# PDF TO HTML CONVERSION
# Progress Report

## Summary

As a result of an extensive investigation into the existing solutions to this problem, it has been decided to modify the aims of this project to remove the emphasis on preserving the original page layout.

The current solutions to the problem have been found to convert PDF files to HTML with fairly high levels of success, accurately preserving the page layout in most cases. There is little sense in simply repeating this work. Furthermore, many of the features and benefits of HTML are lost with these methods of conversion. It has therefore been decided to aim this project at extracting the content from a wide variety of PDF files, and presenting it in a "clean" HTML format, utilizing HTML's features for styles, formatting, bullet points, and any other features that are deemed appropriate. Such files benefit from being re-flowable according to the size of the browser window, easier to edit and look more professional when published to the Web.

The original aim of preserving page layout has not been completely disregarded, and will be investigated if there is time remaining at the end of the project.

## Investigation of existing solutions

The four following programs were found on the Internet to offer a solution to the problem:

- **PDF to HTML Recastor** by Archisoft *(www.archisoftint.com/logiciels/recr_us.htm)*
- **pdftohtml** sponsored by Lincoln & Co *(pdftohtml.sourceforge.net)*
- **pdf2html** by Twibright Labs *(atrey.karlin.mff.cuni.cz/~clock/twibright/pdf2html)*
- Google's **View as HTML** feature for cached PDF files *(www.google.com)*

The first solution, **PDF to HTML Recastor**, was the only commercial solution on the Internet that had a free trial version available for download. This version was researched in detail with the following PDF files:

| Title | Type of layout | Location |
|---|---|---|
| Boston Sunday Globe, Today, October 20, 2002 | Complex newspaper; columns | *www.boston.com/globe/acrobat/today.pdf* |
| White Paper: Is the Network Slow Today? | Word-processed document | *www.netscout.com/files/artmb_wp.pdf* |
| Connex South Eastern Rail Timetable #5 | Tabular | *www.connex.co.uk/upload/timetable/PTT05.pdf* |

All of the files were downloaded on October 21, 2002. Availability, content and location of these files may have changed since this date.

**Results with the Archisoft converter**

This converter generated a series of HTML files from the PDF; one for each page of the document. Each line of text from the PDF was positioned separately in the HTML, with an absolute pixel coordinate relative to the page. All other elements of the page, including images, lines and boxes, were rasterized into a single image, which was shown as the background of the HTML page. The converter also had a feature to generate a page index in a separate frame displayed on the left-hand side of the screen.

The converter coped well with the relatively simple word-processed document. However, the other two documents revealed some limitations with it. The newspaper article was designed for a large (approximately A3-size) page, and could not be viewed on an average computer screen without scrolling. The converter attempted to resize the page to fit in an average browser window (about 800 pixels across). This resulted in text that was too small to read, even with the magnification facility on the converter set to the maximum level.

This problem was exaggerated by the fact that all the text displayed in the HTML file was smaller than in the PDF document when viewed at a similar level of magnification. This was understood to have been done to avoid columns of text running into each other when viewed with a different font or across different platforms.

There were two other main areas of possible improvement with the converter: first, while it attempted to maintain the appearance of the page, it did not include an option to rasterize headlines and other text above a certain size. As many pages often use fancy fonts for headings, this would have enabled the page to more closely resemble the original. It would also have solved the problem of the custom font for the headline in the newspaper article. Another problem with the converter was its crude resizing of images: algorithms which use bilinear or bicubic resampling produce superior results when displaying images not at their native resolution.

One of the most noticeable artefacts of conversion with the article was the headline, which had been created into a font in the PDF document. Each word of the headline was a different character in this custom font. As the converter displayed everything in the Arial font, the headline was displayed as “fghijkl” as shown in Figure 1 below. As this is a rather special case, one would not expect a converter such as this to cope with the headline.


LOTTERY PAGE A2

Volume 262
Number 112
$2.00

fghijkl

October 20, 2002

``My concern is that people who've made allegations in the past don't see this as any lessening of the bishops' resolve.''

The Rev. Kevin E. McKenna , past president of the Canon Law Society of America.

Focus shifts to accused priests' rights

Vatican's revisions put spotlight on `due process' issue

By Michael Paulson and Sacha Pfeiffer
GLOBE STAFF

The scene in June was remarkable: victims of clergy sex abuse at center stage, telling all the bishops of the United States, in public and on national television, about their pain and anger. Some cried. One asked the bishops to pass around a picture of a young man who had killed himself after being molested by a priest. And they did. A lone advocate for priests' rights was given a badge declaring him an ``observer'' at the Dallas bishops' conference, and he spent much of the week in a hotel corridor, but tonholing reporters who might lis ten to his argument that priests weren't getting a fair shake.

Flash forward four months. As US bishops prepare to meet again, next month in Washington, D.C., a new group has moved to center stage: accused priests. All summer, as hundreds of priests slowly lost their jobs over allega tions of sexual abuse, the clamor began to rise. What about due pro cess? What about a fair trial? A right to confront an accuser? A day in court?

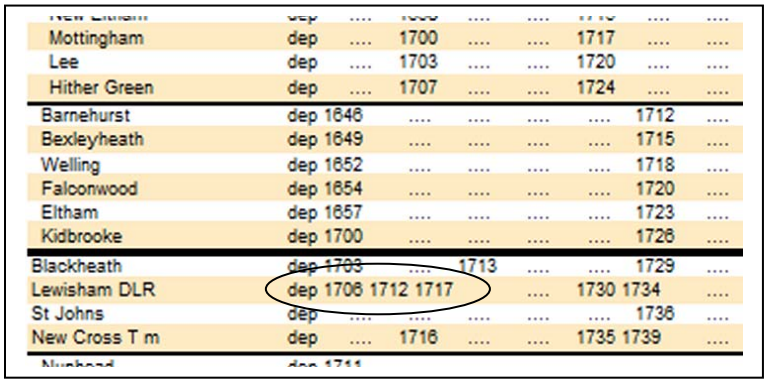
Priests began to organize to as sert their rights and to hire canon and civil lawyers to defend them. Some sued their accusers. Others appealed their removal from min istry to the Vatican. Bishops, in cluding Cardinal Bernard F. Law of Boston, scheduled gatherings to talk with angry clerics.

On Friday, the shift of the spot light was complete. The Vatican said it would not approve the child

VATICAN, Page A30

*Fig 1: a section of the Boston Daily Herald article, as converted by the Archisoft converter*

A different issue was highlighted by the conversion of the train timetable. In Figure 2, the figures should all appear under each other. However, the converter has mistakenly detected the circled side-by-side figures as words in a line of text. Rather than place them separately, it has placed them together as a line of text, with each figure separated by a space. The result is that the figures are not in the right place. The figure "1717" should actually appear underneath the figure "1713" in the line above. Again, this is another special case that highlights a useful feature of the converter designed to improve the appearance of poorly created documents where each character or word may be placed separately. Unfortunately, there was no way to turn off this feature, as this is a general purpose converter designed for a wide variety of documents.



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Mottingham | dep | .... | 1700 | .... | .... | 1717 | .... | .... |
| Lee | dep | .... | 1703 | .... | .... | 1720 | .... | .... |
| Hither Green | dep | .... | 1707 | .... | .... | 1724 | .... | .... |
| Barnehurst | dep | 1646 | .... | .... | .... | .... | 1712 | .... |
| Bexleyheath | dep | 1649 | .... | .... | .... | .... | 1715 | .... |
| Welling | dep | 1652 | .... | .... | .... | .... | 1718 | .... |
| Falconwood | dep | 1654 | .... | .... | .... | .... | 1720 | .... |
| Eltham | dep | 1657 | .... | .... | .... | .... | 1723 | .... |
| Kidbrooke | dep | 1700 | .... | .... | .... | .... | 1726 | .... |
| Blackheath | dep | 1703 | .... | 1713 | .... | .... | 1729 | .... |
| Lewisham DLR | dep | 1706 | 1712 | 1717 | .... | 1730 | 1734 | .... |
| St Johns | dep | .... | .... | .... | .... | .... | 1736 | .... |
| New Cross T m | dep | .... | 1716 | .... | .... | 1735 | 1739 | .... |

*Fig 2: a section of the rail timetable, as converted by the Archisoft converter*

To conclude, the Archisoft converter is a "one step" approach designed to be used by an inexperienced user to convert a wide range of documents to HTML. Its results could be improved by making available more options for the conversion. However, the product is not aimed at users who are likely to understand what these options mean.

**Results with other converters**

As the **Twibright Labs pdf2html** open-source converter simply used Ghostscript to rasterize each page into a PNG image it produced a perfect result. However, as mentioned in the specification, most of the benefits of converting to HTML are lost in this approach.

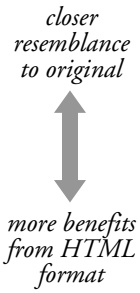
The **Lincoln & Co pdftohtml** open-source converter had two modes of conversion, simple and complex. In the complex mode it performed in a very similar way to the Archisoft converter, and encountered the same problems with the newsletter article. In the simple mode the images were placed at the top of the page, with all the text underneath. No attempt was made to reconstruct full lines or paragraphs of text from the narrow columns of the newsletter article. Horizontal lines in the page had also turned into long strings of repeating "1"s. However, such a file, given some "cleaning up", could be more readily turned into a Web-publishable document.

Finally, **Google's View as HTML** feature also performed similarly to Archisoft's converter, although, for bandwidth and computational reasons, it did not process images at all. The files generated were slightly larger, due to the way that the HTML was structured.

**Conclusions from the investigation**

The investigation highlighted three main approaches to the conversion process:

1. Convert each page to an image and display these images in one or more HTML files (Twibright Labs pdf2html)
2. Place each line of text relative to its position on the printed page, and display images and all other items as a background graphic (Archisoft pdf2html Recastor and Lincoln & Co pdftohtml)
3. Extract the text, but display it as normal HTML text (the standard option in Lincoln & Co pdftohtml begins to attempt this)




Each successive approach produces a HTML file which less resembles the original PDF file than the previous approach. However, the resulting file benefits from more of the features of the HTML format such as editability and reflowability.

As approaches 1 and 2 have already been successfully implemented in the programs studied, it has been decided to concentrate on approach 3. This will involve the following:

- reconstructing complete paragraphs from the lines of text in the PDF file
- dealing with hyphenation, bullet points, numbered lists, indentations and line spaces
- detecting headings and sub-headings and converting them to appropriate HTML styles
- for more complex layouts, attempting to detect columns and boxed sections in a page
- parsing the page elements in the correct order (ie for a column layout, starting at the top of the first column, moving to the second column, etc)
- in multi-page documents, dealing with headers, footers, footnotes and hyphenation, and joining two pages together to create a seamless flow of text

All these tasks are required simply to convert the text from PDF files into a "clean" HTML format suitable for on-screen viewing. A further possibility, if there is time remaining, is to attempt to recreate the page layout of the PDF file by using HTML tables. The advantage of tables is that they can alter their shape according to the size of the browser window. However, because of their inherent limitations, it is unlikely that they will work for all page layouts.

Another possible extension of the project is to implement and improve Approach #2 by offering more options to control the size of the outputted page, and to rasterize text larger than a certain size.

## Programming language selection

The Java programming language was selected for the following reasons:

- The author is already familiar with Java, having been taught it at university in previous years.
- It is inherently a multi-platform language, making it as portable as the PDF and HTML formats it will be used to convert.
- Third party libraries available, such as **JPedal*** for PDF processing and **Swing*** for GUIs.
- Although not known for its speed, it performs adequately on a relatively modern PC.

* *These libraries are described in the following section overleaf; web links are given in the* ***References*** *section.*

## Use of third-party libraries

The Java PDF Extraction Decoding & Access Library, **JPedal,** will be used to parse the PDF, and extract text and images from it. The **Swing** library, with which the author already has experience, will be used to provide the features for creation of the GUI. Use of other libraries to perform image manipulation functions is expected, although these will be researched at a later stage of the project.

---

## Design

As this project is focused on creating a single routine, it is not possible to produce any design work at this stage. Objectives 7 and 8, as listed in the following section, ***Revised objectives and timetable***, are more a task of research than of software engineering, and will involve much trial and improvement in order to discover which techniques work. Therefore, any designs created at this stage would be subject to considerable change.

A thorough design will be created, of the main program and GUI, after Objectives 7 and 8 are complete.

A basic overview of how Objective 7 will be tackled is shown in the pseudocode below.

- open file with library
- decode text as list of XML elements
- separate XML data into text coordinates
- copy list into array
- sort array in order of *x* coordinate
- sort array in order of *y* coordinate
- create a new vector of tuples with type (real * string) to store the *x* coordinate and its corresponding text
- for each item in the array
  - if object's *x* coordinate, or a similar value, already exists in the vector
    - add the *x* coordinate as the real to the vector, and the text as the string
    - else concatenate the text to the string that corresponds to the existing *x* coordinate*

* *at this stage, it will be necessary to check for indents, hyphenation, etc.*

The above pseudocode was written to parse the text of a single- or multi-column page and identify and extract the text from each column, which will be held in the vector, together with its horizontal position. Further processing will be required to identify headings and subheadings, different articles, headers and footers, etc, and may also entail examining other features of the page such as lines, boxes and images. These will be worked on at a later stage.

---

## Revised objectives and timetable

As the aims of this project have changed, and as a third-party library is being used to process the PDF file, it has been decided that the objectives in Phase Three of the initial specification are beyond the scope of this project as their implementation would require substantial modification to the library itself. Instead, it has been decided to investigate HTML tables and to implement the conversion via a different approach if there is time remaining at the end of the project.

The revised objectives are therefore as follows:

| | | |
|---|---|---|
| **PHASE ONE** | | |
| **1** | Get familiar with HTML syntax | Completed |
| **2** | Investigate HTML's facilities for controlling layout and position of text and images | Completed |
| **3** | Investigate graphics formats for web use (GIF, JPEG and PNG) | Completed |
| **4** | Research existing solutions to problem | Completed |
| **5** | Select a suitable programming language | Completed |

| | | |
|---|---|---|
| **PHASE TWO** | | |
| **6** | Investigate functions and abilities of the JPedal library | Completed |
| 7 | Find methods to: | |
| | • extract text and place it on a HTML page | Completed |
| | • combine lines of text into paragraphs | Vacation |
| | • detect new paragraphs by indentations or line spaces | Vacation |
| | • detect bullets and numbered lists | Weeks 11-12 |
| | • detect headings and subheadings | Weeks 11-12 |
| | • detect columns and boxed sections | Weeks 12-13 |
| | • detect features of multiple page documents, such as headers and footers, and combine them into a single stream of text | Weeks 12-13 |
| **8** | Find a method to extract graphics from the PDF, resize them if necessary and place them into an appropriate part of the HTML document | Vacation |
| **9** | Design GUI and main program code (which will be based on above methods) | Weeks 13-14 |
| **10** | Develop the software | Weeks 14-16 |

| | | |
|---|---|---|
| **PHASE THREE** | | |
| **11** | Investigate the possibilities of recreating page layout with HTML tables | Remaining time |
| **12** | Implement Approach #2, but with the improvements stated earlier in this document | |

# References


- Project web site
  *http://tamirhassan.com/project/*
- Initial specification (as included with this document)
  *http://tamirhassan.com/project/spec.pdf*
- Description of PDF format
  *http://www.adobe.com/products/acrobat/adobepdf.html*
- Adobe PDF 1.3 Reference Manual
  *http://partners.adobe.com/asn/developer/acrosdk/docs/PDFRef.pdf*
- A Beginner's Guide to HTML
  *http://archive.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html*
- The Java PDF Extraction Decoder Access Library (JPedal)
  *http://jpedal.org/*
- Creating a GUI with Swing
  *http://java.sun.com/docs/books/tutorial/uiswing/*


Web page addresses of the converters investigated in this project are given in the section entitled ***Investigation of existing solutions***, on the first page.