Lost in OCR-Translation: Pixel-based Text Reflow to the Rescue

Magnification of Archival Raster Image Documents in the Browser without Horizontal Scrolling

Frode Eika Sandnes
Oslo Metropolitan University, 0130 Oslo
frodes@oslomet.no

ABSTRACT

Low-vision users have been accustomed to viewing accessible electronic documents with reflow and responsive layouts that adapt to the user's text size requirements. However, mainstream technologies do not work optimally with image-based documents scanned from paper as well as documents that do not adhere to accessibility guidelines. Users are often dependent on viewing magnifications of such documents by scrolling in two-dimensions which most users find confusing. This paper discusses the capabilities of key state-of-the-art technologies and describes a browser-based document magnification implementation that reflow document contents at the pixel-level to prevent horizontal scrolling. The implementation provides low-vision users a practical alternative for simplified access to the content of archival documents with a different view than the state-of-the-art technologies. The paper also discusses practical and operational issues as well as unresolved challenges.

CCS CONCEPTS

• **Human-centered computing**; • **Accessibility**; • **Accessibility systems and tools**;

KEYWORDS

Accessibility, Low-vision, Reading, Magnification, Zoon, Reflow, Responsive layout, Text-size, Luminance contrast

ACM Reference Format:
Frode Eika Sandnes. 2022. Lost in OCR-Translation: Pixel-based Text Reflow to the Rescue: Magnification of Archival Raster Image Documents in the Browser without Horizontal Scrolling. In *The15th International Conference on PErvasive Technologies Related to Assistive Environments (PETRA '22), June 29–July 01, 2022, Corfu, Greece.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3529190.3534734

1 INTRODUCTION

Reduced visual acuity is one of the most prevalent forms of reduced human sensory function, especially as most individuals experience reduced vision as part of the natural aging process. Traditionally, low vision readers used magnifying glasses, or more recently electronic magnifiers, to read printed text. Only a portion of the magnified text is visible through the looking glass. To read text passages with Latin scripts the magnifier typically needs to be moved in a zig-zag motion from left-to-right along the lines, from top to bottom. This magnifying metaphor has been directly transferred to the digital domain with electronic screen magnifiers, document viewers, and early web browser technologies. However, it is difficult for users to orient themselves in the two-dimensional space as they do not have the same physical landmark cues and motor memory as with physical magnifiers that can be physically moved across a page. When viewing magnified documents typically the content is accessed by the means of horizontal and vertical scrolling leaning on the papyrus metaphor. However, papyrus scrolls were only scrolled along one dimension, and it has been pointed out that two-dimensional scrolling is a misguided use of the papyrus scroll metaphor.

The problems of two-dimensional scrolling are intuitive and a generally agreed-upon challenge among low-vision users (see for instance the technical note by Wayne E. Dick [7]). The W3C Web Content Accessibility Guidelines (WCAG) success criterion 1.4.10 address reflow and states that it should not be necessary to access web content by scrolling in two dimensions. This success criterion is also adopted for general electronic documents and graphical user interfaces in the standard EN 301.549 that applies to the public sector in the European Union as part of the Web Accessibility Directive (WAD). Accessibility legislation such as WAD applies to new content and does not affect existing archived content.

Current computer platforms provide several built-in functions to assist individuals with reduced vision. For instance, web browsers allow users to quickly adjust the text size to meet their viewing requirements. Many modern websites are responsive and adapt the content to the user's viewport width and text size configuration. Numerous blog posts and tutorials indicate that many web designers invest effort, interest, and pride in accessible responsive web design.

Web content is primarily intended for screen viewing. Hybrid documents are intended to serve screen reading as well as physical hardcopy paper reading. Several pdf-document viewers can display such content with different text sizes and viewport widths by reflowing the text dynamically. Acrobat Reader is one widely used pdf-reading application. However, experience shows that the reflow function in this document viewer can behave in an unpredictable manner with certain pdf documents that do not fully adhere to recent accessibility standards. Fortunately, commonly used software such as recent versions of Microsoft Word makes it easier for users to generate accessible pdf-documents that reflow well. However, there are still many older systems in use today, such as Latex, that by default generate pdf-documents that hinders effective reflow viewing.

In certain situations, documents are scanned from paper hardcopies and archived in pdf format, for example documents signed by a manager in a company. Such scanned raster image documents do not work with reflow functionality in Acrobat Reader. Note that

PETRA '22, June 29–July 01, 2022, Corfu, Greece
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9631-8/22/06.
https://doi.org/10.1145/3529190.3534734

the range of pdf-readers available were not surveyed as it is possible that some do provide reflow of raster image documents.

Optical character recognition (OCR) is occasionally used as a means of extracting the text from raster images in scanned documents. Modern OCR engines are powerful with high recognition rates, they cannot guarantee 100% text recognition rates. More importantly, much of the OCR research is targeting English and works in varying degrees for other languages with variations on the Latin script and other written languages [3]. Mathematics is another type of language that poses a challenge to OCR technology. It is not straightforward to convert mathematical expressions from raster images into semantically coded expressions. Even simple expressions such as a variable name with a subscript label can pose a challenge. The OCR engine may successfully recognize the variable name and the subscript label, but not register the subscript coding. A human reader will usually have very little trouble identifying a variable with a subscript given it is sufficiently magnified.

Microsoft has recently introduced a feature called PDF-reflow in a relatively recent version of Office. Users open a pdf document in Word. Microsoft pdf-reflow performs OCR on scanned documents and converts these into Word documents that can be edited and viewed according to users' needs. Microsoft pdf-reflow conserves layout features such as paragraphs, heading, images, double columns, international characters, and even simple mathematical equations. One may therefore argue that state-of-the-art OCR technology is indeed available to most users.

Another issue with many non-accessible pdf-documents is that the user is unable to override the reading colors. Even worse, in some situations, some text styles will respond to user settings while others will not. For example, body text may respond to user configurations while headings do not. Users should be able to adjust the colors to ensure sufficient contrast between the text and the background [13, 21–24]. Some low-vision users report that it is easier to read inverted text, and dark-mode has become a widely embraced mode for users in general [20] as it is believed to be less straining on the eyes. Readability has also been connected to issues such as line spacing [9–11]. Obviously, the line spacing in a scanned document cannot be altered using a simple document viewer.

Although making all new pdf-documents accessible is a laudable goal it is unrealistic to make existing pdf-documents in various archives compliant with recent accessibility guidelines. For instance, academics occasionally need to read classic research papers where the source files are no longer available.

The purpose of this project was therefore to revisit an existing idea from the literature [3, 18] that has received limited attention, and making it available to low-vision users. The goal is to provide a simple and easy means of reading archival pdf documents with reflow. Instead of treating text as text, texts are treated as images. Each word is identified as an image and repositioned relative to its preceding and succeeding neighbors. With this approach long lines can be broken to fit the viewport width. Moreover, the formatting of individual words is preserved. The process of converting images to text is avoided, thereby eliminating the error-prone task of identifying formatting, mathematics, and special characters. For instance, mathematical texts often contain non-alphabetic symbols such as Greed letters and operators. Another goal was to make use of modern web browsers as these serve as universal document viewing tools familiar to most users.

This paper refers to low vision users as individuals that prefer the visual channel over other modalities. Yet, text needs to be enlarged due to the reduced visual acuity. The implementation does not address the needs of screen reader users, that is, individuals without visual sensory function or individuals that prefer audio or tactile modalities.

This paper is organized as follows: The next section provides a review of related work. Then, the browser-based magnification implementation is detailed. Finally, the performance of the magnifier is discussed.

2 RELATED WORK

The issue of text reflow can probably be considered a relatively straightforward engineering problem. This could explain why there are few academic texts addressing both the problems of two-dimensional scrolling and compensating technical solutions. Of the few available studies Breuel [3] presented the idea of reflowing raster images of text on the web nearly 20 years ago. He pointed out the challenges with formatting being lost in the OCR process. However, the web technology at the time did not provide the same possibilities as that of current web technology.

In fact, ideas connected to analyzing printed layouts for improved electronic viewing goes back at least 30 years (see for instance [17]). Through the last decades, general document segmentation has been an active topic of research [6]. Ittner and Baird [14] argues that at pixel level text can be processed without knowledge about the language such as reading direction and script type.

Panjwani, Uppal and Cutrell [18] revisited the problem ten years ago using similar arguments to the ones used herein. They termed the process script agnostic reflow, in that the details of the content are preserved when reflowing word images rather than the text. However, the system documented by the authors does not appear to be available to users. Web technology has also matured over the decade perhaps making the proposition more practical and realistic today compared to a decade ago.

Projections are commonly used in image analysis algorithms and were also used in the reflow algorithm by Panjwan et al. [18]. This study assumes that the lines are horizontal which is the case with most electronic documents. Algorithms have also been proposed for arbitrary text line angles such as found in handwritten texts [8]. Detection of the text angle has also been discussed extensively in the context of OCR research [19, 26].

Related to the issue of reflow is the research into annotation of text with reflow [1, 5] as active reading often involves note taking and scribbling directly onto the text. The technical problem is thus to ensure that handwritten annotations in documents are preserved and understandable to readers when text moves around because of reflow. Choudhury et al. [4] addressed several issues related to contextual reflow of html elements.

This study rests on the assumption that users prefer not to scroll in two dimensions as argued by Dick [7]. In addition, Hallet et al. did a controlled experiment with 16 participants comparing zoom magnification and responsive layout [12]. Their results support the general belief that responsive layout is preferable as it led to shorter task completion times. Although not addressing two-dimensional scrolling per se, Öquist and Lundin [29] did a controlled experiment on reading through a small mobile display. They explored paging, scrolling, leading and Rapid Visual Serial Presentation (RSVP). Their results showed that paging led to the fastest reading speeds yet with no significant differences in comprehension.

Billah et al. [2] pointed out that in addition to two-dimensional scrolling screen magnifiers occupy valuable screen real-estate. They proposed a simplified magnification model where zoomed versions of important regions are accessed one-dimensionally using a dial. Hence, the complex two-dimensional scrolling of the magnified zoom window is reduced to a conceptually simpler one-dimensional scrolling task. They observed a 20% web browsing performance increase.

Lee, Ashok and Ramakrishnan [15] studied low-vision use of magnification in office applications. Their results showed that the distance between the controls such as menu commands and the editing areas were an issue for users of screen magnifiers. They concluded that application controls should be placed closer to the elements they control to reduce screen-travel.

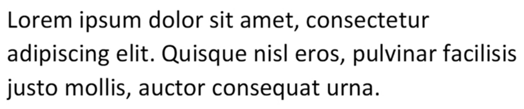
Handheld magnifiers face similar challenges as document viewing of scanned documents as the viewer must move the magnifier device in two dimensions to view text on a surface. Several alternative concepts have been explored such as tracking a finger on the page [27] and device tilting [16]. In an attempt not to overload viewers Woodruff, Landay and Stonebraker [28] proposed the idea of keeping the information density fixed in interfaces that relies on zooming. Their proposal was demonstrated for a geoinformation visualization application.

3 MAGNIFICATION WITH PIXEL REFLOW

3.1 Browser platform

The magnifier implementation presented herein is oriented around the browser for three main reasons. First, we rely on users' experience and existing skills using web-browsers. It is assumed that low-vision users know how to navigate a browser document with the mouse, and how to dynamically adjust the text size (typically with CTRL+/-). Second, we rely on the web-browser capabilities of reflowing the contents to fit the viewport width and user's preferred text size. Modern browsers perform such tasks effectively. Third, the browser is a pervasive component available to most users. There is no need to install a particular software or app which can be challenging in some workplaces due to computer security policies that prohibit installing third party software. The JavaScript implementation is released with an open-source license (see https://github.com/frode-sandnes/PDF-FLOWER) and only relies on other open-source libraries.

The magnifier takes document images as input and converts these to a sequence of images representing the individual words and other graphical elements which can subsequently be presented in the browser according to the visual requirements of the user. The implementation accepts pdf-documents (portable document format) as input and first converts these to a set of images – one image per page. For pdf document reading and rendering the PDF.js library was used (see https://mozilla.github.io/pdf.js/).



Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Quisque nisl eros, pulvinar facilisis
justo mollis, auctor consequat urna.

Figure 1: Image of text

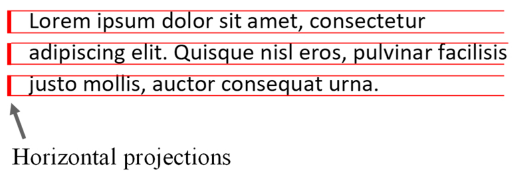

Lorem ipsum dolor sit amet, consectetur
adipiscing elit. Quisque nisl eros, pulvinar facilisis
justo mollis, auctor consequat urna.
Horizontal projections

Figure 2: Line detection with horizontal projections

3.2 Word image extraction

Each image is converted to an ordered sequence of images of the individual words. Two approaches were taken. First, the Tesseract.js library (see https://github.com/naptha/tesseract.js) was used for extracting the text contents. Tesseract.js a mature JavaScript implementation of the Tesseract optical character library. Although the main purpose of Tesseract is to extract the actual text it also provides useful information about the text such as the bounding boxes of the lines on the page and the bounding boxes of each individual word.

Although Tesseract.js is a robust general-purpose library it is also computationally intensive. Tesseract's processing delays are probably beyond the threshold that most users will tolerate in a practical use context. A simpler engine for extracting the word bounding boxes was therefore implemented from scratch through direct pixel access of html canvas elements (see Figure 1). First, the bounds of the text lines were identified using vertical projections of the image pixels (see Figure 2). A projection with pixels set indicates text, while a projection with the pixel unset indicates spacing. The regions of text and space are thus identified by tracing consecutive sequences of set and unset pixels, respectively.

Next, each line is split into words using the horizontal projection of the pixels within the bounding box of the line (see Figure 3). It is assumed that the lines are horizontal. Again, spacings are identified by consecutive sequences of unset pixels. A typical line will contain two types of spaces, namely small spacings between letters within words, and larger spacings between words. To separate the two the largest spacing is identified and a spacing threshold is set to half the largest spacing detected on a given line. Spacings larger than the threshold are used to indicate word boundaries, and spacings below the threshold are ignored. Once the bounding box of each word is identified the respective word sub-image can be extracted from the page (see Figure 4).

3.3 Text and background color

The approach described herein assumes limited use of colors which is a realistic assumption for archival documents from the era of black-and-white printers. Black and white documents usually contain tones of gray and dithering for the presentation of anti-aliased

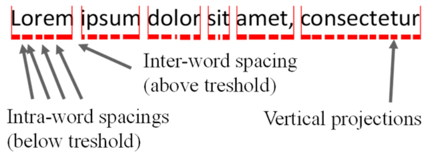

Lorem ipsum dolor sit amet, consectetur
Inter-word spacing (above treshold)
Intra-word spacings (below treshold)
Vertical projections

Figure 3: Word detection with vertical projections. Smaller intra-word spacings are ignored, the larger spacing between words are used.

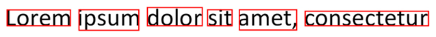


Figure 4: Word bounding boxes

characters. A threshold function is therefore used to separate pixels into foreground (text) and background, as set and unset, respectively. Although most documents present dark text on a white background a check is performed to identify the background and foreground color. These are identified by taking a small snapshot of the diagonal pixels in a central region of the document. The two most extreme colors in the color space are identified and the most frequent of the two is classified as the background, and the other is classified as the foreground text color.

3.4 Paragraph spacing

Spacing helps make text easier to read [9–11]. Spacing gives the reader clues about the structure and how text is divided into paragraphs and simplifies navigating the text and provides opportunities for pausing. Simple rules for identifying potential paragraph boundaries were deployed. Whenever a potential paragraph boundary was detected a line break with spacing was inserted into the stream of image words helping to break up long passages of text. A range of formatting styles exist, and the following rules were used as they cover a large portion of documents although not all:

Larger spacing between lines: If a space between two lines is larger than the typical line spacing the spacing is classified as a paragraph boundary. A line break is detected between two consecutive words if the x-coordinate of the bounding box of the first word is larger than the x-coordinate of the bounding box of the preceding word.

Indented paragraphs: If a line is indented it is classified as a paragraph boundary. Indentations are identified as follows: First the margin is estimated by finding the word bounding box on the page with the smallest x-coordinate. Then, if a word starts at an offset that is equal to the height of the word bounding box it is defined as being indented.

3.5 Two-column documents

Most documents are presented in a single column, but two column documents are also quite common, especially in academic papers. If the simple left-to-right, top-to-bottom approach will therefore mingle together words from the two columns. A simple approach for handling two column documents was devised.

First, the pixels along the vertical midline that divides each page into two parts are analyzed. If this midline contains successive unset pixels that exceed a fifth of the page height, the page is classified as having two columns. Then the left and right parts of the regions with the unset midline are classified as left and right columns, while the other parts of the document are classified as single column parts. Next, the word extraction analysis outlined above is applied to each of the identified regions. This approach allows a page to contain several two-column and single column regions which is commonly found in academic papers such as those hosted by ACM Digital Library and IEEE Xplore.

3.6 Negative text and contrast enhancements

It is usually not straightforward to read raster documents with negative text in an ordinary document reader. However, these operations are easy to implement when managing the documents at the pixel level. options for negative text and contrast enhancements were therefore implemented (see Figure 2 (e) and (f)). Negative text is simply achieved by inverting the intensity value of each pixel. Simple contrast enhancements were implemented using a simple linear compression of the intensity dynamic range.

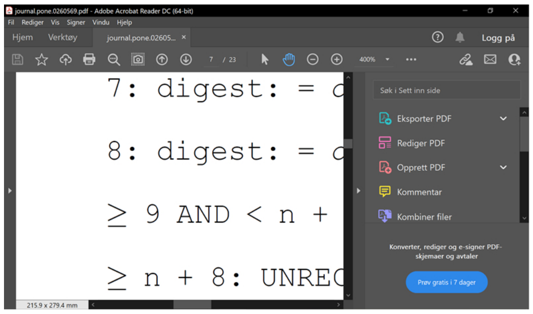
4 PRELIMINARY RESULTS AND DISCUSSION

Although tests involving actual users is the gold standard when assessing new technology, it was decided to perform technical assessments for this report. This is because text reflow is a relatively agreed-upon goal among low-vision users. The challenge has been that the technology has not yet lived up to the needs of the users. The document viewer is currently still being developed. User testing and validation is planned with a more streamlined implementation that can be used as an authentic reading tool in practice. Active users of magnification technology will be recruited specifically.

Testing was performed during development on an arbitrary selection of documents with different types of layouts. Examples include documents from different publishers with specific "house"-styles, image-rich documents, text-heavy documents, documents with equations, single and double column, archival scanned documents published before 1970, recent non-accessible pdf-documents, etc. Also, accessible pdf-documents were included in the tests since these are processed in the same manner at pixel level. Assessments were performed by simple visual inspection of the results to check that words were successfully reflowed (absence of a horizontal scrollbar), just zoomed presence of a horizontal scrollbar), and identification of other processing mistakes that makes the results hard to read.

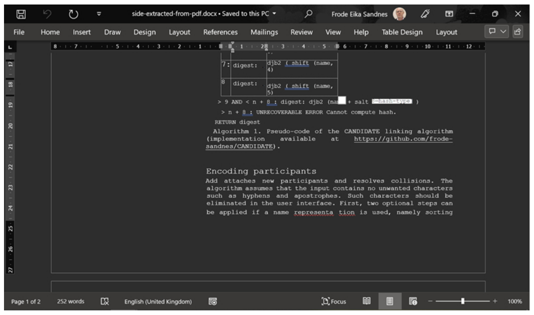
4.1 Comparing document viewing, OCR and pixel-reflow

The following examples illustrate key benefits and drawbacks of three magnification approaches exemplified. Ultimately, it would be beneficial to illustrate the approaches with an authentic archival paper. However, many relevant archival documents are copyright protected. Therefore, one of the author's own papers was used [25] in this example as it is published open access with a Creative Commons BY license. Fragments of the paper can thus be reproduced



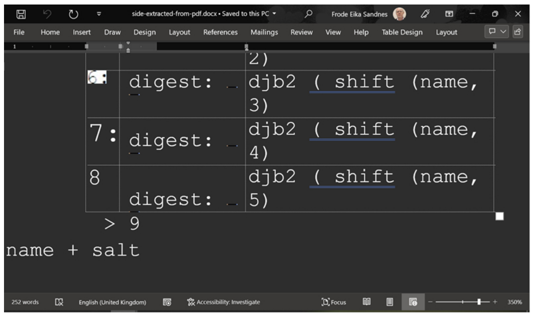
```
7: digest: = d
8: digest: = d
≥ 9 AND < n +
≥ n + 8: UNREC
```

Figure 5: Document viewer (Acrobat) with 400% zoom.



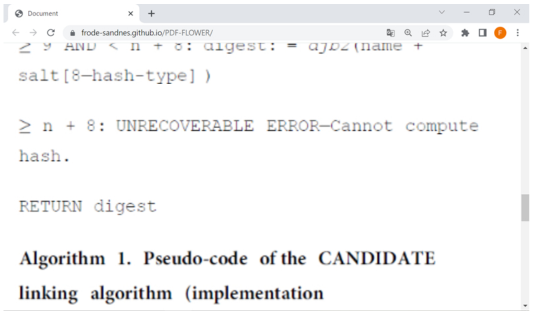
```
7: digest:   djb2 ( shift (name,
             4)
8  digest:   djb2 ( shift (name,
             5)
 > 9 AND < n + 8 : digest: djb2 (na + salt 8-hash-type )
  > n + 8 : UNRECOVERABLE ERROR Cannot compute hash.
RETURN digest
 Algorithm 1. Pseudo-code of the CANDIDATE linking algorithm
(implementation available at https://github.com/frode-
sandnes/CANDIDATE).

Encoding participants
Add attaches new participants and resolves collisions. The
algorithm assumes that the input contains no unwanted characters
such as hyphens and apostrophes. Such characters should be
eliminated in the user interface. First, two optional steps can
be applied if a name representa tion is used, namely sorting
```

Figure 6: OCR'ed document (Microsoft Word pdf-reflow) with 100% zoom (default).



```
             2)
6:  digest:  djb2 ( shift (name,
             3)
7:  digest:  djb2 ( shift (name,
             4)
8   digest:  djb2 ( shift (name,
             5)
  > 9
name + salt
```

Figure 7: OCR'ed document in web viewing mode (Microsoft Word) with 350% zoom.



```
≥ 9 AND < n + 8: digest: = djb2(name +
salt[8—hash-type] )

≥ n + 8: UNRECOVERABLE ERROR—Cannot compute
hash.

RETURN digest
```

Algorithm 1. Pseudo-code of the CANDIDATE linking algorithm (implementation

Figure 8: Pixel-based reflow (200% browser zoom).

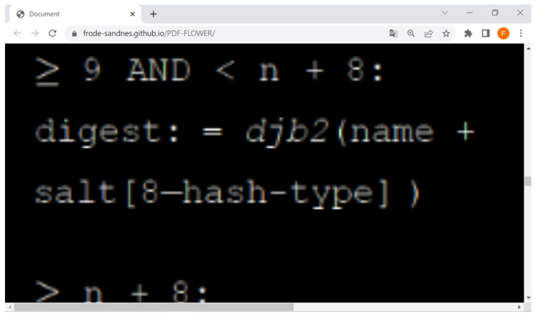
herein without copyright infringement. First the paper was rasterized to simulate the scan, i.e., each page converted to an image. This was done digitally, but could have also been performed by printing the paper and then scanning the printed hardcopy. The paper was chosen as it contains algorithm fractions that were expected to cause OCR errors. An arbitrary selected extract from page 7 was selected as it contains both algorithm and text fragments.

Figure 5 shows the document as viewed in Adobe Acrobat Reader with 400% magnification. One key benefit of the document viewer is that the reader sees the document in its original unaltered form. The user can therefore use the visual details to decipher the contents such as the algorithm. The user will also see the raster aliasing artifacts. A key drawback is that the reflow functionality does not work on this raster example. The user will thus have to scroll in two dimensions to within columns and across columns. Moreover, the document viewer does not allow the user to override the text and background colors.

Figure 6 shows the same document imported into Microsoft Word using pdf-reflow. The result shows that the OCR engine successfully recognizes most of the text as well as the formatting. To read the text with reflow the user may view the document in Web view as shown in Figure 7. Obviously, the text is more readable when converted with OCR as the raster artifacts are removed. Another benefit is that the users personalized dark mode ensures the document is automatically viewed in negative mode according to the user preferences. Microsoft pdf-reflow also appears to be sufficiently fast for practical use.

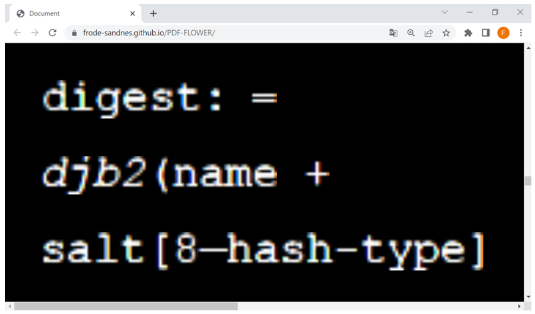
However, Figures 6 and 7 also reveal several OCR errors. First, the algorithm is interpreted as a table. Second, some of the symbols were not detected and were instead replaced by the original image. These mistakes stand out in the dark mode view as they have maintained the original white background. The OCR engine has also included some pixel noise as separate images. In this example, the result of the Word pdf-reflow is understandable as no vital information was lost. The tests perform on authentic archival papers showed that Microsoft pdf-reflow was unable to handle more complex equations, Greek letters, and other special symbols, as well as documents with a non-white background. In some instances some elements were simply replaced by blank space. In sum, this confirms that it is generally challenging to overcome the limitation of OCR-based magnification methodologies.

Figures 8, 9 and 10 show the document using the pixel-based reflow method presented herein, viewed in a Chrome web browser with text sizes set to 200%, 400% and 500%, respectively. Clearly, the words wrap as expected according to the viewport width and the selected text size. The user navigates through the document by scrolling vertically (using the mouse scroll wheel) as the two columns were successfully converted in the same stream of text. Also, the words in the algorithm are wrapped across several lines. Obviously, the original raster aliasing artifacts are present, but no errors were introduced. Figures 9 and 10 illustrate the document presented in negative. In addition, Figure 10 illustrates contrast enhancements.



```
≥ 9 AND < n + 8:
digest: = djb2(name +
salt[8—hash-type] )

> n + 8:
```

Figure 9: Pixel-based reflow with negative text (400% browser zoom).



```
digest: =
djb2(name +
salt[8—hash-type]
```

Figure 10: Pixel-based reflow with negative text and contrast enhancement (500% browser zoom).

4.2 Practical and operational issues

Assistive technology needs to be pragmatic and practical to use in daily operation. One key issue is that of processing time. Timing measurements using a low-spec laptop computer showed that the pixel-based processing in the browser took around 4-5 milliseconds per word. In contrast, the Tesseract based word wrapping took about 10 times as long with more than 30 milliseconds per word. Using the pixel-based implementation a page took typically about 4 seconds to process (around 800 words). To process a 10-page document takes about 40 seconds.

The time to process a document is relatively long. A function to save a copy of the reflowable html document was therefore created such that the processing can be eliminated if the document is to be reopened again.

The word images were integrated in the generated html. Typically, around 2 Kb are needed to store the image of each word. Just under 2 Mb are needed to store a typical text-oriented page and 20 Mb are needed for a 10-page document. By reducing the image quality, the size is reduced by about two thirds.

4.3 Unresolved issues

Currently, the page analysis implementation is very rudimentary. For instance, it is not capable of handling layouts with more than two columns, layouts with different regions in multiple colors or more sophisticated layouts found in some magazines. However, such elaborate layouts are probably more likely to be present in more recent documents that adhere to accessibility standards. The simple model implemented is however capable of managing layouts found in academic texts such as conference and journal papers with single or two column formats. The following issues were identified, but has not yet been addressed, for instance:

Vertical decorations: Vertical decorations such as lines prevent the line segmentation algorithm from successfully identifying the text lines. Blocks of multiple lines with vertical decorative elements will not be reflowed. Users must use horizontal scrolling to read such parts.

Hanging indents: Hanging intends are interpreted as paragraph markers if there is no extra vertical space between the previous paragraph and extra vertical space is thus inserted in the middle of the paragraph. Hanging intends are sometimes used without extra vertical spacing in reference lists. This extra space reduces the visual aesthetics but does not affect the content. Accurate representations and convenience are probably preferable to aesthetics.

Diagrams with disconnected elements: The approach typically will place images and figures in separate flowable blocks. However, illustrations with elements that are not connected are segmented into multiple figures that reflow unintentionally. Such images may be misleading.

4.4 Future directions

One possibility is to employ a hybrid approach combining OCR and pixel-based reflow. For example, the Tesseract library assigns a confidence score to every word. One could for instance use the text of words recognized with a high confidence score and make use of the image otherwise. Microsoft pdf-reflow demonstrates that it is possible to capture basic formatting changes such as italics, boldface, font changes, etc., with OCR technology.

5 CONCLUSIONS

The issue of reflow of magnified text in archival documents was addressed. Support for low vision users seems adequate when documents satisfy recent accessibility standards. However, reading of archival scanned documents is still a challenge for low-vision users even with state-of-the-art tools. Of the state-of-the-art tools explored, Microsoft pdf-reflow seemed to provide the most practical and robust results with scanned documents, but still introduced OCR errors with special symbols and mathematics. It also requires the user to have a software license. This probably lesser-known function in Microsoft Word will probably suffice for many use cases and has the potential of benefitting users that prefer non-visual modalities. Still, the OCR based approach fail for certain types of documents. The existing idea of pixel-based reflow still deserves further exploration as an alternative to OCR-based methods as it can handle documents which are difficult to process with OCR technology. Humans are very good at interpreting visual stimuli if these can be perceived sufficiently. The implementation described herein was successfully tested on a small sample of archival academic papers. Still, more work is needed to cope with more exotic layouts, tables, images, and illustrations. The pixel-based pdf viewer can be accessed at: https://frode-sandnes.github.io/PDF-FLOWER/. The source code is available at: https://github.com/frode-sandnes/PDF-FLOWER.

REFERENCES

[1] David Bargeron and Tomer Moscovich. 2003. Reflowing digital ink annotations. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03). Association for Computing Machinery, New York, NY, USA, 385–393. DOI:https://doi.org/10.1145/642611.642678

[2] Syed Masum Billah, Vikas Ashok, Donald E. Porter, and I.V. Ramakrishnan. 2018. SteeringWheel: A Locality-Preserving Magnification Interface for Low Vision Web Browsing. Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. Association for Computing Machinery, New York, NY, USA, Paper 20, 1–13. DOI:https://doi.org/10.1145/3173574.3173594

[3] Thomas Breuel. 2003. Reflowable document images for the Web. In Proceedings of WDA 2003, the 2nd International Workshop on Web Document Analysis.

[4] Rajorshi G. Choudhury, Arohi Kumar, Devesh Varshney, Prachi Jain, and Varun Malhotra. 2013. Context Sensitive Reflow of HTML Objects. In Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) - Volume 01 (WI-IAT '13). IEEE Computer Society, USA, 147–153. DOI:https://doi.org/10.1109/WI-IAT.2013.22

[5] Kevin Conroy. 2004. Digital Document Annotation and Reflow. Thesis, University of Maryland at College Park.

[6] Praveen Dasigi, Raman Jain, and C. V. Jawahar. 2008. Document image segmentation as a spectral partitioning problem. In Proceedings of 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing. IEEE, USA, 305-312.

[7] Wayne E. Dick. 2017. Operational Overhead Caused by Horizontal Scrolling Text. Technical note: Accessed 19/03/2022: http://nosetothepage.org/Fitz/2dScroll.html

[8] Xiaojun Du, Wumo Pan, and Tien D. Bui. 2009. Text line segmentation in handwritten documents using Mumford–Shah model. Pattern Recognition 42, 12 (2009), 3136-3145.

[9] Evelyn Eika. 2016. Universally designed text on the web: towards readability criteria based on antipatterns. Stud. Health Technol. Inform. 229, 461-470.

[10] Evelyn Eika and Frode Eika Sandnes. 2016. Authoring WCAG2. 0-compliant texts for the web through text readability visualization. In Proceedings of International Conference on Universal Access in Human-Computer Interaction. Springer, Cham, 49-58.

[11] Evelyn Eika and Frode Eika Sandnes. 2016. Assessing the reading level of web texts for WCAG2. 0 compliance—can it be done automatically?. In Proceedings of Advances in Design for Inclusion. Springer, Cham, 361-371.

[12] Elyse C. Hallett, Blake Arnsdorff, John Sweet, Zach Roberts, Wayne Dick, Tom Jewett and Kim-Phuong L. Vu. 2015. The usability of magnification methods: A comparative study between screen magnifiers and responsive web design. In Proceedings of International Conference on Human Interface and the Management of Information. Springer, Cham, 181-189. DOI: https://doi.org/10.1007/978-3-319-20612-7_18

[13] Fredrik Hansen, Josef Jan Krivan, and Frode Eika Sandnes. 2019. Still Not Readable? An Interactive Tool for Recommending Color Pairs with Sufficient Contrast based on Existing Visual Designs. In The 21st International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '19). Association for Computing Machinery, New York, NY, USA, 636–638. DOI:https://doi.org/10.1145/3308561.3354585

[14] David J. Ittner and Henry S. Baird. 1993. Language-free layout analysis. In Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR'93). IEEE, USA, 336-340.

[15] Hae-Na Lee, Vikas Ashok, and IV Ramakrishnan. 2021. Bringing Things Closer: Enhancing Low-Vision Interaction Experience with Office Productivity Applications. Proc. ACM Hum.-Comput. Interact. 5, EICS, Article 197. DOI:https://doi.org/10.1145/3457144

[16] Farhani Momotaz and Syed Masum Billah. 2021. Tilt-Explore: Making Tilt Gestures Usable for Low-Vision Smartphone Users. In The 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21). Association for Computing Machinery, New York, NY, USA, 1154–1168. DOI:https://doi.org/10.1145/3472749.3474813

[17] George Nagy, Sharad Seth, and Mahesh Viswanathan. 1992. A prototype document image analysis system for technical journals. Computer 25, 7 (1992), 10-22.

[18] Saurabh Panjwani, Abhinav Uppal, and Edward Cutrell. 2011. Script-agnostic reflow of text in document images. In Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '11). Association for Computing Machinery, New York, NY, USA, 299–302. DOI:https://doi.org/10.1145/2037373.2037419

[19] Subhash Panwar and Neeta Nain. 2012. A novel approach of skew normalization for handwritten text lines and words. In 2012 Eighth International Conference on Signal Image Technology and Internet Based Systems. IEEE, USA, 296-299.

[20] Lasse Apalnes Pedersen, Svavar Skuli Einarsson, Fredrik Arne Rikheim, and Frode Eika Sandnes. 2020. User interfaces in dark mode during daytime–improved productivity or just cool-looking?. In International Conference on Human-Computer Interaction. Springer, Cham, 178-187.

[21] Frode Eika Sandnes and Anqi Zhao. 2015. An interactive color picker that ensures WCAG2. 0 compliant color contrast levels. Procedia Computer Science, 67, 87-94.

[22] Frode Eika Sandnes. 2016. Understanding WCAG2. 0 color contrast requirements through 3D color space visualization. Stud. Health Technol. Inform. 229, 366-375.

[23] Frode Eika Sandnes. 2018. An image-based visual strategy for working with color contrasts during design. In International Conference on Computers Helping People with Special Needs. Springer, Cham, 35-42.

[24] Frode Eika Sandnes. 2021. Inverse Color Contrast Checker: Automatically Suggesting Color Adjustments that meet Contrast Requirements on the Web. In The 23rd International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '21). Association for Computing Machinery, New York, NY, USA, Article 72, 1–4. DOI:https://doi.org/10.1145/3441852.3476529

[25] Frode Eika Sandnes. 2021. CANDIDATE: A tool for generating anonymous participant-linking IDs in multi-session studies. PloS one 16.12: e0260569. https://doi.org/10.1371/journal.pone.0260569

[26] Ray Smith. 1995. A simple and efficient skew detection algorithm via text row accumulation. In Proceedings of 3rd International Conference on Document Analysis and Recognition. IEEE, USA, Vol. 2, 1145-1148.

[27] Alireza Shoaei Shirehjini and Frode Eika Sandnes. 2020. Using smartphones as magnifying devices: a comparison of reading surface finger tracking and device panning. In Proceedings of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments (PETRA '20). Association for Computing Machinery, New York, NY, USA, Article 79, 1–2. DOI:https://doi.org/10.1145/3389189.3397645

[28] Allison Woodruff, James Landay, and Michael Stonebraker. 1998. Constant information density in zoomable interfaces. In Proceedings of the working conference on Advanced visual interfaces (AVI '98). Association for Computing Machinery, New York, NY, USA, 57–65. DOI:https://doi.org/10.1145/948496.948505

[29] Gustav Öquist and Kristin Lundin. 2007. Eye movement study of reading text on a mobile phone using paging, scrolling, leading, and RSVP. In Proceedings of the 6th international conference on Mobile and ubiquitous multimedia (MUM '07). Association for Computing Machinery, New York, NY, USA, 176–183. DOI:https://doi.org/10.1145/1329469.1329493