Unleash the benefit of Optical Character Recognition

Reading Rights

The ability to lift printed text off the page and turn it into editable text is a wonderful skill, a skill that no longer remains in the sole domain of the typing pool, any two fingered typist can do it these days. Nor does it necessarily have to remain a process confined to big businesses with endless finances

BY COLIN MURPHY

There are many reasons why you might want to gather up the text from a printed page and have it stored in some character form. Businesses might get deluged with paper correspondence that they are obliged to keep, maybe even to 10 years for some documentation like invoices and receipt notes. The shear volume of such paper opens up the need for electronic document management.

It does not have to be limited to a large business. A SOHO user might generate the same amount of correspondence pro rata, but run out of room under the bed in which to store it. Document archiving would be of benefit here too. Though there is a different order of scale and throughput required.

Saving space isn't the only issue, you need to consider the management of the data too. You, as a SOHO user might want to archive lot of paper correspondence, some of the older stuff originally typewritten. If you have a scanner you could save each page as an image file. Such a typewritten page might taken 400KBytes to be stored, you might even build a database up giving brief details about each document so that you have some way of finding these documents again once you get past your 3rd CDs worth.

You could soon be starting to archive quite a lot of data, a lot of it would be worthless. A typed sheet scanned might come in at around 400KBytes for the image, which is a lot of data for the small amount of information that is actually held in the letter, about 4,000 characters maximum for an A4 piece of paper, probably half that once you add margins and formatting, so, 2KBytes of information is being held on a page.

Even if size were no limitation to you, you would still be missing a trick if it was saved in a graphical format. If you had access to the text itself on the pages you would be able to search through all the documents just like you could *cat* a readme file. You could take it further, build a database using the information contained in the correspondence.

What you need to do it

Somehow you need to get your text into a graphical format, it is from this stage that the OCR software can start working. How you do this depends very much on the form of your data. Most of the time your original input is going to be paper bound, so you will need some kind of optical scanner.

Alternatively, you may find your data is on your computer already with the text locked away, say, in the form of a .pdf file. Here you could print the PDF out an then scan it, better still, why not just grab the image with something like *xgrab*, which will allow you to capture your screen, or selected parts of it, saving it as a graphics file, like *.ppm*. This will work just in the same way, after all, your optical scanner is just going to make a similar type of file so there is little difference.

You will need some Optical Character Recognition. OCR software really is the clever bit of this process. It is pattern recognition software, but with a tightly defined set of patterns that it will recognize. This pattern is usually limited to the shape of the fonts. Most Linux distributions come with a package called GOCR. If you find you do not have this package, you can download a copy from [1]. Luckily enough GOCR is not that limited with the range of fonts that it will recognize as you can see in Figure 1. It is not an unlimited set though and it is unlikely to cope with any fancy fonts.

Much depends on the quality of the image that you are working with. If you are dealing with old paper that might have yellowed or has coffee mug rings on it you will find that parts of the image might not be recognized very successfully. A lot has to do with resolution as well, scanning your image down to the size of a postage stamp is going to be just as unreadable for the computer as it will be for an human.

Alternatively, scanning at maximum resolution is just going to waste time and create huge files that will take even more time to process. There are no hard and fast guidelines, but scanning somewhere around 300 – 600 dpi should see you right. Should your OCR passes be constantly failing then adjusting the resolution would be a good first step on the way to a solution.

Command line OCR

You can use GOCR in a variety of ways. From the command line you will get access to batch processing of image files, very convenient because the OCR process is likely to be the most time consuming operation. Passing parameters to GOCR on the command line is less than intuitive, so we can be thankful for the existence of the GOCR.tcl program, see

Figure 2, that will allow us to configure and run the OCR process from a GUI. It is unlikely that you will find GOCR on any of your desktop menus, so you will need to call it up from the the command line. There are other options, more of which later.

In Figure 2 you can see the GOCR.tcl front-end program along with a copy of Adobe Acrobat and *xgrab*. I used xgrab to 'screen capture' the XaoS story that you can see in the .pdf image. The subsequent image file */home/colin/txt3.ppm* was fed into the GOCR interface and the 'Run It' button was hit. About 10 seconds later up came the recognized text, which you can see in the main body of the GOCR front-end.

You should note the top four lines of text, these are left over from a previous attempt where the settings were not quite good enough, I had to adjust the 'greylevel' setting a little bit more to get an acceptable result. Also note how I had to enlarge the .pdf display, by 600%, to get a resolution that would work. Modern optical scanners have a higher resolution than that of a monitor, which was running at 1024x800.

To help you figure out how to drive GOCR from the command line, if you look down the bottom you will see the output that the front-end program passes to the OCR program. You would pass a similar command from a terminal. To get a full list of these commands you will need to look at *man gocr*.

OCR directly from a scanner

This might all seem a little complex at first, what you all really want is just point and click OCR, going through the GOCR.tcl front-end does make you aware of the technicalities though. Thankfully there are easier ways. Kooka is a scanning package which comes with the KDE desktop. You do not need to be using KDE just to run Kooka, you just need the right code available, like having the Qt and the base KDE libraries installed.

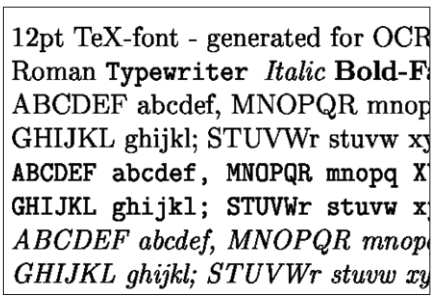

Figure 1: A selection of the range of fonts that GOCR will make an attempt to recognize

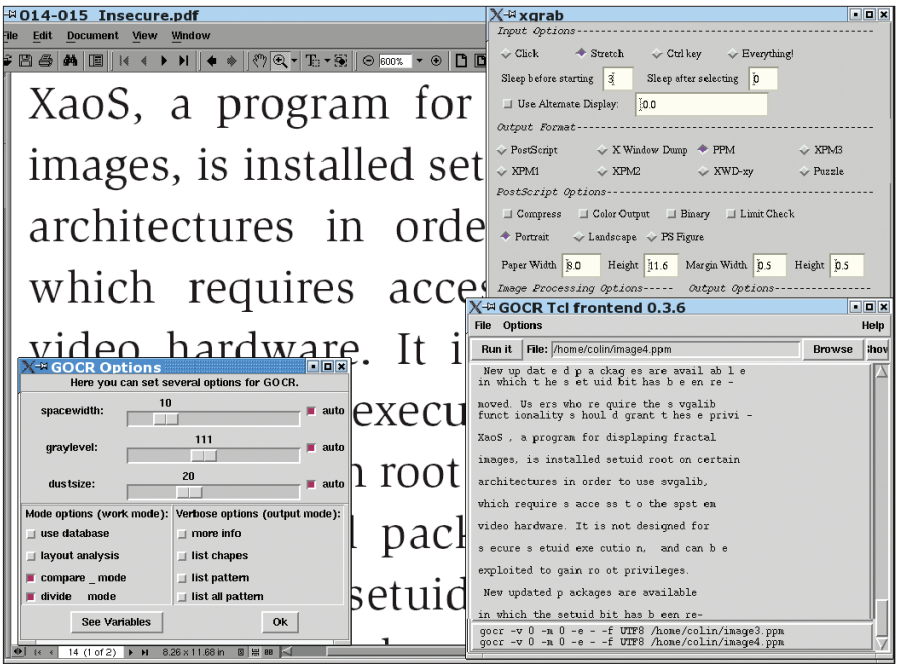

Figure 2: Using the GOCR front-end you can scan from a .pdf file just as easily as from a sheet of paper

Kooka is a front-end for the Sane scanner libraries, but as an extra facility, it allows you to carry out OCR functions from the one application. By default, this OCR process also uses GOCR.

Kooka offers you the chance to OCR the entire page or once you have taken a preview scan, limit yourself to a region. Any text picked up can be dropped into a text editor for further scrutiny and / or spell checking.

Proprietary alternatives

GOCR is not a new program, officially it is still in beta and has not seen an update for nearly a year. That is the way with some Open Source projects, they reach a certain level of maturity, they work as well as anyone reasonably expects and they go hardly any further.

If you are grabbed by the OCR bug in a big way, where you will want to invest real money into your document archiving plans, then there are a few proprietary solutions that you could try.

OCR Shop from Vividata [2] is one solution which, for a cost of nearly $1.500 for a minimum configuration, will give you a new range of features that GORC just can't touch. It is based on the ScanSoft OCR engine which has been used to great success on Windows and Mac systems. It will scan documents much faster, maybe page a second and cope with a much larger range of fonts in sizes from 5 – 72 dpi.

Remember, if you are looking to process hundreds of documents a day then these are the facilities you need, probably alongside an optical scanner with a sheet feeder, which will be another expense in the $1,000's.

KaDMOS [3] is another alternative, with a professional price tag to suit, if used in Windows platforms anyway. Should you be lucky enough to be a SuSE user, with an 8.2 box set, then you will find this in your installation already. This is fully integrated into the Kooka application and provides a very comprehensive character recognition solution.

Not only does this have the facility to recognize a huge range of fonts, it will also cope with handwritten scripts. The validity of any text output by the KaDMOS OCR is also much better scrutinized automatically. ■

INFO

[1] GOCR: *http://www-e.uni-magdeburg.de/jschulen/ocr/*

[2] Vividata: *http://www.vividata.com/ocrshop.html*

[3] KaDMOS: *http://www.rerecognition.com/*