Enhancing Readability of Scanned Picture Books

Chang Hu
Computer Science Department
Human-Computer Interaction Lab
University of Maryland
College Park MD 20742 USA
changhu@cs.umd.edu

ABSTRACT
I describe a system that enhances the readability of scanned picture books. Motivated by our website of children's books in the International Children's Digital Library, the system separates textual from visual content which decreases the size of the image files (since their quality can be lower) while increasing the quality of the text by displaying it as computer-generated text instead of an image. This text-background separation combines image processing and human validation in an efficient manner and results in a system that not only is more readable, but also accessible, searchable, and translatable.

Categories and Subject Descriptors
H5.2 [**Information interfaces and presentation**]: User Interfaces. - Graphical user interfaces.

General Terms
Design, Human Factors.

Keywords
Children's books, Accessibility, Online access, Readability, Digital Libraries.

1. INTRODUCTION
The International Children's Digital Library (ICDL) is an online digital library that offers free scanned children's books [1]. It includes over 2,500 books in 41 languages. It has been visited by over two million unique users from 166 different countries in the past five years. The book pages in the ICDL collection are scanned images of their physical counterparts. They are shown in users' web browsers as images. While the scanned books provide users worldwide with an abundance of children's books, the pages suffer from readability problems. There are several reasons for this. The text in the scanned pages is often too small to read on the small screens that many visitors use. The images, until recently, were scaled dynamically by the browser to fit in the browser window without scrolling – but it turns out that most browsers use a fast, but very poor quality image scaling algorithm (Figure 1). The ICDL website allows users to zoom into images, but this solution is problematic from a usability perspective since users must scroll in two dimensions. And from a task perspective, the readers must choose between seeing the full page of the picture book, which is valuable in itself, and zooming in far enough to make the text readable.

These problems apply to all digital libraries where scanned pages are displayed – but they are particularly significant for pages with images or other visual displays that are designed to be seen in concert with the text. Thus, these issues are particularly important for the ICDL, which has a large collection of picture books.

For scanned pages, the human reader has different needs regarding the resolution of the text and pictures. From a reader's perspective, lower resolution and aliasing artifacts are less problematic for pictures and background images. On the other hand, textual content in images becomes difficult to read with even the slightest aliasing, which can occur at a relatively high resolution. To enhance the overall reading experience while still preserving the context, our solution is to separate text and background on the pages, so that they do not have to be displayed at the same resolutions. The text-background separation also offers the potential to add other major features such as content search, read-aloud for increased accessibility and translation.

I built an interface called ClearText [3] which takes advantage of this separation, presented as a simple web-based interface for presenting books with these two layers. Standard DHTML with CSS is used to display the text-free image with HTML-rendered text on top of that image (Figure 2). I also built another interface, PopoutText, which simply renders a high-resolution version of the textual part of a scanned page on top of its original position on demand. PopoutText uses some of the same underlying technology I developed for ClearText, but is simpler (and less powerful) as it doesn't separate the textual and image layers. Our user study [3] showed that both ClearText and PopoutText are significantly better than the original interface in terms of both readability and preservation of author's creative intent. In fact, the two interfaces are close to physical books on these criteria.

In order to separate the textual and image layers, there are two key problems: 1) locating and transcribing the text so it can be

Figure 1. This shows: (left) the effect of browser scaling of images; and (right) pre-computing an image of exactly the right size that fits the browser window.

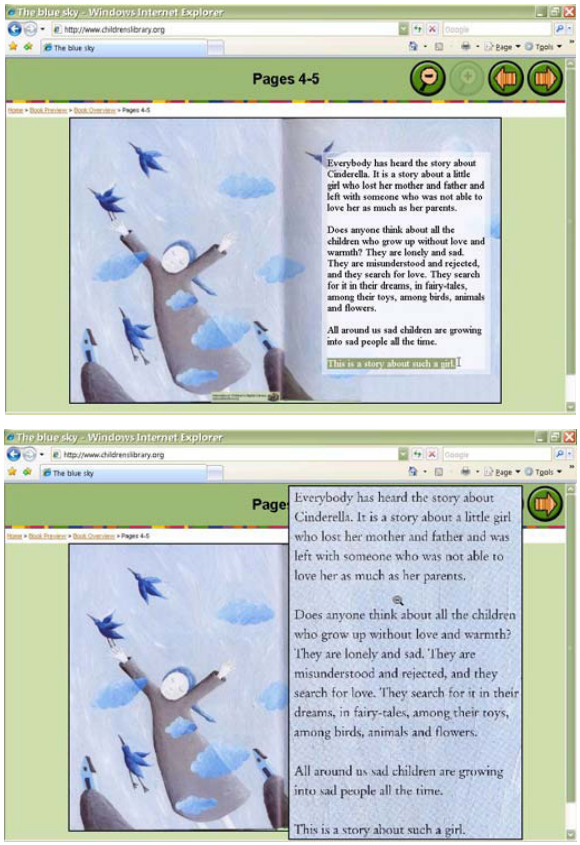

Figure 2. Reader interfaces.
(Top) ClearText. (Bottom) PopoutText

displayed as computer text on top of the scanned image, and 2) removing the text from the scanned image so the new text can be displayed clearly.

The most obvious solution to the first problem is to use Optical Character Recognition (OCR) which is the industry standard approach and works well in many settings [2]. However, it is difficult to apply OCR directly to picture books in the ICDL due to language, background and layout issues. First of all, OCR systems usually assume one or a small number of working languages, and the working languages are mostly Latin-based [10-15]. The ICDL has books in 41 languages, in which 33% are written in non-Latin character sets. Its second and third largest languages, Persian and Mongolian, are not Latin-based. Using one OCR system per language is both inefficient and very expensive, and using one OCR system for many languages decreases accuracy significantly.

Aside from language issues, most OCR systems assume that pages have a simple white background, whereas many picture books have background illustrations behind the text. Another common assumption of OCR is a simple layout with traditional fonts, which is often not true for books with more artistic representations. For example, many books have a large graphic letter at the beginning of a chapter which is usually not handled by OCR software.

I avoid these challenges with OCR by using human volunteers to transcribe the books. Because our books tend to be short without huge amounts of text, and because I have many volunteers, this approach is feasible. Or, if I had longer, more textual books, I more likely would have explored the use of a combined OCR / human approach where OCR does a first pass, and humans correct the results like the Distribute Proofreaders project [7].

However, for locating the text, and ensuring that it has been removed properly, I do use a semi-automatic system that combines an automated first pass with human correction step – a process sometimes referred to as Distributed Human Computation.

To solve the second problem (removing text from the image), I use an image processing technique called inpainting [9] that mends the texture of a region by imitating the texture of the neighboring region recursively. Overall, our approach obtains language-independent text-background separation on scanned pages with complex pictorial backgrounds.

2. RELATED WORK

Our user study [3] showed that book pages with text rendered at different scales are closer to physical books in terms of readability and preservation of creative content. They are also superior in readability than plain scanned images of the same book pages. As for computer-generated text, it has also been reported in [4] that readers prefer anti-aliased text to bitmap fonts at the same point size.

Commercial systems for Latin-based languages abound. However, multilingual OCR, layout analysis and text location on complex backgrounds remain to be open problems. Kangugo et al. [16] has given a comparative evaluation on OCR systems in different languages, which showed that OCR accuracy for some common languages is not satisfactory. A survey of layout analysis algorithms has been given by Mao et al. [17]. Most application domains mentioned are documents with simple layout, e.g. technical reports, journals and book pages. Jung et al. [18] wrote a survey on text detection and location, in which complex backgrounds are one of the main factors calling for more investigation.

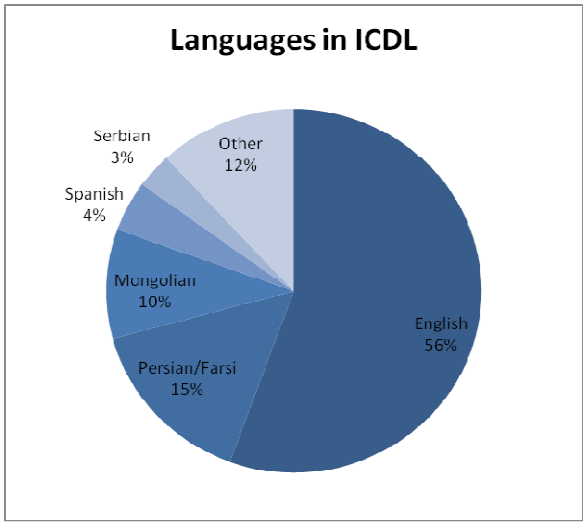

Figure 3. Percentage of books in different languages

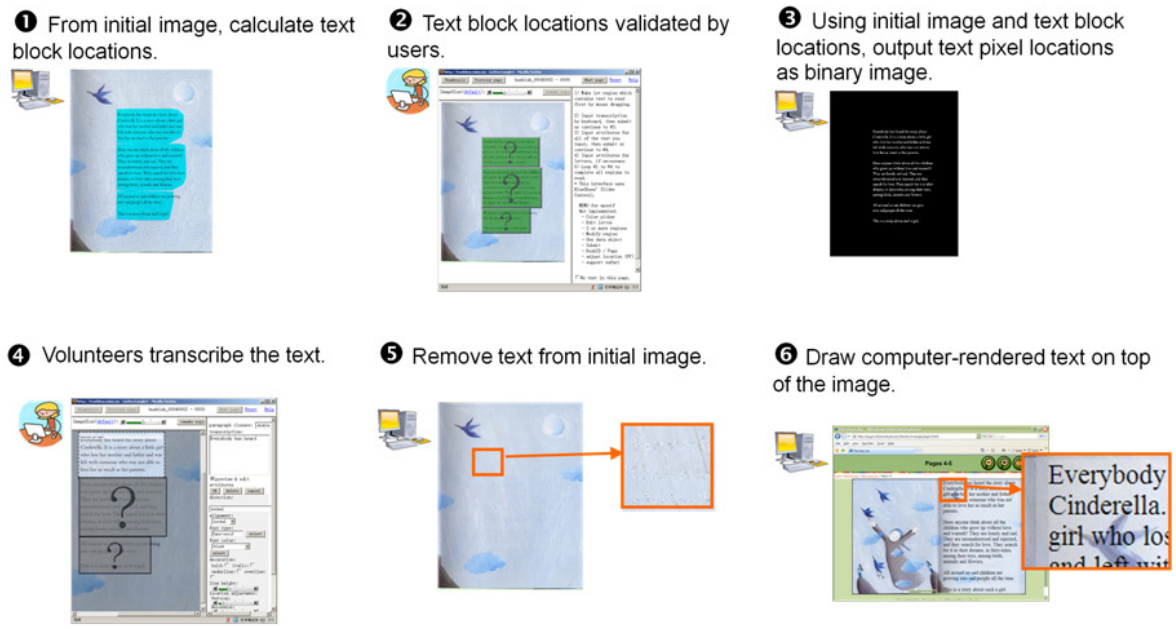

Figure 4. Major steps of processing

In practice, OCR is widely used with mass digitization for large online digital libraries [2], where little human intervention is involved. However, those mass digitization projects either require human intervention afterwards [7] or do not correct OCR errors [8]. None of these projects try to preserve layout and formatting information.

I also looked at what commercial and open source OCR software claim to offer. With maximum accuracy under 85%, most of the open source OCR software solutions are not accurate enough to use. The only exception, Tessearact (accuracy > 95%) [5], is still inferior to its commercial counterparts. Commercial OCR software usually claim a much higher accuracy, but I believe they are not applicable due to the unsolved research problems of language, layout and background. The ICDL also has books in uncommon languages such as Khmer and Rarotongan that none of the available commercial OCR systems support (Figure 3).

Motivated by the ESP game [6] and the Distributed Proofreaders program [7] in Project Gutenberg, I am trying to compensate for the insufficiency of an entirely automated approach by including some human effort from volunteers. Unlike the ESP game which relies solely on human users, I am taking a semi-automatic approach with multiple iterations. A similar effort to combine automatic processing and human validation is SAPHARI [20]. Our approach is also similar to the Distributed Proofreaders program in that a semi-automatic process is used to transcribe scanned book pages. In Project Gutenberg, scanned pages first undergo an OCR process to obtain relatively accurate transcriptions. Then, human volunteers in the Distributed Proofreaders program proofread OCR results through an online user interface. However, that solution would not be adequate for our problem because it does not collect any information except for the raw text. For our purposes, I also need to preserve the artistic representation through text layout and pictorial background. Our work is also closely related to the semi-automatic adaptive OCR of Rawat et al. [19], although I focus on mainly picture books with complex page background, and I have a very different interface design.

3. SYSTEM DESIGN

The goal for text-background separation is to remove all the text pixels from the page and fill in the missing pixels in a way that follows the texture and structure of the surrounding background. The text-background separation system consists of four parts: text location, transcription, inpainting and rendering. Text location combines automatic image processing algorithms with human validation to find the text blocks, and then locates the text pixels in a page (Steps 1-3, Figure 4). The system then provides an interface for volunteers to transcribe the pages with necessary layout formatting (Step 4). With text pixel locations, pictures in the background can be mended using inpainting [8] (Step 5). At the end, the system renders the transcribed text with a more readable and more efficient representation (Step 6). I now look at the main components of the algorithms involved here more closely.

3.1 Text Location

The first step is to determine which pixels in the image scan correspond to text. Text location is a binary-value function

$$f(x,y)=\begin{cases} true & if\ P(x,y)\ is\ text \\ false & otherwise \end{cases} \qquad (1)$$

where $P(x,y)$ is the pixel at position (x,y). In other words, text location takes every pixel in the image, and then classifies it as text or non-text.

A semi-automatic iterative text location algorithm that I designed is then used. For each page containing textual content, automatic image processing techniques are initially applied to the whole

```
Image LocateText(Image imgIn, out Array<Rectangle> textBlocks) {
  Image imgThed = ColorThredsholding(imgIn, PARAM_THRESHOLD);
    Image imgCleaned = CleanConnectedComponents(imgThed,
                       PARAM_MIN_WORD_SIZE,
                       PARAM_MAX_WORD_SIZE);
  Image imgTextBlocks = Dilate(imgCleaned, PARAM_BLOCK_DILATE_SIZE);
  Array<Rectangle> textBlocks = GetTextBlocks(imgTextBlocks);
  return imgThed;
}
```

Figure 5. Text location algorithm

page to perform text location. Text pixels then form text blocks which undergo human validation. If a block is manually corrected, the text location process is repeated on the block to get a more accurate result. It is much faster for humans to validate text blocks than working on the pixel-level. Automatic algorithms also work very accurately within one text block since most complex backgrounds are outside of the block.

Our algorithm uses the image processing techniques of color thresholding, connected component analysis and morphological transformation. This algorithm is based on two assumptions: 1) Text is darker than the background; and 2) within a page, the font is homogeneous. For most pages, these assumptions are true. For pages that don't meet these assumptions, the human validators will have to correct it.

Under the first assumption, color thresholding separates dark pixels from the rest of the background (Fig.10-2). Color thresholding is a binary-value function

$$f(x,y)=(R(x,y)<r_0)\wedge(G(x,y)<g_0)\wedge(B(x,y)<b_0) \qquad (2)$$

where $R(x,y), G(x,y)$ and $B(x,y)$ are the red, green and blue values of the pixel at position (x,y), and that r_0, g_0 and b_0 are predefined thresholds for the corresponding color values. The output of color thresholding is a binary image, where dark pixels in the original image are white and others are black.

A connected component analysis is then performed to remove connected components too small or too large compared to known fonts (Fig.10-3). A connected component is a collection of foreground pixels that are adjacent to each other. In our application, foreground pixel are white and background pixels are black. Two pixels are adjacent if and only if they share the same edge. Under this definition, a pixel can be adjacent to a maximum of four pixels (4-connection). A connected component is considered text only if it is not significantly wider and not taller than a word.

After the connected component analysis, most text pixels are preserved while most non-text pixels are removed. The regions of text pixels (i.e. letters) are then dilated to form text blocks

```
Image ColorThredsholding(Image imgIn, Double t) {
  foreach (Pixel pix in imgIn) {
    if ((pix.R < t) && (pix.G < t) && (pix.B < t)) {
      pix.Color = Color.White;
    } else {
      pix.Color = Color.Black;
    }
  }
  return imgIn;
}
```

Figure 6. Color thresholding algorithm

```
Image CleanConnectedComponents(Image imgIn, Size minSize,
Size maxSize) {
  Array<ConnectedComponent> components =
                                GetConnectedComponents(imgIn);
  foreach (ConnectedComponent c in components) {
    if ((c.Width > maxSize.Width) || (c.Height > maxSize.Height) ||
        (c.Width < minSize.Width) || (c.Height < minSize.Height))
{
      foreach (Pixel pix in c.Pixels) {
        pix.Color = Color.Black;
      }
    }
  }
  return imgIn;
}
```

Figure 7. Connected component analysis algorithm

```
Image Dilate(Image imgIn, Double dilateRadius) {
  Image imgOut = imgIn;
  foreach (Pixel pixOut in imgOut) {
    if (pixOut.Color == Color.White) continue;
    foreach (Pixel pixIn in imgIn) {
      if (pixIn.Color == Color.Black) continue;
      if ((Abs(pixOut.X - pixIn.X) < dilateRadius) ||
                    (Abs(pixOut.Y - pixIn.Y) < dilateRadius)) {
        pixOut.Color = Color.White;
      }
    }
  }
  return imgOut;
}
```

Figure 8. Dilation algorithm

(Fig.10-4). Dilation [22] is a mathematical morphological operator that enlarges boundaries of regions of foreground pixels. When the letters are dilated, the level of enlargement can be defined so that neighboring letters become connected, and then each block of text is represented by one connected component.

Color thresholding alone can roughly locate text pixels, but the two steps that follow are still necessary. Connected component analysis can remove most false-positive background pixels. Text blocks give users a quick way to identify errors, and provide a more restricted area for more accurate re-processing.

The dark text and homogeneous font assumptions are the basis of the above algorithm. Although there are some examples in our collection where the two assumptions do not hold, errors can be quickly corrected by users with the validation/transcription interface. Automatic re-processing will proceed henceforth with the corrected text blocks. Initial parameters for the algorithms, i.e. font and line spacing are predefined. Through our experiments, one set of global parameters gives correct results

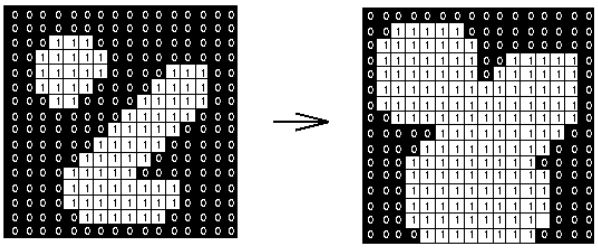

Figure 9. Dilation (from [23])

Figure 10. Steps of automatic image processing (left-to-right, top-to-bottom). (1) Input image; (2) Color thresholding; (3) Connected component analysis; (4) Dilation; (5) Text block formation; (6) Result.

with automatic text location on 58 out of the 75 pages processed.

3.2 Validation/Transcription Interface

A web-based interface enables humans to validate the automatically generated text blocks and to transcribe pages. I decided to use the same interface for validation and transcription since these two tasks are closely related, and could be completed in parallel. For most simple cases, human validation is redundant and can be done very quickly. The interface shows books that have not been validated or transcribed. Upon loading a book, the thumbnails of every page in the book are shown. Users can click on a thumbnail to perform verification in a page view (Figure 12). In the page view which is used to verify and transcribe the page (Figure 13), the automatically generated text blocks are shown on the interface as semi-transparent blocks on top of the scanned image of the page. When users click on a block, the interface zooms in to a detail view with the text block. A user input text field is overlaid on top of the original text. A panel with a textbox and font/layout controls appears to the right hand side of the page to allow with the human transcriber to specify font and layout information. The layout information includes direction, alignment, font, color, decoration, light height, and adjustment. Using this panel, users can tune those parameters until the rendered transcription is aligned with the underlying textual content of the page. This step isn't necessary, but if this information is provided, then I can render the text in a way that corresponds more closely to the original visual presentation. An incorrect text block can be removed, and new blocks can be created by dragging out an area with the mouse on the page. Users can save their work on each page and return to it later, or

finalize a page by submitting it to the database. The same interface can also be used for translating books. Instead of transcribing, users enter translated text into the textbox. Although overlaying the user's text onto the original is not necessary in this case, it is still helpful for font/layout alignment. Text block validation remains the same in the case of translation.

3.3 Inpainting

After a page is transcribed, the original text needs to be removed by spreading the background texture over text pixels. This process is called inpainting. Inpainting means "the modification of images in a way that is undetectable for an observer who does not know the original image is" [9]. Starting from the boundary of the region to fill, inpainting recursively replaces text pixels on the boundary with neighboring texture, shrinking the region and shortening the boundary on each step.

The inpainting algorithm in our system is a type of geometric inpainting, which is dependent only on the geometry of the original image. It first calculates level lines on which the intensity of pixels is similar, then fills each empty pixel based on its level-line neighbors by keeping the level line continuous. Continuation is preserved by solving the third-order partial differential equation

$$D^3 I \left(D^{\perp} I, D^{\perp} I, D^{\perp} I \right) = 0 \qquad (3)$$

where $D^3 I$ is the third-order gradient of the original I, and $D^{\perp} I$ is the 90º rotation of the gradient of I, or the direction of the level line.

As shown in Figure 11, the input to the inpainting algorithm is the original image and the location of text pixels. The output is an image that contains only the background and no text.

3.4 Text Rendering

As mentioned, the interface to ClearText is designed to improve readability [3], embodying the idea that the human reader has different needs for the resolution of the text and pictures. In ClearText, transcriptions obtained from the validation/transcription interface are used to re-render the text into a resizable text box on top of the inpainted background. The text box has a semi-transparent background to ensure that the text won't be obscured, even if it runs onto an area of the illustration with a busy background. The text box is located at the position of the original text blocks, while the text is rendered onto it with as close to the original layout, font and formatting as possible. ClearText makes it possible for the users to resize the text without resizing the background, preserving the context while enhancing the readability of the text.

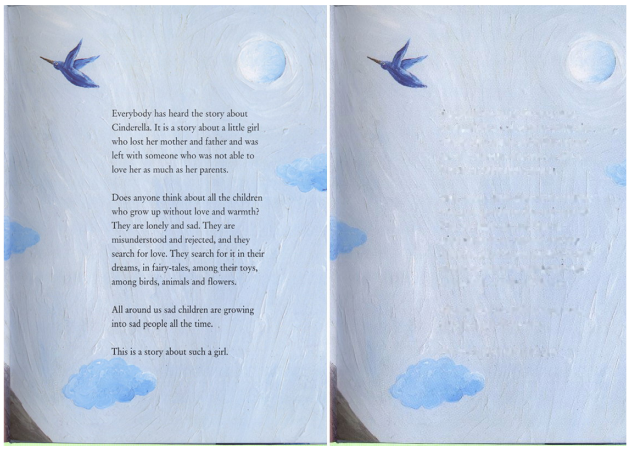

Figure 11. Effect of inpainting

I also designed another interface, PopoutText, for an evaluative comparison which has turned out to be interesting in its own right. PopoutText selectively magnifies the portion of the image that contains the text blocks when the mouse is clicked on those text blocks. The magnified portion is opaque so its background is occluded. Since the text is still displayed in its original form, no infilling is needed. PopoutText is interesting because it better preserves the artistic intent of the book author and illustrator – since it uses the original image. However, it is limited because it does not work well for pages that are full of text, and it has no support for the added features of translation, read-aloud or search. On the other hand, it is technically much simpler as it only needs the text bounds detection that our algorithms offer, and does not need transcription or inpainting.

4. IMPLEMENTATION

I implemented ClearText and PopoutText with the just-mentioned algorithms as follows.

The automatic text location algorithm is implemented using the existing Image Processing Toolbox in Matlab®. Since the text location algorithm is relatively simple, it is not a bottleneck and I would expect very little difference in efficiency between an implementation written in Matlab® versus one written in another programming language (i.e. C/C++).

The inpainting algorithm is implemented in C++ based on the

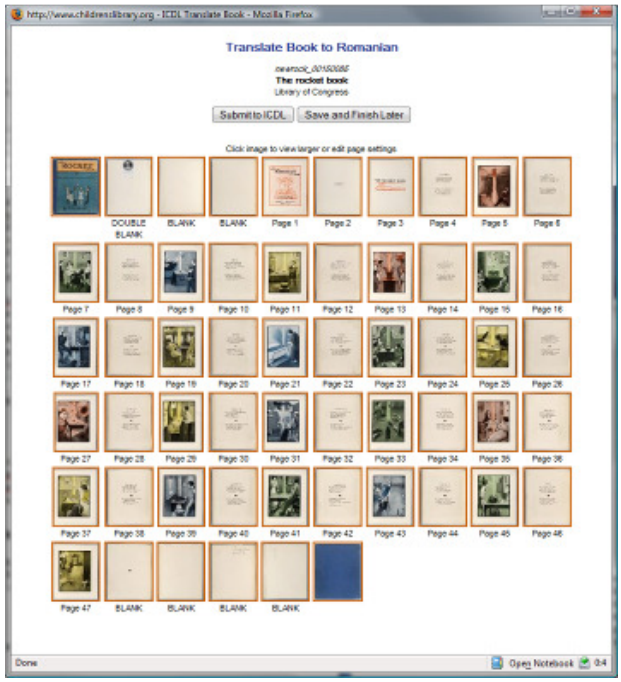

Figure 12. Thumbnail view of transcription/verification interface

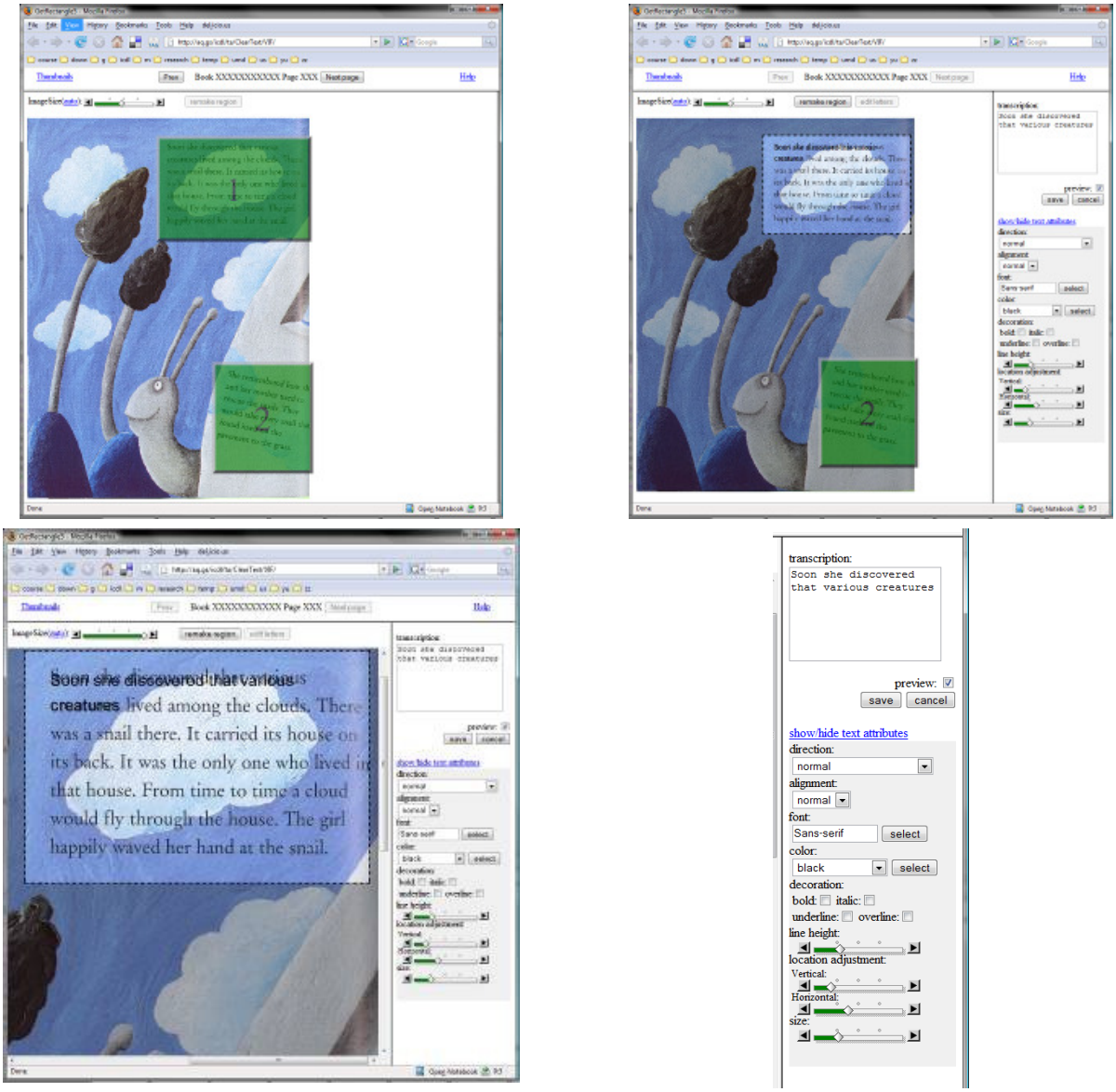

Figure 13. Transcription/Verification Interface.
(top-left) blocks overview; (top-right) blocks editing view;
(bottom-left) editing view close up; (bottom-right) font/layout panel close up.

code shared by Bertalmío and described by his team [9], The numerical solution of Eq.(X) has been implemented using an explicit, forward time, finite differences scheme with the monotonized central difference slope limiter of Van Leer .

The validation/transcription interface was implemented using a mixture of Java and JavaScript, consistent with the rest of ICDL interface. The book reader interface is implemented using standard DHTML with CSS to display the text-free image, with HTML-rendered text on top of that image. Those implementations have been tested on multiple browsers including Internet Explorer, Mozilla Firefox and Opera.

The reading interfaces have been deployed internally for user studies. A modified version of the reading interfaces has also been tested on the children's laptop provided by the One Laptop Per Child Foundation [21]. The ClearText system will be deployed to the main ICDL web site soon.

5. CONCLUSION

I described a system that enhances the readability of scanned books by decoupling the text and visual background of the page. Decoupling is obtained through a semi-automatic image processing approach using image processing for a first pass and humans to correct the results. This approach is crucial because at this point, generating the new interfaces with an entirely automated process is impossible because image processing techniques simply cannot do the task at sufficient quality.

The combined computer/human approach enabled us create two book reader interfaces, ClearText and PopoutText, which are designed for an enhanced reading experience. ClearText makes it possible for the users to resize the text without resizing the background, whereas PopoutText selectively magnifies the image portion that contains the text blocks.

With a combination of simple computational and manual processing, this system is able to yield an enhanced reading experience, which no purely automated system has achieved before. I will continue to try and improve the automated part of the system which will allow us to decrease the human effort required to correct the inevitable mistakes. However, while a semiautomatic system will become more efficient as more of the manual tasks are be replaced by reliable computational equivalents, for any semi-automatic system, there will always be a tradeoff between developers' efforts spent on automation, and users' efforts spent on manual tasks.

It would also be a very interesting future direction to design a text block validation/transcription interface to motivate massive online users, similar to the ESP Game. This approach could enhance the scalability of the system, so that it becomes a better fit for mass digitization.

6. ACKNOWLEDGMENTS

The author would like to thank Dr. Marcelo Bertalmío for sharing his implementation of his inpainting algorithms. I would also like to thank Alexander J. Quinn, Anne Rose and Takeshi Arisaka and my advisor Benjamin B. Bederson for their advice.

7. REFERENCES

[1] ICDL facts page, http://www.childrenslibrary.org/about/fastfacts.shtml as of Dec 2007.

[2] Coyle, K. Mass Digitization of Books. In *Journal of Academic Librarianship*, v32 n6 p641-645 Nov 2006

[3] Quinn, A., Hu, C., Bederson B., Rose, A. and Arisaka T., Reading Scanned Books in Web Browsers, in *CHI 2008* (in press)

[4] Boyarski, D., Neuwirth, C., Forzlizzi, J., and Regli, S. H. A Study of Fonts Designed For Screen Display. In *Proc. CHI 2002*, ACM Press (2002), 87-94.

[5] Google Tesseract, http://code.google.com/p/tesseract-ocr/

[6] Ahn, L. v. 2006. Games with a Purpose. *Computer* 39, 6 (Jun. 2006), 92-94.

[7] Distributed proofreaders, http://www.pgdp.net/c/

[8] Google Book Search, http://books.google.com/

[9] Bertalmin, M. Strong-Continuation, Contrast-Invariant Inpainting With a Third-Order Optimal PDE. In *Image Processing, IEEE Transactions on*, v15 n7 1057-1938.

[10] MODI, Microsoft Office Document Imaging, http://office.microsoft.com/en-us/help/HP030763951033.aspx

[11] OmniPage, http://www.nuance.com/omnipage/languages/

[12] Readiris, http://www.irislink.com/c2-479-189/features.aspx#rec

[13] ABBYY reader, http://www.abbyy.com/finereader8/?param=44919#nowhere

[14] Ocrad, http://www.gnu.org/software/ocrad/ocrad.html

[15] GOCR, http://jocr.sourceforge.net/

[16] Kanungo, T., Resnik, P., Mao, S., Kim, D., and Zheng, Q. 2005. The Bible and multilingual optical character recognition. *Commun. ACM* 48, 6 (Jun. 2005), 124-130.

[17] Mao, S., Rosenfeld, A., and Kanungo T. Document structure analysis algorithms: a literature survey, in *Proc. SPIE* 5010, 197-207.

[18] Jung, K., Kim, K., Jain, A K. Text information extraction in images and video: a survey, in *Pattern Recognition*, 2004 Volume 37, Issue 5, (May 2004), 977-997.

[19] Rawat, S., Kumar, K.S.S., Meshesha, M., Sikdar, I.D., Balasubramanian, A., and Jawahar, C. V., A Semi-automatic Adaptive OCR for Digital Libraries. *Lecture Notes in Computer Science* 3872/2006, 13-24.

[20] Suh, B. and Bederson, B. B. 2007. Semi-automatic photo annotation strategies using event based clustering and clothing based person recognition. *Interact. Comput.* 19, 4 (Jul. 2007), 524-544.

[21] OLPC Project, http://laptop.org/

[22] Gonzalez, R. and Woods, R. Digital Image Processing, Addison-Wesley Publishing Company, 1992, pp 518 - 519, 549.

[23] Morphology – Dilation, http://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.htm