is Analyzed Layout and Text Object (ALTO) [2], used by CCS in the docWORKS software. 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 for its complexity and being binary-coded. And while ALTO contains many of the features desired in a COF and has been heavily used by the Library of Congress and others, it lacks certain desirable capabilities, such as image maps and specialized viewer preferences. Therefore, due to the lack of an enforced standard and the concerns mentioned regarding DjVuXML, DAFS, and ALTO, a COF was designed to specifically accommodate the representation of image-on-text documents.

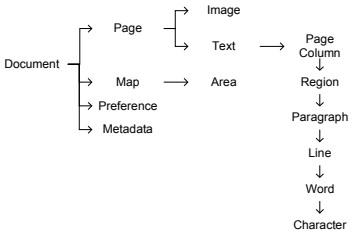


Figure 4. Structure of the COF

The UDBE's COF is an XML [20] format inspired by both DjVuXML and DAFS. Unlike DAFS, the COF is not binary-coded but based on XML since it is a widely used standard that represents information in a clear and self-explanatory fashion, making it less complex to parse than binary data. The structure of the COF is illustrated in Figure 4. An image-on-text document in the COF consists of pages, maps, a preference block, and metadata. Text elements contain multilingual strings with bounding box coordinates. The COF embeds CSS formatting, HTML-like image maps, image-on-text-specific viewer preferences, and Dublin Core bibliographic metadata. It is always possible to write an *OCR Converter* for an OCR engine as bounding box information is always available as a product of layout analysis, segmentation, and recognition.

3.3. 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.

3.3.1. Image Encoding. The *Image Encoder* receives 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 Encoder* decides on the compression scheme to apply according to the target format's specification. Certain compression algorithms cause a level of loss of detail in order to achieve larger compression ratios. In developing an *Image Encoder*, therefore, it is important to judge to what degree a lossy encoding method is acceptable. In addition, *Image Encoding* could perform resolution downsampling 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.

3.3.2. Text Insertion. In *Text Insertion*, encoded page images are associated with OCR text to produce searchable images. The *Text Inserter* processes image-only documents in the target format and text in the COF then encodes image-on-text documents in the target format. 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 Inserter* 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.

3.3.3. Page Bundling. *Page Bundling* groups individual searchable page images produced during *Text Insertion* into one bundle containing all pages. The functionality of a *Page Bundler* consists of constructing a blank document in the target format and inserting each individual searchable page image into it in the correct page sequence.

4. Implementation

The current implementation of the UDBE supports Arabic, Persian, and 18 Latin languages through the integration of a multilingual OCR engine, and encodes into two target formats, namely, DjVu and PDF. This implementation is continuing to be used at BA and has thus far produced more than 23,000 books. The scanned images are manually processed for text enhancement to improve the recognition accuracy of the OCR output. It is expected that future OCR suites supporting Arabic will be able to automate this process.

4.1. OCR Converter for Automatic Reader

An *OCR Converter* was implemented for the native format of Automatic Reader, an OCR product that features engines for the recognition of Arabic, Persian, and 18 Latin 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. Because this current implementation does not recognize text blocks and paragraphs, elements in the COF higher than the line element are unused. In the future, bounding box coordinates could be analyzed to provide a more detailed description of the text.

4.2. Format Handlers for DjVu and PDF

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 at AT&T Labs, DjVu [8] 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. 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.

The UDBE's implementation of the DjVu *Format Handler* was built around DjVu Libre in order to provide a purely free solution. Alternatively, however, a solution built around LizardTech's Document Express was also implemented. While bilevel encoding in DjVu Libre is competitive with LizardTech's, color *Image Encoding* remains superior in the commercial product. Both DjVu Libre and LizardTech's Document Express use the JB2 shape clustering compression scheme with bilevel content and the IW44 wavelet-based compression scheme with RGB content. However, LizardTech's Document Express features an image segmenter to separate text, illustrations, and background, compressing each separately into an MRC document [8].

The Portable Document Format (PDF) [15] from Adobe has earned an extremely wide popularity. The PDF format is robust enough to represent a wide variety of document types, including image-on-text documents. The three functionalities of *Image Encoding*, *Text Insertion*, and *Page Bundling* have been implemented in a PDF *Format Handler*.

The PDF *Image Encoder* encodes each page image into a PDF page with equivalent dimensions to the input using one of the compression methods supported by the PDF specification [15] that best suits the type of the image and yields the smallest file size. Specifically, CCITT G4 is used with bilevel images, and JPEG, which is based on the Discrete Cosine Transform (DCT), is used with RGB images. The image resolution is also downsampled to 150 dpi in order to further reduce the file size. As the PDF format does not provide for image-on-text functionality as an explicit specification, the PDF *Text Inserter* makes use of the PDF text and font operators to emulate bounding box behavior. In the UDBE, where supporting Arabic was a primary objective, it was necessary to ligaturize Arabic characters in order for matched text to display correctly

in the search side pane in Adobe Reader. This ligaturization consists of shaping each character into its appropriate presentation form according to its position in the word. Ligaturization is required for cursive scripts, and, therefore, is not applicable to Latin languages. Finally, the PDF *Page Bundler* joins together individual image-on-text PDF pages into a single multipage PDF file, which is linearized in order to optimize it for "fast Web viewing." Linearization allows pages to be downloaded in the background as the document is being viewed.

Support for the PDF output target was written in the Java programming language based on the iText API (www.lowagie.com/iText), an open source API for manipulating documents in the PDF format.

5. Performance evaluation

To evaluate the performance of the UDBE in producing image-on-text output in terms of file size, we compare image-only and image-on-text documents produced by different systems across eight different data sets of digitized books. The eight data sets are made up of four data sets of Arabic books and four data sets of Latin books. In turn, each of these four data sets is made up of two bilevel and two RGB data sets. Finally, each of these two data sets consists of a data set of small books and a data set of large books, selected according to page dimensions.

For each book in each data set, we produce an image-only and an image-on-text DjVu using the UDBE's implementation based once on DjVu Libre and once on LizardTech's Document Express. In addition, we also produce an image-on-text DjVu using Document Express's built-in Expervision OCR engine, which applies only to the four Latin data sets. Along with DjVu, we produce image-only and image-on-text PDF documents using the UDBE's implementation. Further, we produce image-only PDF documents for all sets in addition to image-on-text documents for Latin sets using Acrobat. Finally, we produce image-on-text documents for Latin sets using the FineReader OCR software. Image-only and image-on-text output is compared in terms of file size. Output from different systems is also compared. Average page file sizes of image-only and image-on-text output of the bilevel-small sets, the bilevel-large sets, and the RGB-large sets are plotted in Figure 5 through Figure 10. Plots of the two RGB-small sets were omitted due to space constraints. Note that FineReader's image-on-text output is plotted along with a plot of Acrobat's image-only output for comparison purposes.

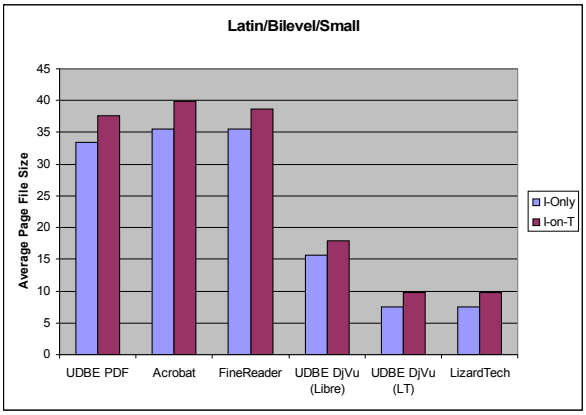


Figure 5. Latin/bilevel/small

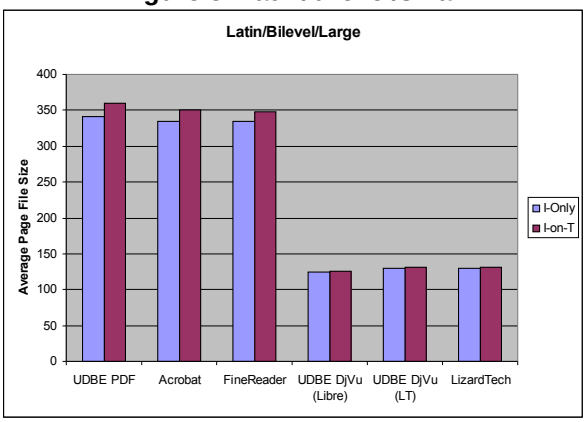


Figure 6. Latin/bilevel/large

Examining the file size increase associated with the UDBE's image-on-text compared to image-only, the increase in image-on-text PDF averages 5 kB in the Latin-bilevel-small set, as in Figure 5, and 19 kB in the large set, while the increase in image-on-text DjVu averages 2 kB in both bilevel sets. In Arabic-bilevel, these increases average 2 and 5 kB for PDF in the small and large sets, as shown in Figure 7 and Figure 8, respectively, and 1 and 3 kB for DjVu, respectively. The increase shows to be quite small and this extra overhead is justified by the added value and functionality associated with image-on-text formats. It is noticed also that for Latin sets the PDF average increases incurred with the UDBE are almost identical to those incurred with Acrobat and FineReader, and, similarly, DjVu average increases incurred with the UDBE are almost identical to those incurred by LizardTech.

Across Latin plots, it is apparent that the UDBE's image-on-text output is equivalent in size to image-on-text produced from other systems, namely, Acrobat and FineReader for PDF, and LizardTech for DjVu. In the case of Arabic, where an alternative to the UDBE for image-on-text is not available, it is observed that the UDBE's image-only PDF output is almost the same size

as Acrobat's, and the UDBE's image-only DjVu output is equivalent to what DjVu Libre and LizardTech would have independently produced. Since the text overhead incurred was found to be small and image-only file sizes comparable to other systems, this indicates that the UDBE performs well for producing Arabic image-on-text documents.

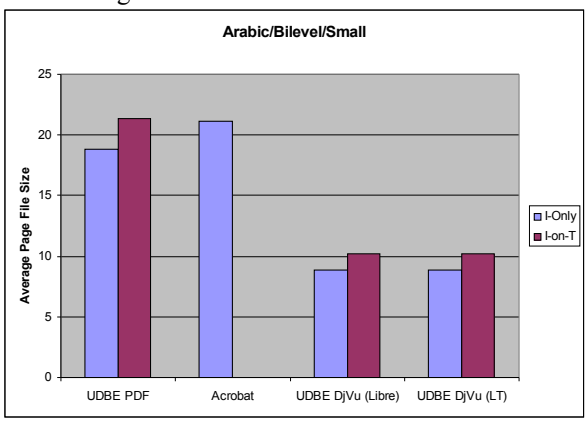


Figure 7. Arabic/bilevel/small

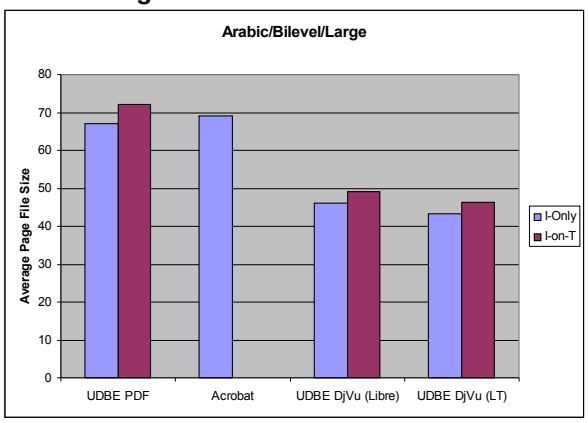


Figure 8. Arabic/bilevel/large

For RGB-large data sets shown in Figure 9 and Figure 10, it is similarly observed that the text overhead imposed by image-on-text is small and the relative increase is even smaller than in bilevel sets due to the nature of large color images, which occupy large memory space. From Figure 9, it is noted that Acrobat's image-on-text output presents an unusually large overhead, which is attributed to Acrobat's conversion of JPEG images to ZIP compression during OCR, which is less efficient for continuous tone images.

In bilevel as well as RGB plots, it is evident that DjVu consistently yields higher compression ratios than CCITT G4 and JPEG compression in PDF with the exception of DjVu Libre's color compression. While DjVu Libre's bilevel compression generally proves competitive with LizardTech, its color compression lags behind. In fact, LizardTech's performance on RGB sets consistently outruns PDF in general and DjVu Libre in particular, which is attributed to LizardTech's image segmentation functionality that enables the encoding of MRC images [8].

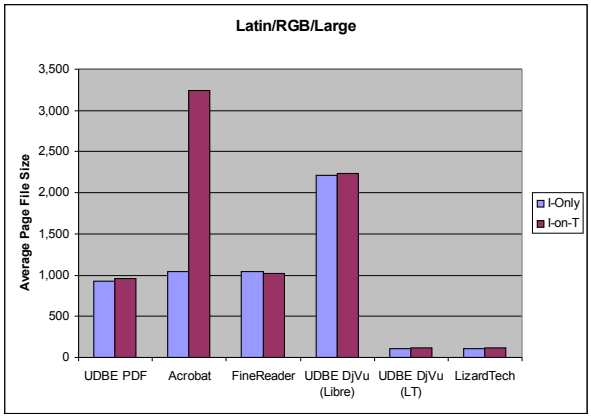


Figure 9. Latin/RGB/large

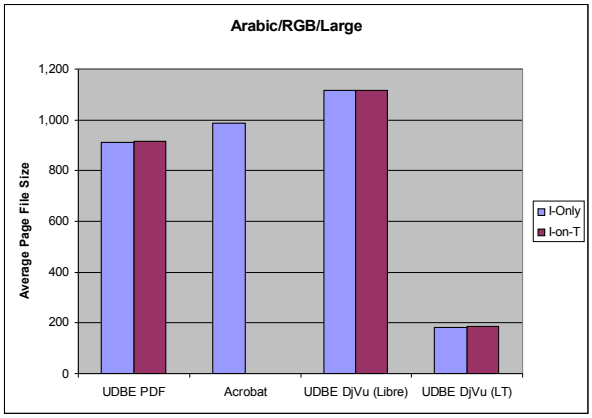


Figure 10. Arabic/RGB/large

6. Conclusion

The image-on-text approach offers a robust solution to digital publishing in imaging-and-OCR-based digital libraries, as it seamlessly combines the best in the previous image-only and text-only publishing approaches. Adequate publishing of image-on-text must provide efficient image compression, multilingual text support, and multipaging. The presented UDBE framework renders it possible to utilize OCR results of any engine to compile image-on-text documents in any valid target format by adopting a Common OCR Format (COF). The current implementation of the UDBE showcases the concept with an *OCR Converter* for Automatic Reader and *Format Handlers* for DjVu and PDF, making it possible to produce multilingual – namely, Latin, Arabic, and Persian – image-on-text documents in an automated fashion. It has been shown 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 documents in image-on-text to achieve the strongly desired searchability and ability to otherwise process text in applications such as machine translation. It has also been shown that the performance of the UDBE is comparable to other systems capable of producing Latin image-on-text, namely, Acrobat, FineReader, and Document Express.

Further work is necessary to integrate more OCR engines and research alternatives to DjVu and PDF. The current PDF *Image Encoder* also requires further work to achieve better compression through JBIG2 and JPEG2000 encoding and possibly through image segmentation and MRC. In addition, extensions to existing viewers is desirable to add features specific to image-on-text.

7. Acknowledgments

We thank Mohamed Ramadan for contributing in writing the *OCR Converter* for Automatic Reader and *Text Inserter* for PDF.

8. References

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

[2] ALTO: Analyzed Layout and Text Object. http://www.ccs-gmbh.com/alto/.

[3] Ding, X., Wen, D., Peng, L.; and Liu, C. Document digitization technology and its application for digital Library in China. In Proceedings of the first international conference on Document image analysis for libraries (2004). 46-53.

[4] DjVuXML manual page. http://djvulibre.djvuzone.org/doc/man/djvuxml.html.

[5] Document Attribute Format Specification. RAF Technology, Inc., Redmond, Washington, 1994.

[6] Entlich, R., Olsen, J., Garson, L., Lesk, M., Normore, L., and Weibel, S. Making a digital library: the contents of the CORE project. ACM Trans. IS, 15, 2 (Apr 1997), 103-123.

[7] Giuffrida, G., Shek, E., Yang, J. Knowledge-based metadata extraction from PostScript files. In Proceedings of the fifth ACM conference on Digital libraries (San Antonio, US, Jun 2000). 77-84.

[8] Haffner, P., Bottou, L., Howard, P., Le Cun, Y. DjVu: analyzing and compressing scanned documents for Internet distribution. In Proceedings of international conference on Document analysis and recognition (ICDAR '99) (1999). 625-628.

[9] Hong, T., Srihari, S. Representing OCRed documents in HTML. In Proceedings of the fourth international conference on Document analysis and recognition (Aug 1997). 831-834.

[10] Ishitani, Y. Document transformation system from papers to XML data based on pivot XML document method. In Proceedings of the seventh international conference on Document analysis and recognition (Aug 3-6, 2003). 250-255.

[11] Lawrence, S., Bollacker, K., Giles, C. Indexing and retrieval of scientific literature. In Proceedings of the eighth international conference on Information and knowledge management (Kansas City, US, Nov 1999). 139-146.

[12] Lefevre, P., and Reynaud, F. ODIL: an SGML description language of the layout structure of documents. In Proceedings of the third international conference on Document analysis and recognition (Aug 14-16, 1995). 480-488.

[13] Lesk, M. Substiuting images for books: the economics for libraries. In Proceedings of symposium on Document analysis and information retrieval (Apr 1996). 1-16.

[14] Marinai, S., Marino, E., Cesarini, F., and Soda, G. A general system for the retrieval of document images from digital libraries. In Proceedings of the first international workshop on Document image analysis for libraries (DIAL) (2004). 150-173.

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

[16] Phelps, T., Wilensky, R. The Multivalent browser: a platform for new ideas. In Proceedings of the ACM symposium on Document engineering (Atlanta, US, Nov 2001).

[17] Smeulders, A., Worring, M., Santini, S., Gupta, A., and Jain, R. Content-based image retrieval at the end of the early years. IEEE Trans. PAMI, 22, 12 (Dec 2000), 1349–1380.

[18] Taghva, K., Nartker, T., Borsack, J. Information access in the presence of OCR errors. In Proceedings of the first ACM workshop on Hardcopy document processing (Nov 2004).

[19] Wang, Y., Phillips, I., and Haralick, R. “From image to SGML/XML representation: one method. In Proceedings of Document layout interpretation and its application (DLIA) (1999).

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