Automated trimming of scanned images

Caversham Project Occasional Technical Paper **Code Number:**020426CTP
Date: 26th April 2002
Author: David Hood
E-mail: caversham@otago.ac.nz
Keywords: scanning, image manipulation, automation, batch, trim

The Technical papers series is intended to document specific technical solutions to problems encountered within the project. It is being made available to the public in the event the techniques outlined may be of use to others.

Synopsis: Scanning in historical material from second hand reproductions can often result in extra material being captured in the image. This can confuse the OCR (optical character recognition) process. The techniques outlined within this document use graphic manipulation software, textual manipulation software, database software, and scripting control to automatically trim images to the desired area for OCR. While the implementation of this solution is Macintosh based, the techniques could be applied to any computer system.

Description of Problem:

Much of large scale scanning done by the project involves the scanning, and OCR of public historical sources such as trades directories, telephone directories, and electoral rolls. These are normally scans of photocopied reproductions of the original source, as this is the only way we are able to get access to the material. In many cases the material contains extraneous information such as sections of the facing page (see figure 1).

Figure 1: Example page including section of facing page, compared to desired page.

As well as including extra information, the location of the original document on the photocopied page may vary depending on where the original was in relation to the copier.

The above problems have meant scanning is a very manual, labour intensive, process. The techniques described below offer automatic solutions to these problems, requiring less manual labour.

The overall solution is to use techniques to locate the original page on the scanned photocopy and to trim the excess, unwanted, image prior to optical character recognition. For the specific implementation below to succeed the following is assumed:

- The photocopies of the original pages were made at the same size. As such, the desired original page will occupy the same area on every photocopy, though it may be in a different location.
- The photocopies were scanned taking the same area of the photocopied page at a constant resolution. This ensures that the original images occupy the same area in the scans.
- The images are scanned in black and white. This makes automatic identification of the printed area unambiguous. If non-text images were being analysed, this would not be a requirement.
- Metrics of the text width can be obtained for one image. As the originals are all at the same size and the same scanned resolution, the setting for one source apply to all others.

Implementation of the solution:

The material used in this example was scanned at 300 dpi black and white tiffs (with a threshold of 128). The scanned in area took in slightly less than an A4 page, being 2488*3450 pixels.

Several measures then need to be taken of one of the pages, these provide settings for preforming the page location calculations

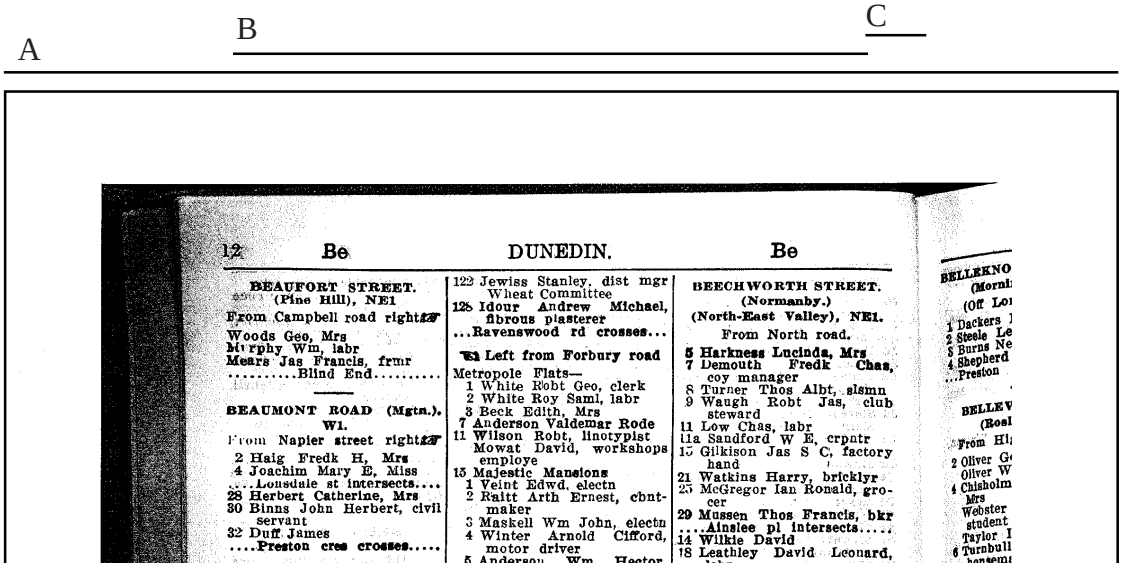

Figure 2: Important metrics

A) The total width of the page. Page Width.
B) The Maximum width of the text on the original page. Text Width
C) The area in which a cut is desirable. Trim Width.

As well as the three main metrics some assessment needs to be made of page slope – how much allowance needs to be made for different angles of photocopying the original page. Any trim lines should be broad enough to avoid trimming sloping text.

The material was then sorted into left and right facing pages, if the images are saved using file names that are consecutive in the directory structure this is a trivial job on the Macintosh using applescript.

Note: The example code provided in the boxed areas was used on a dual processor Macintosh G4 450 Mhz with 384 M Ram running System 10.1.4 and Applescript 1.8.2 (b3). It works fine on the machine it was written on, it may need adaptation for other systems. The code has been provided as batch jobs for each step in the process, in order to make it easier to understand what is going on. The provided code has also been written in a slightly less efficient more readable fashion.

The aim is to take the contents of the folder and place the files alternately into odd and even paged folders. This is to sort out what side of the page to trim. For left pages trim the right side. For right pages trim the left. For the purposes of this document the explanation will be with reference to left facing pages, a similar process applies to the right pages.

Sample code for sorting files:

```
set OddsTurn to true
-- assuming you start on page 1

tell application "Finder"
	set theFolderIWant to choose folder
	set ListofDocuments to every document file of theFolderIWant
	set Oddfolder to make new folder at desktop with properties {name:"OddFolder"}
	set EvenFolder to make new folder at desktop with properties {name:"EvenFolder"}
end tell

repeat with loopCount from 1 to number of items in ListofDocuments

	if OddsTurn then
		tell application "Finder"
			set nextfile to item loopCount in ListofDocuments
			move nextfile to Oddfolder
		end tell
		set OddsTurn to false
	else
		tell application "Finder"
			set nextfile to item loopCount in ListofDocuments
			move nextfile to EvenFolder
		end tell
		set OddsTurn to true
	end if

end repeat
```

Having sorted the pictures, the next step is to save the images as ascii text, recording the colour of each pixel as a text code. This resulting text file will then be sent onto a database for analysis. Before doing this, however, consider if some trimming is

possible before the analysis. For images on either face of the page the requirements for determining the region to cut are locating the sides of the original image. If the image occupies a significant proportion of the page, it may be possible to only sample a section across it.

The image from figure 1 occupies 2488 by 3450 pixels, which will require analysing 8583600 pixels of information. As the image occupies most of the page, it is always present in the middle of the page. By sampling the middle 20 per cent of the page, the computational time is dramatically reduced. The middle 20 per cent of the image in figure 1 is an area of 2488 by 690 pixels, containing 1716720 individual pixels of information. In the case of the machine this was trialed on, sampling reduced the time from approximately 14 mintues an image to less than a minute an image. I would recommend that any trim of the height of the image leave more than 400 pixels of information, to make the numbers involved in calculation more significant. If trimming to reduce the calculation overhead, pay particular attention to the margin for sloping images, as any slop in the trimmed area will not be detected in the calculations.

The program Graphic Converter (version 4.2.1) was used to accomplish the crop and save as ascii text. Before the batch job was run an image was saved as ascii text and under options in the save dialog the ‘write height and width to file’ was unticked. This meant that only the pixel information was recorded in the file, if other information is recorded, then the row and column location calculations in the database (see later) need to be adjusted. Because Graphic Converter created two files for each image, applescript was then used to separate the larger .txt file from the smaller .pal colour information file.

Sample code for cropping to middle 20% and saving as text:

```
tell application "Finder"
	activate
	set pathName to choose folder
	set picList to every file of folder pathName whose file type is "TIFF"
	set loopEnd to count of items of picList
	set cutFolder to make new folder at desktop with properties {name:"CutFolder"}
end tell

tell application "GraphicConverter"
	activate
	repeat with n from 1 to loopEnd
		set currentFile to item n of picList
		open currentFile as alias
		set imageInQuestion to image dimension of window 1
		set originalwidth to item 1 of imageInQuestion
		set originalheight to item 2 of imageInQuestion
		change margins of window 1 with {0, -(2 * originalheight / 5), 0, -(2 * originalheight
/ 5)}
		-- properties are Left Top Right Bottom
		save window 1 in (((cutFolder as text) & name of window 1) & ".txt") as ASCII
		close window 1
	end repeat
end tell

tell application "Finder"
	set textFolder to make new folder at desktop with properties {name:"TextFolder"}
```

```
        set bigFiles to every document file of cutFolder whose size > 100
        repeat with loopCount from 1 to number of items in bigFiles
                set largeFile to item loopCount in bigFiles
                move largeFile to textFolder
        end repeat
        move cutFolder to trash
        empty trash
end tell
```

Before analysing the records in a database, it is necessary to change the tab marks between the pixel information in each line to a carriage return symbol. This will enable each pixel to be saved as a separate record in the database. The carriage return (\r) is the Macintosh end of line symbol. Other computer systems will require different adjustments.

While there are many ways of changes tabs to end of lines, in this case the Macintosh OS X command line was the easiest means. Because the images have been transformed into very large text files, the unix tr tool offers an easy route to near instantly replace tabs. If the full image in figure 1 was saved as text, it would need 8580150 tabs replaced, so an efficient tool is required.

Sample code for command line tab replacement

```
tell application "Finder"
        activate
        set NotTabbed to make new folder at desktop with properties {name:"NoTabs"}
        set PosNoTab to POSIX path of (NotTabbed as alias)
        set Startfolder to choose folder
        set ListofTabbed to every document file of Startfolder
end tell

repeat with loopCount from 1 to number of items in ListofTabbed
        set tabbedFile to item loopCount in ListofTabbed
        set sameName to name of tabbedFile
        set butcherME to POSIX path of (tabbedFile as alias)
        set command to "tr '\\11' '\\015' <" & butcherME & " > " & PosNoTab & sameName
        tell application "Terminal"
                do shell script command
        end tell
end repeat
```

With the tabs replaced, the text file can now be analysed using a database program. The first stage of analysis is to determine the threshold values. The areas of the image that actually contain text can be distinguished from other areas by the amount of black in each column of the image. To have the computer recognise the text areas from the solid black band at the edge of Figure 2, the white areas of the photocopy edges, and the occasional specks of dirt on the white areas, the representative range of black in the image needs to be established. This was done by using a database program (Valentina 1.9.1 b2) to calculate the amount of black in each column and send that information to excel for graphing.

Using the graph of a sample image (Figure 3) the range of values for text can be easily assessed.

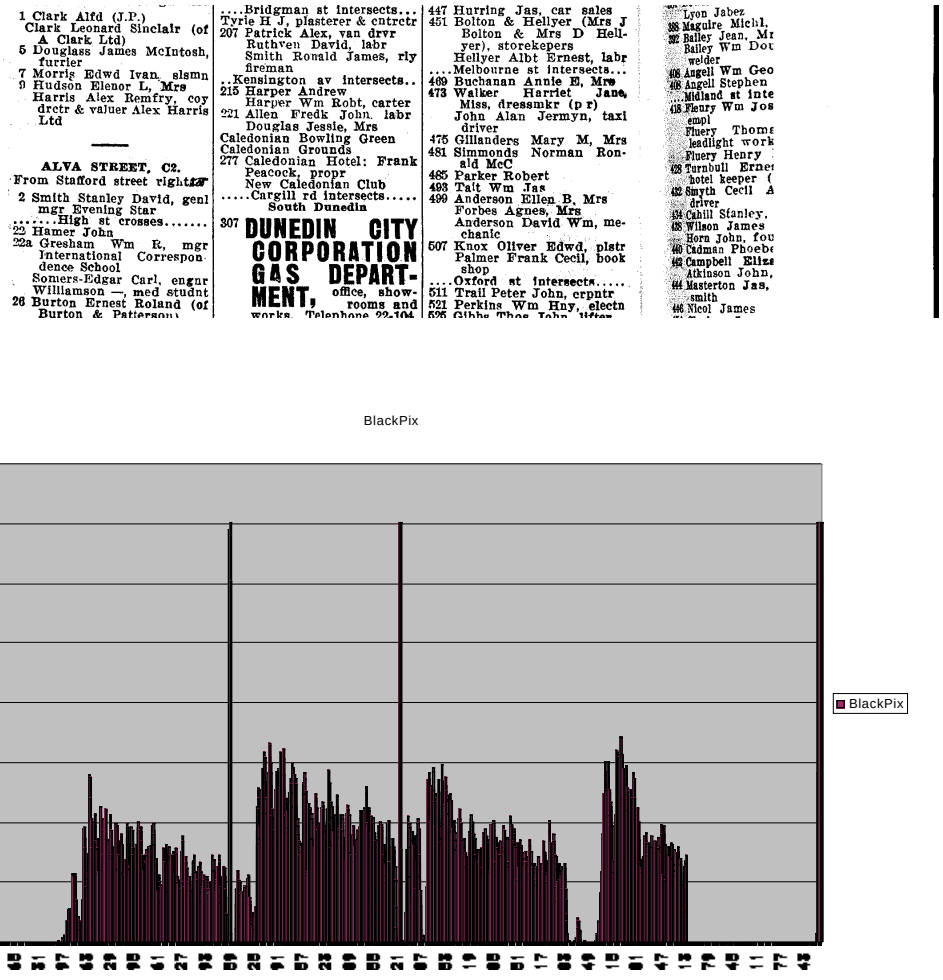

Figure 3: Sample Image and its accompanying graph of black pixels.

In The above example, the threshold for identifying text can be set from 3 to 450 pixels of black in a column. In order to assure that it is text that has been located, rather than an isolated scratch, part of the searching algorithm is to scan along several columns, confirming that there is a text range value of black in each, before declaring that the text area has been located.

In the following code the imageWidth variable at the start is set to the total width of the image (Page width from Figure 2). The field columnNumber created in the database is a calculation field that determines the column position of the pixel from the number of the database record, if a vertical analysis of the database was required the row number could also be calculated.

Sample code for analysing image

```
set imageWidth to 2488

--
--
tell application "Finder"
	set putItThere to (make new folder at desktop with properties {name:"DBtempFolder"}) as
text
end tell
tell application "Valentina Carbon beta"
	activate
	set theDB to make new database with data file (putItThere & "picture")
	tell theDB
		set theBaseObject to make new base object with properties {name:"image"} at end
		tell theBaseObject
			make new field with properties {name:"pixel", type:tBool} at end
			make new field with properties {name:"columnNumber", type:tUShort,
method:"RecID- ((CEILING(RecID/" & imageWidth & ")-1)*" & imageWidth, indexed:true} at end
		end tell
		set theFile to choose file with prompt "Text File"
		set theCursor to SQL theDB Select "SELECT pixel FROM image"
		import theCursor from ascii file theFile
		delete theCursor
	end tell
end tell
--
--
tell application "Microsoft Excel"
	set FormulaR1C1 of Range "R1C1" to "Column"
	set FormulaR1C1 of Range "R1C2" to "BlackPix"
end tell

repeat with loopCount from 1 to imageWidth
	tell application "Valentina Carbon beta"
		set tempCursor to SQL theDB Select ("SELECT pixel From image WHERE pixel =
1 and columnNumber = " & loopCount)
		get loopCount
		set Recs to get count of records of tempCursor
		delete tempCursor
	end tell

	tell application "Microsoft Excel"
		set FormulaR1C1 of Range ("R" & (loopCount + 1) & "C1") to loopCount
		set FormulaR1C1 of Range ("R" & (loopCount + 1) & "C2") to Recs
	end tell
end repeat
tell application "Microsoft Excel"
	Activate
	Select CurrentRegion of Range "R1C1"
	Create New Chart
	Delete Series 1 of ActiveChart
end tell
--
--
tell application "Valentina Carbon beta"
	quit
end tell
tell application "Finder"
```

```
        set disposeMe to putItThere as alias
        move disposeMe to trash
        empty trash
end tell
```

Now that the threshold values are known the folder of text files of picture information can be analysed, at the results applied to automatically crop the images in the left or right sorted folder. The script below sorts left facing pages, for right facing pages the direction of search should be reversed.

While this technique was developed to solve a particular problem, the methodology may be useful for other occasions, such as the analysis of images, or the comparison of timed pictures.

Sample code for batch detection and trimming

```
--Settings

set lowendThreshold to 10 -- high enough to screen out specks
set highendThreshold to 600 -- a little above the maximum level of text
set safetyMargin to 18 --allowance for slope
set totalImageWidth to 2488 -- total width of the image
set textArea to 1390 -- width of the desired text area
set cutArea to 80 -- area in which cut can be made

-----------------------------------------------------------------
-- Set Up
tell application "Finder"
        activate
        set outputFolder to (make new folder at desktop with properties {name:"FinalOutput"}) as
text
        set textFolder to choose folder with prompt "Select folder with text versions"
        set imageFolder to choose folder with prompt "Select folder with image versions"
        set textList to every file of folder textFolder
        set imageList to every file of folder imageFolder
        set loopEnd to count of items of textList
end tell


-------------------------------------------------------------------------
repeat with i from 1 to loopEnd

        tell application "Finder"
                set tempDBFolder to (make new folder at desktop with properties
{name:"TempDB"}) as text
        end tell

        tell application "Valentina Carbon beta"
                activate
                set theDB to make new database with data file (tempDBFolder & "picture")
                tell theDB
                        set theBaseObject to make new base object with properties {name:"image"}
at end
                        tell theBaseObject
                                make new field with properties {name:"pixel", type:tBool} at end
```

```
                                        make new field with properties {name:"columnNumber",
type:tUShort, method:"RecID- ((CEILING(RecID/" & totalImageWidth & ")-1)*" & totalImageWidth,
indexed:true} at end
                                end tell
                                set theFile to item i of textList as alias
                                set theCursor to SQL theDB Select "SELECT pixel FROM image"
                                import theCursor from ascii file theFile
                                delete theCursor
                        end tell
                end tell
                ------------------------------------------------------------------------
                -- Analysis
                set cutFromColumn to 0
                set cutToColumn to 0
                set verfiedTextareas to 0
                set searchCutColumn to 0
                set trimthreshold to 12
                set checkThisColumn to 1

                repeat until verfiedTextareas = 8

                        tell application "Valentina Carbon beta"
                                set tempCursor to SQL theDB Select ("SELECT pixel From image
WHERE pixel = 1 and columnnumber = " & checkThisColumn)
                                set Recs to get count of records of tempCursor
                                delete tempCursor
                        end tell
                        set checkThisColumn to checkThisColumn + 1
                        if ((Recs > lowendThreshold) and (Recs < highendThreshold)) then
                                set cutFromColumn to checkThisColumn
                                set verfiedTextareas to verfiedTextareas + 1
                        else
                                if (verfiedTextareas > 0) then
                                        set verfiedTextareas to verfiedTextareas - 1
                                end if
                        end if
                end repeat
                set cutFromColumn to checkThisColumn - (safetyMargin + 9)
                ---------------------------------------------

                set checkThisColumn to cutFromColumn + textArea
                repeat until cutToColumn > 0
                        repeat until checkThisColumn = (cutFromColumn + textArea + cutArea)
                                tell application "Valentina Carbon beta"
                                        set tempCursor to SQL theDB Select ("SELECT pixel From image
WHERE pixel = 1 and columnnumber = " & checkThisColumn)
                                        set Recs to get count of records of tempCursor
                                        delete tempCursor
                                end tell
                                set checkThisColumn to checkThisColumn + 1
                                if Recs < trimthreshold then
                                        set cutToColumn to checkThisColumn
                                        set checkThisColumn to cutFromColumn + textArea + cutArea
                                end if
                        end repeat
                        set trimthreshold to trimthreshold + 12
                        if trimthreshold > highendThreshold then
                                set cutToColumn to totalImageWidth - safetyMargin
                        end if
                        set checkThisColumn to cutFromColumn + textArea
```

```
		end repeat
		set cutToColumn to cutToColumn + safetyMargin
		----------------------------------------
		tell application "Valentina Carbon beta"
			quit
		end tell
		tell application "GraphicConverter"
			set currentFile to item i of imageList
			open currentFile as alias
			set imageInQuestion to image dimension of window 1
			set originalwidth to item 1 of imageInQuestion
			set originalheight to item 2 of imageInQuestion
			change margins of window 1 with {-cutFromColumn, 0, -(totalImageWidth -
cutToColumn), 0}
			save window 1 in ((outputFolder as text) & name of window 1)
			close window 1
			quit
		end tell
		----------------------------------------------------------
		----Clean Up

		tell application "Finder"
			set disposeMe to tempDBFolder as alias
			move disposeMe to trash
			empty trash
		end tell

end repeat
```

Relevant addresses:
Graphic Converter: http://www.lemkesoft.de
Valentina: http://www.paradigmasoft.com