Digitization of Text documents Using PDF/A

Yan Han
and Xueheng Wan

ABSTRACT

The purpose of this article is to demonstrate a practical use case of PDF/A for digitization of text documents following FADGI's recommendation of using PDF/A as a preferred digitization file format. The authors demonstrate how to convert and combine TIFFs with associated metadata into a single PDF/A-2b file for a document. Using real-life examples and open source software, the authors show readers how to convert TIFF images, extract associated metadata and International Color Consortium (ICC) profiles, and validate against the newly released PDF/A validator. The generated PDF/A file is a self-contained and self-described container that accommodates all the data from digitization of textual materials, including page-level metadata and ICC profiles. Providing theoretical analysis and empirical examples, the authors show that PDF/A has many advantages over the traditionally preferred file format, TIFF/JPEG2000, for digitization of text documents.

BACKGROUND

PDF has been primarily used as a file delivery format across many platforms in almost every device since its initial release in 1993. PDF/A was designed to address concerns about long-term preservation of PDF files, but there has been little research and few implementations of this file format. Since the first standard (ISO 19005 PDF/A-1), published in 2005, some articles discuss the PDF/A family of standards, relevant information, and how to implement PDF/A for born-digital documents.[1]

There is growing interest in the PDF and PDF/A standards after both the US Library of Congress and the National Archives and Records Administration (NARA) joined the PDF Association in 2017. NARA joined the PDF Association because PDF files are used as electronic documents in every government and business agency. As explained in a blog post, the Library of Congress joined the PDF Association because of the benefits to libraries, including participating in developing PDF standards, promoting best-practice use of PDF, and access to the global expertise in PDF technology.[2]

Few articles, if any, have been published about using this file format for preservation of digitized content. Yan Han published a related article in 2015 about theoretical research on using PDF/A for text documents.[3] In this article, Han discussed the shortcomings of the widely used TIFF and JPEG2000 as master preservation file formats and proposed using the then-emerging PDF/A as the preferred file format for digitization of text documents. Han further analyzed the requirements

Yan Han (yhan@email.arizona.edu) is Full Librarian, the University of Arizona Libraries, and **Xueheng Wan** (wanxueheng@email.arizona.edu) is a student, Department of Computer Science, University of Arizona.

of digitization of text documents and discussed the advantages of PDF/A over TIFF and JPEG2000. These benefits include platform independence, smaller file size, better compression algorithms, and metadata encoding. In addition, the file format reduces workload and simplifies post-digitization processing such as quality control, adding and updating missing pages, and creating new metadata and OCR data for discovery and digital preservation. As a result, PDF/A can be used in every phase of a digital object in an Open Archival Information System (OAIS)—for example, a Submission Information Package (SIP), Archive Information Package (AIP), and Dissemination Information Package (DIP). In summary, a PDF/A file can be a structured, self-contained, and self-described container allowing a simpler one-to-one relationship between an original physical document and its digital surrogate.

In September 2016, the Federal Agencies Digital Guidelines Initiative (FADGI) released its latest guidelines for digitization related to raster images: *Technical Guidelines for Digitizing Heritage Materials*.[4] The de-facto best practices for digitization, these guidelines provide federal agencies guidance and have been used in many cultural heritage institutions. Both the PDF Association and the authors welcomed the recognition of PDF/A as the preferred master file format for digitization of text documents such as unbound documents, bound volumes, and newspapers.[5]

GOALS AND TASKS

Since Han has previously provided theoretical methods of coding raster images, metadata, and related information in PDF/A, the goals of this article are threefold:

1. present real-life experience of converting TIFFs/JPEG2000s to PDF/A and back, along with image metadata
2. test open source libraries to create and manipulate images, image metadata, and PDF/A
3. validate generated PDF/As with the first legitimate validator for PDF/A validation

The tasks included the following:

- Convert all the master files in TIFFs/JPEG2000 from digitization of text documents into single PDF/A files losslessly. One document, one PDF/A file.
- Evaluate and extract metadata from each TIFF/JPEG2000 image and encode it along with its image when creating the corresponding PDF/A file.
- Demonstrate the runtimes of the above tasks for feasibility evaluation.
- Validate the PDF/A files against the newly released open source PDF/A validator veraPDF.
- Extract each digital image from the PDF/A file back to its original master image files along with associated metadata.
- Verify the extracted image files in the back-and-forth conversion process against the original master image files

Choices of PDF/A Standards and Conformance Level

This article demonstrates using PDF/A-2b as a self-contained self-describing file format. Currently, there are three related PDF/A standards (PDF/A-1, PDF/A-2, and PDF/A-3), each with

HTTPS://DOI.ORG/10.6017/ITAL.V37I1.9878

three conformance levels (a, b, and u). The reasons for choosing PDF/A-2 (instead of PDF/A-1 or PDF/A-3) are the following:

- PDF/A-1 is based on PDF 1.4. In this standard, images coded in PDF/A-1 cannot use JPEG2000 compression (named in PDF/A as JPXDecode). One can still convert TIFFs to PDF/A-1 using other lossless compression methods such as LZW. However, the space-saving benefits of JPEG2000 compression over other methods would not be utilized.
- PDF/A-2 and PDF/A-3 are based on PDF 1.7. One significant feature of PDF 1.7 is that it supports JPEG2000 compression, which saves 40–60 percent of space for raster images compared to uncompressed TIFFs.
- PDF/A-3 has one major feature that PDF/A-2 does not have, which is to allow arbitrary files to be embedded within the PDF file. In this case, there is no file to be embedded.

The authors chose conformance level *b* for simplicity.

- *b* is basic conformance, which requires only necessary components (e.g., all fonts embedded in the PDF) for reproduction of a document's visual appearance.
- *a* is accessible conformance, which means *b* conformance level plus additional accessibility (structural and semantic features such as document structure). One can add tags to convert PDF/2b to PDF/2a.
- *u* represents *a* conformance level with the additional requirement that all text in the document have Unicode equivalents.

This article does not cover any post-processing of additional manual or computational features such as adding OCR text to the generated PDF/A files. These features do *not* help faithfully capture the look and feel of original pages in digitization, and they can be added or updated later without any loss of information. In addition, OCR results rely on the availability of OCR engines for the document's language, and results can vary between different OCR engines over time. OCR technology is getting better and will produce better results in the future. For example, current OCR technology for English gives very reliable (more than 90 percent) accuracy. In comparison, traditional Chinese manuscripts and Pashto/Persian give unacceptably low accuracy (less than 60 percent). The cutting edge on OCR engines has started to utilize artificial intelligence networks, and the authors believe that a breakthrough will happen soon.

Data Source

The University of Arizona Libraries (UAL) and Afghanistan Center at Kabul University (ACKU) have been partnering to digitize and preserve ACKU's permanent collection held in Kabul. This collaborative project created the largest Afghan digital repository in the world. Currently the Afghan digital repository (http://www.afghandata.org) contains more than fifteen thousand titles and 1.6 million pages of documents. Digitization of these text documents follows the previous version of the FADGI guideline, which recommended scanning each page of a text document into a separate TIFF file as the master file. These TIFFs were organized by directories in a file system, where each directory represents a corresponding document containing all the scanned pages of this title. An example of the directory structure can be found in Han's article.

PDF/A and Image Manipulation Tools

There are a few open source and proprietary PDF software development kits (SDK). Adobe PDF Library and Foxit SDK are the most well-known commercial tools to manipulate PDFs. To show readers that they can manipulate and generate PDF/A documents themselves, open source software, rather than commercial tools, was used. Currently, only a very limited number of open source PDF SDKs are available, including iText and PDFBox. iText was chosen because it has good documentation and provides a well-built set of APIs to support almost all the PDF and PDF/A features. Initially written by Bruno Lowagie (who was in the ISO PDF standard working group) in 1998 as an in-house project, Lowagie later started up his own company, iText, and published *iText in Action* with many code examples.[6] Moreover, iText has Java and C# coding options with good code documentation. It is worth mentioning that iText has different versions. The author used iText 5.5.10 and 5.4.4. Using an older version in our implementation generated a non-compatible PDF/A file because the it was not aligned with the PDF/A standard.[7]

For image processing, there were a few popular open source options, including ImageMagick and GIMP. ImageMagick was chosen because of its popularity, stability, and cross-platform implementation. Our implementation identified one issue with ImageMagick: the current version (7.0.4) could not retrieve all the metadata from TIFF files as it did not extract certain information such as the Image File Directory and color profile. These metadata are critical because they are part of the original data from digitization. Unfortunately, the author observed that some image editors were unable to preserve all the metadata from the image files during the conversion process. Hart and De Varies used case studies to show the vulnerability of metadata, demonstrating metadata elements in a digital object can be lost and corrupted by use or conversion of a file to another format. They suggested that action is needed to ensure proper metadata creation and preservation so that all types of metadata must be captured and preserved to achieve the most authentic, consistent, and complete digital preservation for future use.[8]

Metadata Extraction Tools and Color Profiles

As we digitize physical documents and manipulate images, color management is important. The goal of color management is to obtain a controlled conversion between the color representations of various devices such as image scanners, digital cameras, and monitors. A color profile is a set of data that control input and output of a color space. The International Color Consortium (ICC) standards and profiles were created to bring various manufacturers together because embedding color profiles into images is one of the most important color management solutions. Image formats such as TIFF and JPEG2000 and document formats such as PDF may contain embedded color profiles. The authors identified a few open source tools to extract TIFF metadata, including ExifTool, Exiv2, and tiffInfo. ExifTool is an open source tool for reading, writing, and manipulating metadata of media files. Exiv2 is another free metadata tool supporting different image formats. The tiffInfo program is widely used in the Linux platform, but it has not been updated for at least ten years. Our implementations showed that ExifTool was the one that most easily extracted the full ICC profiles and other metadata from TIFF and JPEG2000 files. ImageMagick and other image processing software were examined in Van der Knijff's article discussing JPEG2000 for long-term preservation.[9] He found that ICC profiles were lost in ImageMagick. Our implementation has

HTTPS://DOI.ORG/10.6017/ITAL.V37I1.9878

showed the current version of ImageMagick has fixed this issue. A metadata sample can be found in appendix A.

IMPLEMENTATION

Converting and Ordering TIFFs into a Single PDF/A-2 File

When ordering and combining all individual TIFFs of a document into a single PDF/A-2b file, the authors intended to preserve all information from the TIFFs, including raster image data streams and metadata stored in each TIFF's header. The raster image data streams are the main images reflecting the original look and feel of these pages, while the metadata (including technical and administrative metadata such as BitsPerSample, DateTime, and Make/Model/Software) tells us important digitization and provenance information. Both are critical for delivery and digital preservation.

The TIFF images were first converted to JPEG2000 with lossless compression using the open source ImageMagick software. Our tests of ImageMagick demonstrated that it can handle different color profiles and will convert images correctly if the original TIFF comes with a color profile. This gave us confidence that past concerns about JPEG2000 and ImageMagick had been resolved. These images were then properly sorted into their original order and combined into a single PDF/A-2 file. An alternative is to directly code TIFF's image data stream into a PDF/A file, but this approach would miss one benefit of PDF/A-2: tremendous file size reduction with JPEG2000. The following is the pseudocode of ordering and combining all the TIFFs in a text document into a single PDF/A-2 file.

```
CreatePDFA2(queue TiffList) {
      Create an empty queue XMLQ;
      Create an empty queue JP2Q;

      /* TiffFileList is pre-sorted queue based on the original order */

      /* Convert each TIFF to JPEG2000 losslessly, then add each JPEG2000 and its
            metadata into a queue */
      while (TiffList is NOT empty) {
            String TiffFilePath = TiffList.dequeue();
            string xmlFilePath = Tiff metadata extracted using exiftool;
            XMLQ.enqueue(xmlFilePath);
            String jp2FilePath = JPEG2000 file location from Tiff converted by
                  ImageMagick;
            JP2Q.enqueue(jp2FilePath);
      }

      /* Convert each image's metadata to XMP, add each JPEG2000 and its metadata
            into the PDF/A-2 file based on its original order */
      Document pdf2b = new Document();

      /* create PDF/A-2b conformance level */
      PdfAWriter writer = PdfAWriter.getInstance(doc, new
            FileOutputStream(PdfAFilePath),PdfAConformaceLevel.PDF_A_2B);

      writer.createXmpMetadata(); //Create Root XMP
```

```
      pdf2b.open();
      while(JP2Q is NOT empty){
            Image jp2 = Image.getInstance(JP2Q.dequeue());
            Rectangle size = new Rectangle(jp2.getWidth(), jp2.getHeight()); //PDF
                  page size setting
            pdf2b.setPageSize(size);
            pdf2b.newPage(); // create a new page for a new image
            byte[] bytearr = XmpManipulation(XMLQ.dequeue()); // convert original
                  metadata based on the XMP standard
            writer .setPageXmpMetadata(bytearr);
            pdf2b.add(jp2);
      }
      pdf2b.close();
}
```

Converting PDF/A-2 Files back to TIFFs and JPEG2000s

To ensure that we can extract raster images from the newly created PDF/A-2 file, the authors also wrote code to convert a PDF/A-2 file back to the original TIFF or JPEG2000 format. This implementation was a reverse process of the above operation. Once the reverse conversion process was completed, the authors verified that the image files created from the PDF/A-2 file were the same as before the conversion to PDF/A-2. Note that we generated MD5 checksums to verify image data streams. Images data streams are the same, but metadata location can be varied because of inconsistent TIFF tags used over the years. When converting one TIFF to another TIFF, ImageMagick has its implementation of metadata tags. The code can be found in appendix B.

PDF/A Validation

PDF/A is one of the most recognized digital preservation formats, specially designed for long-term preservation and access. However, no commonly accepted PDF/A validator was available in the past, although several commercial and open source PDF preflight and validation engines (e.g., Acrobat) were available. Validating a PDF/A against the PDF/A standards is a challenging task for a few reasons, including the complexity of the PDF and PDF/A formats. The PDF Association and the Open Preservation Foundation recognized the need and started a project to develop an open source PDF/A validator and build a maintenance community. Their result, VeraPDF, is an open source validator designed for all PDF/A parts and conformance levels. Released in January 2017, the goal of veraPDF is to become the commonly accepted PDF/A validator.[10] Our generated PDF/As have been validated with veraPDF 1.4 and Adobe Acrobat Pro DC Preflight. Both products validated the PDF/A-2b files as fully compatible. Our implementations showed that veraPDF 1.4 verified more cases than Acrobat DC Preflight. Figure 1 shows a PDF file structure and its metadata.

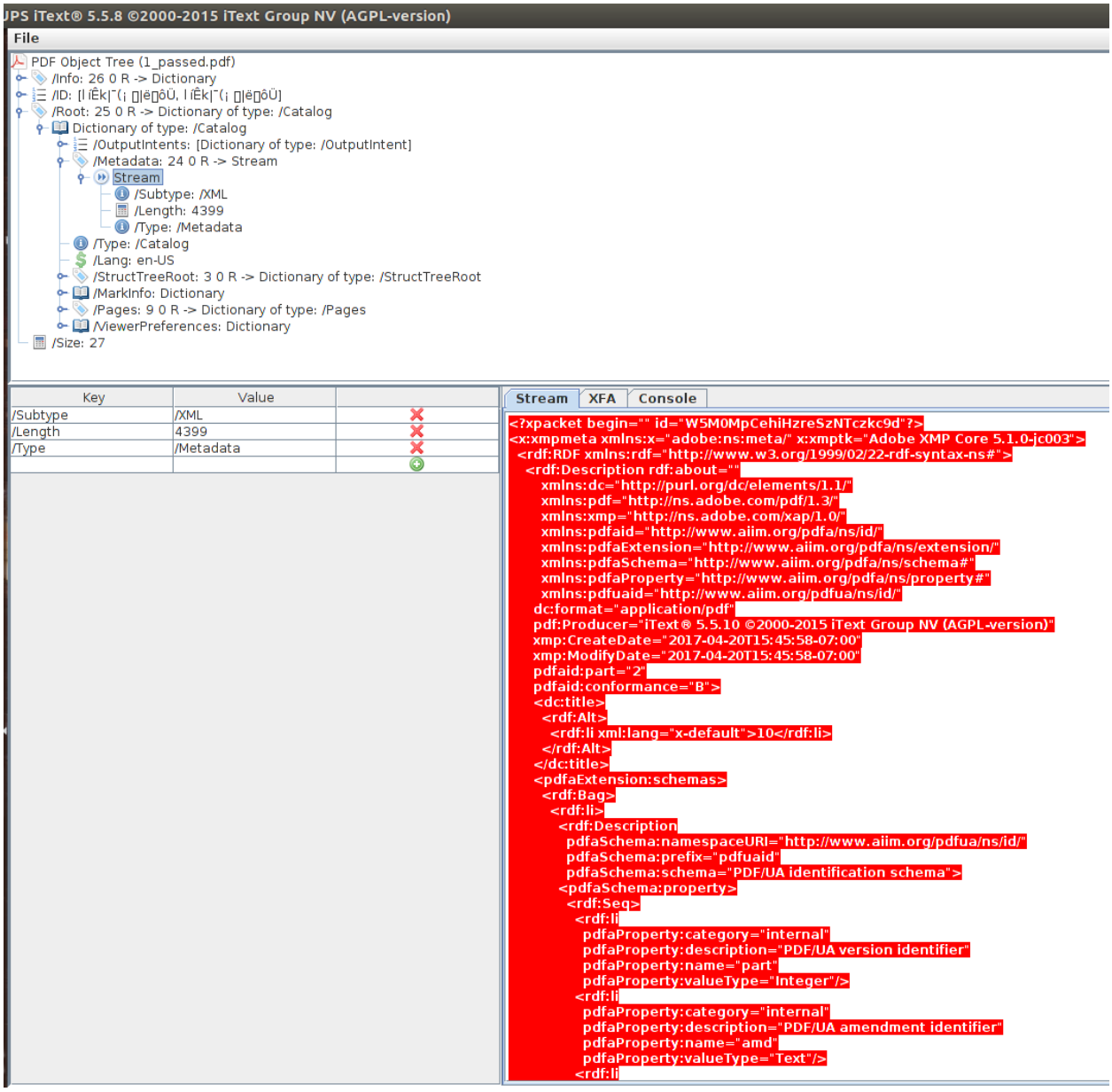

Figure 1. A PDF object tree with root-level metadata.

RUNTIME AND CONCLUSION

The time complexity of our code is $O(\log n)$ because of the sorting algorithms used. TIFFs were first converted to JPEG2000. When JPEG2000 images are added to a PDF/A-2 file, no further image manipulation is required because the generated PDF/A-2 uses JPEG2000 directly (in other words, it uses the JPXDecode filter). Tables 1 and 2 show the performance comparison running in our computer hardware and software environment (Intel Core i7-2600 CPU@3.4GHz, 8GB DDR3 RAM, 3TB 7200-RPM 64MB-cache hard disk running Ubuntu 16.10).

Table 1. Runtimes of converting grayscale TIFFs to JPEG2000s and to PDF/A-2b

No. of Files	Total File Size (MB)	Image Conversion Runtime (TIFFs to JP2s in seconds)	Total Runtime (TIFFs to JP2s to a single PDF/A-2b in seconds)
1	9.1	3.61	3.98
10	91.1	35.63	36.71
20	182.2	71.83	73.98
50	455.5	179.06	184.63
100	910.9	358.3	370.91

Table 2. Runtimes of converting color TIFFs to JPEG2000s and to PDF/A-2b

No. of Files	Total File Size (MB)	Image Conversion Runtime (TIFFs to JP2s in seconds)	Total Runtime (TIFFs to JP2s to a single PDF/A-2b in seconds)
1	27.3	14.80	14.94
10	273	150.51	151.55
20	546	289.95	293.21
50	1,415	741.89	749.75
100	2,730	1490.49	1509.23

The results show that (a) the majority of the runtime (more than 95 percent) is spent in converting a TIFF to a JPEG2000 using ImageMagick (see figure 2); (b) the average runtime of converting a TIFF has a constant positive relationship with the file's size (see figure 2); (c) in

HTTPS://DOI.ORG/10.6017/ITAL.V37I1.9878

comparison, the runtime of converting a color TIFF is significantly higher than that of converting a greyscale TIFF (see figure 2); and (d) it is feasible in terms of time and resources to convert existing master images of digital document collections to PDF/A-2b. For example, the runtime of 1 TB of conversion of color TIFFs will be 552,831 seconds (153.5 hours; 6.398 days) using the above hardware. The authors have already processed more than 600,000 TIFFs using this method.

The authors conclude that using PDF/A gives institutions advantages of the newly preferred master file format for digitization of text documents over TIFF/JPEG2000. The above implementation demonstrates the ease, the reasonable runtime, and the availability of open source software to perform such conversions. From both the theoretical analysis and empirical evidences, the authors show that PDF/A has advantages over the traditional preferred file format TIFF for digitization of text documents. Following best practice, a PDF/A file can be a self-contained and self-described container that accommodates all the data from digitization of textual materials, including page-level metadata and ICC profiles.

SUMMARY

The goal of this article is to demonstrate empirical evidences of using PDF/A for digitization of text document. The authors evaluated and used multiple open source software programs for processing raster images, extracting image metadata, and generating PDF/A files. These PDF/A files were validated using the up-to-date PDF/A validators veraPDF and Acrobat Preflight.

The authors also calculated the time complexity of the program and measured the total runtime in multiple testing cases. Most of the runtime was spent on image conversions from TIFF to JPEG2000. The creation of the PDF/A-2b file with associated page-level metadata accounted for less than 5 percent of the total runtime. Runtime of conversion of a color TIFF was much higher than that of a greyscale one. Our theoretical analysis and empirical examples show that using PDF/A-2 presents many advantages over the traditional preferred file format (TIFF/JPEG2000) for digitization of text documents.

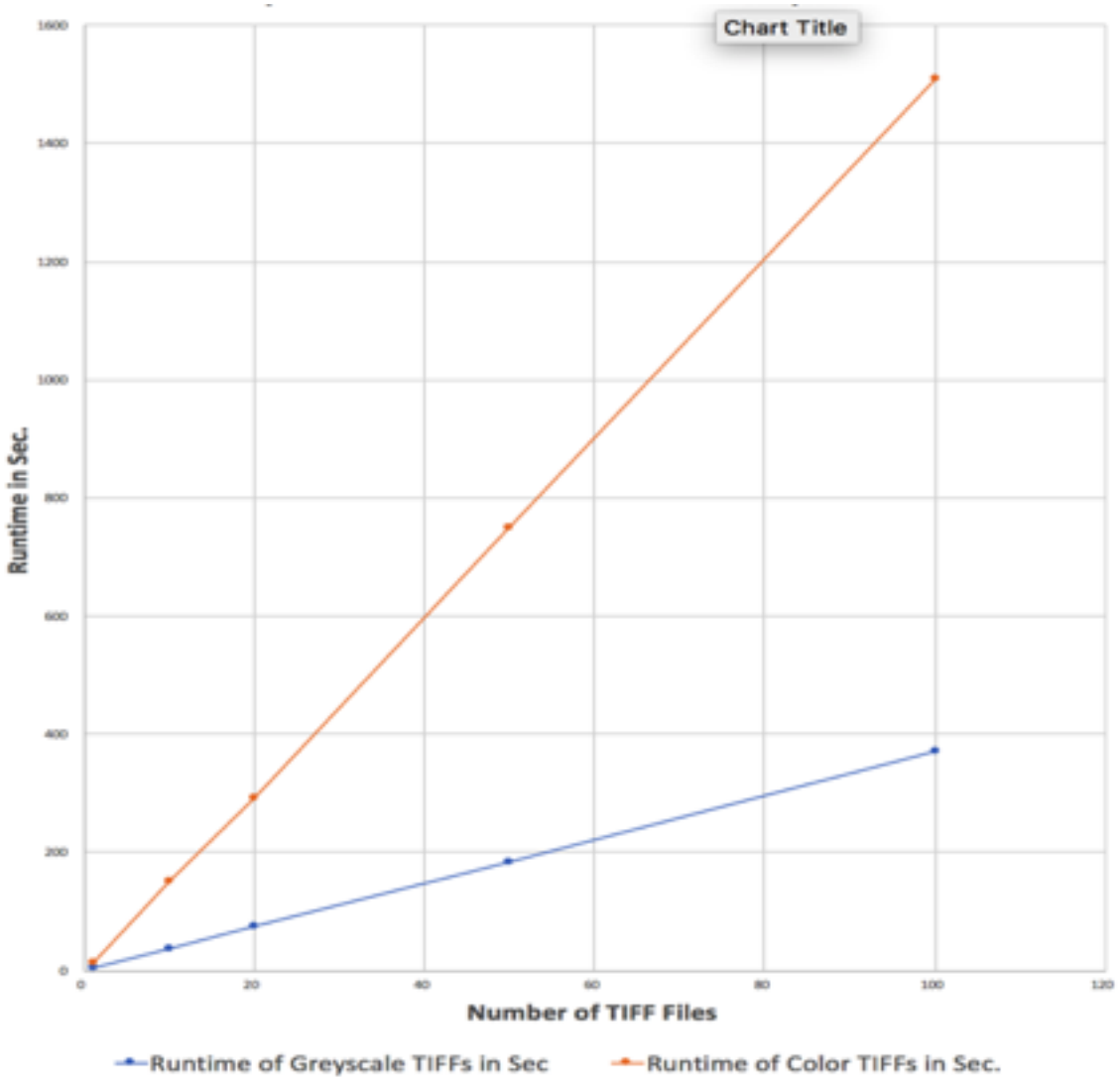

Figure 2. File size, greyscale and color TIFFs and runtime ratio.

HTTPS://DOI.ORG/10.6017/ITAL.V37I1.9878

APPENDIX A: SAMPLE TIFF METADATA WITH ICC HEADER

```
<tiff:BitsPerSample>8</tiff:BitsPerSample>
<IFD0:ImageWidth>3400</IFD0:ImageWidth>
<IFD0:ImageHeight>4680</IFD0:ImageHeight>
<IFD0:BitsPerSample>8 8 8</IFD0:BitsPerSample>
<IFD0:Compression>Uncompressed</IFD0:Compression>
<IFD0:PhotometricInterpretation>RGB</IFD0:PhotometricInterpretation>
<IFD0:StripOffsets>(Binary data 41025 bytes, use -b option to
      extract)</IFD0:StripOffsets>
      <IFD0:SamplesPerPixel>3</IFD0:SamplesPerPixel>
<IFD0:RowsPerStrip>1</IFD0:RowsPerStrip>
<IFD0:StripByteCounts>(Binary data 28079 bytes, use -b option to
      extract)</IFD0:StripByteCounts>
<IFD0:XResolution>400</IFD0:XResolution>
<IFD0:YResolution>400</IFD0:YResolution>
<IFD0:PlanarConfiguration>Chunky</IFD0:PlanarConfiguration>
<ICC-header:ProfileCMMType>APPL</ICC-header:ProfileCMMType>
<ICC-header:ProfileVersion>2.2.0</ICC-header:ProfileVersion>
<ICC-header:ProfileClass>Display Device Profile</ICC-header:ProfileClass>
<ICC-header:ColorSpaceData>RGB </ICC-header:ColorSpaceData>
<ICC-header:ProfileConnectionSpace>XYZ </ICC-header:ProfileConnectionSpace>
<ICC-header:ProfileDateTime>2006:02:02 02:20:00</ICC-header:ProfileDateTime>
<ICC-header:ProfileFileSignature>acsp</ICC-header:ProfileFileSignature>
<ICC-header:PrimaryPlatform>Apple Computer Inc.</ICC-header:PrimaryPlatform>
<ICC-header:CMMFlags>Not Embedded, Independent</ICC-header:CMMFlags>
<ICC-header:DeviceManufacturer>none</ICC-header:DeviceManufacturer>
<ICC-header:DeviceModel></ICC-header:DeviceModel>
<ICC-header:DeviceAttributes>Reflective, Glossy, Positive, Color</ICC-
      header:DeviceAttributes>
<ICC-header:RenderingIntent>Perceptual</ICC-header:RenderingIntent>
<ICC-header:ConnectionSpaceIlluminant>0.9642 1 0.82491</ICC-
      header:ConnectionSpaceIlluminant>
<ICC-header:ProfileCreator>EPSO</ICC-header:ProfileCreator>
<ICC-header:ProfileID>0</ICC-header:ProfileID>
<ICC_Profile:ProfileDescription>EPSON  sRGB</ICC_Profile:ProfileDescription>
<ICC_Profile:RedMatrixColumn>0.43607 0.22249 0.01392</ICC_Profile:RedMatrixColumn>
<ICC_Profile:GreenMatrixColumn>0.38515 0.71687
      0.09708</ICC_Profile:GreenMatrixColumn>
<ICC_Profile:BlueMatrixColumn>0.14307 0.06061 0.7141</ICC_Profile:BlueMatrixColumn>
<ICC_Profile:MediaWhitePoint>0.95045 1 1.08905</ICC_Profile:MediaWhitePoint>
<ICC_Profile:ProfileCopyright>Copyright (c) SEIKO EPSON CORPORATION 2000 - 2006.
      All rights reserved.</ICC_Profile:ProfileCopyright>
<ICC_Profile:RedTRC>(Binary data 8204 bytes, use -b option to
      extract)</ICC_Profile:RedTRC>
<ICC_Profile:GreenTRC>(Binary data 8204 bytes, use -b option to
      extract)</ICC_Profile:GreenTRC>
<ICC_Profile:BlueTRC>(Binary data 8204 bytes, use -b option to
      extract)</ICC_Profile:BlueTRC>
<ICC_Profile:MediaBlackPoint>0 0 0</ICC_Profile:MediaBlackPoint>
```


APPENDIX B: SAMPLE CODE TO CONVERT PDF/A-2 BACK TO JPEG2000S

```
/* Assumption: The PDF/A-2b file was specifically generated from image objects
      converted from TIFF images with JPXDecode along with page-level metadata */

public static void parse(String src, String dest) throws IOException{
      PdfReader reader = new PdfReader(src);
      PdfObject obj;
      int counter = 0;
      for(int i = 1; i <= reader.getXrefSize(); i ++){
            obj = reader.getPdfObject(i);
            if(obj != null && obj.isStream()){
                  PRStream stream = (PRStream) obj;
                  byte[] b;
                  try{
                        b = PdfReader.getStreamBytes(stream);
                  }catch(UnsupportedPdfException e){
                        b = PdfReader.getStreamBytesRaw(stream);
                  }
            PdfObject pdfsubtype = stream.get(PdfName.SUBTYPE);
            FileOutputStream fos = null;

            if (pdfsubtype != null &&
                  pdfsubtype.toString().equals(PdfName.XML.toString())) {
                        fos = new FileOutputStream(String.format(dest + "_xml/" +
                              counter+".xml", i));
                        System.out.println("Page Metadata Extracted!");
                  }
            if (pdfsubtype != null &&
                  pdfsubtype.toString().equals(PdfName.IMAGE.toString())) {
                  counter ++;
                  fos = new FileOutputStream(String.format(dest + "_jp2/" +
                        counter+".jp2", i));
                  }
            if (fos != null) {
                  fos.write(b);
                  fos.flush();
                  fos.close();
                  System.out.println("JPEG2000s Conversion from PDF completed !");
            }
      }
}

/* Then Use ImageMagick library to convert JPEG2000s to TIFFs */
```


HTTPS://DOI.ORG/10.6017/ITAL.V37I1.9878

REFERENCES

[1] PDF-Tools.com and PDF Association, "PDF/A—The Standard for Long-Term Archiving," version 2.4, white paper, May 20, 2009, http://www.pdf-tools.com/public/downloads/whitepapers/whitepaper-pdfa.pdf; Duff Johnson, "White Paper: How to Implement PDF/A," Talking PDF, August 24, 2010, https://talkingpdf.org/white-paper-how-to-implement-pdfa/; Alexandra Oettler, "PDF/A in a Nutshell 2.0: PDF for Long-Term Archiving," Association for Digital Standards, 2013, https://www.pdfa.org/wp-content/until2016_uploads/2013/05/PDFA_in_a_Nutshell_211.pdf; Library of Congress, "PDF/A, PDF for Long-Term Preservation," last modified July 27, 2017, https://www.loc.gov/preservation/digital/formats/fdd/fdd000318.shtml.

[2] Library of Congress, "The Time and Place for PDF: An Interview with Duff Johnson of the PDF Association," *The Signal* (blog), December 12, 2017, https://blogs.loc.gov/thesignal/2017/12/the-time-and-place-for-pdf-an-interview-with-duff-johnson-of-the-pdf-association/.

[3] Yan Han, "Beyond TIFF and JPEG2000: PDF/A as an OAIS Submission Information Package Container," *Library Hi Tech* 33, no. 3 (2015): 409–23, https://doi.org/10.1108/LHT-06-2015-0068.

[4] Federal Agencies Digital Guidelines Initiative, *Technical Guidelines for Digitizing Cultural Heritage Materials*. (Washington, DC: Federal Agencies Digital Guidelines Initiative, 2016), http://www.digitizationguidelines.gov/guidelines/FADGI%20Federal%20%20Agencies%20Digital%20Guidelines%20Initiative-2016%20Final_rev1.pdf.

[5] Duff Johnson, "US Federal Agencies Approve PDF/A," PDF Association, September 2, 2016, http://www.pdfa.org/new/us-federal-agencies-approve-pdfa/.

[6] Bruno Lowagie, *iText in Action*, 2nd ed. (Stamford, CT: Manning, 2010).

[7] "iText 5.4.4," iText, last modified September 16, 2013, http://itextpdf.com/changelog/544.

[8] Timothy Robert Hart and Denise de Vries, "Metadata Provenance and Vulnerability," *Information Technology and Libraries* 36, no. 4 (2017), https://doi.org/10.6017/ital.v36i4.10146.

[9] Johan Van der Knijff, "JPEG 2000 for Long-Term Preservation: JP2 as a Preservation Format," *D-Lib* 17, no. 5/6 (2011), https://doi.org/10.1045/may2011-vanderknijff.

[10] PDF Association, "How veraPDF does PDF/A Validation," 2016, http://www.pdfa.org/how-verapdf-does-pdfa-validation/.