Foxit

UNDERSTANDING OCR

An In-Depth Look at Optical Character Recognition

WHITE PAPER

INTRODUCTION TO OCR

Optical Character Recognition, also known as OCR, is the process of converting scanned paper documents into searchable, electronic files. In many office applications, such as invoice processing, there are clear time and cost efficiencies in converting paper documents into electronic documents. For example, entering document information into a database, also known as field coding, is often a very expensive proposition. Using electronically converted files, instead of the original paper, the process of field coding can be sent offshore or even automated.

The field of OCR even predates the advent of computers, with original OCRrelated patents dating back as far as 1929. Of course, much of the significant advances in OCR are directly related to the computer age, and usually, the more advanced the OCR system is, the more computer resources (e.g., faster CPU) are required to run the OCR system. OCR, and its related discipline ICR (Intelligent Character Recognition), are changing the way industry handles its documents. ICR is defined as the computer translation of manually entered text characters into machine-readable characters.

In many applications, including legal, accounting, banking, digital libraries, nsurance, remote backups, and records management, OCR is automating the way that businesses process files. Accurate OCR directly lends itself to data extraction which reduces the costs associated with form processing.

Advanced image processing, grayscale thresholding, and mixed raster content (MRC) techniques, to achieve compression rates ranging from 10 to 1 on simpler, text-heavy black-and-white images to more than 100 to 1 on graphics heavy color files, and OCR accuracy rates greater than 99% (when measured by characters).

Deployable within a user's capture workflow, PDF Compressor brings multiple benefits to end users, including potentially drastically reducing their total cost of storage—a savings that by itself will typically provide a cost justification for the software.

For a document imaging reseller, PDF Compressor represents a new product that can be added to a capture stack, without displacing any existing software. It is complementary to both capture and ECM—because it makes both applications work better. Additionally, PDF Compressor can be used to upgrade existing implementations, as well as differentiate new systems—and make them more compatible with this emerging world of portable documents.

DOCUMENT CAPTURE & OCR

Document Capture is the first step in the OCR process. This process is alternatively known as scanning. Common capture devices include scanners, digital copiers, MFPs, fax machines, and cell phones. Technically, the capture process is usually a conversion of photonic flux to electronic flux. This conversion takes place using a charge-coupled device (CCD).

The method in which a document is captured affects the subsequent usefulness of the document. Consider a faxed document. Although usually human readable, these documents are often not very machine readable. This is usually directly related to the fax capture process. Because fax machines typically communicate over phone lines, fax scanning resolutions are set to low resolutions to keep the file size transmitted as small as possible. So, for example, normal fax mode is 203x98 dpi, which means that the vertical sampling rate is less than 100 dpi. This poor scan rate might result in a smaller size CCITT file that needs to be encoded and transmitted. This fax-scanned file might also transfer faster and still be human readable on the receiving fax end. However, since this file was captured under less than ideal scanning conditions, at very low resolution, there is a high probability that machine text readability, aka OCR, recognition rates are not very high.

So there is generally this tradeoff between capture resolution and recognition rates. The higher the scanning resolution, up to say 300 dpi, the higher the OCR text recognition rates.

A similar relationship exists between color depth and OCR-based recognition rates. Namely, the greater the bits per pixel, the better the OCR recognition. Consequently, the same document scanned at 150 dpi (dots per inch) in both bitonal (black and white) and greyscale will have better recognition rates for the file captured to greyscale. If the file size is reduced by excessive JPEG quantization before OCR, this will also negatively impact on the OCR recognition rates.

There is usually some degree of skew, or page slant, during the capture process. This is true for manually fed and auto-feed devices. Many capture devices have some image processing capability that includes deskew, despeckle, and thresholding.

> **"So there is generally this tradeoff between capture resolution and recognition rates. The higher the scanning resolution, up to say 300 dpi, the higher the OCR text recognition rates.**
>
> **—Ari Gross**

THRESHOLDING WITHIN OCR

Thresholding is the simplest method of grouping an image into regions, aka image segmentation. In the case of thresholding, there are only two types of pixels: foreground and background. In general, the foreground pixels correspond to the text and the background pixels correspond to everything else, e.g., background texture, embedded images. Individual pixels in a grayscale image are typically marked as “object” pixels if their value is greater than some threshold value and as “background” pixels otherwise. Typically, an object pixel is given a value of “1” while a background pixel is given a value of “0.” This method employs a static threshold, namely, one value is used to threshold the entire page.

The key parameter in thresholding is obviously the choice of the threshold. Several di□erent methods for choosing a “static” threshold exist. The simplest method would be to choose the mean or median value of the image, the rationale being that if the object pixels are brighter than the background, they should also be brighter than the average value. In a noiseless image with uniform background and object values, the mean or median will work quite well as the threshold. In many situations, however, this will not be the case.

A more sophisticated approach might be to create a histogram of the image pixel intensities and use the valley point as the threshold. The histogram approach assumes that there is some average value for the background and object pixels, but that the actual pixel values have some variation around these average values. However, computationally this is not as simple as we’d like, and many image histograms do not have clearly defined valley points. Ideally we’re looking for a method for choosing the threshold which is simple, does not require too much prior knowledge of the image, and works well for noisy images.

Clearly, if the image page contains both video, i.e., dark text on light background, and reverse video, i.e., light text on dark background, then a single static threshold for the page will not su□ce. A more complex thresholding algorithm may first try to segment the image into diffierent backgrounds, not assuming a uniform image background. Then, for each background region, a static threshold value is selected. Methods such as this one, that are static for some local region but not for the entire image, are sometimes referred to as semistatic.

Of course, even the above method has its limitations. So for a book page where the background intensity varies smoothly this method may not be appropriate. Undersampled text, or documents that are cell phone scanned, may need special treatment including upsampling prior to thresholding. Gradient methods, akin to edge detection used in computer vision, may sometimes be appropriate for hard to threshold images.

HOW TEXTURE PATTERNS RELATE TO OCR

Many methods in OCR and image processing make assumptions about the image background. Often, a constant background is assumed. A texture can be defined as a tessellated, approximately repeating pattern in an image. This

texture might be real, e.g., wood paneling, or synthetic, e.g., the screening pattern caused by a color printer to represent a constant color background. Sometimes, it is beneficial to descreen the image before thresholding or further image processing.

> **Understanding textured regions can be very complex, but is sometimes necessary for proper separation of foreground and background.**

Solving for texture patterns is very helpful in segmentation and MRC based coding. Effective compression of scanned documents, and reliable OCR output, require accurate background foreground discrimination. When lifting the foreground in segmentation or MRC coding, the original background pattern, or some facsimile thereof, must be reconstituted. There are several ways to do this, which include building up a Markov model of the original texture and using this statistical model to regenerate the background regions that need to be covered. Alternatively, one can find a tessellation element and replace the lifted text region with a “pure” background region.

SMALL FONTS & OCR

One of the problem areas in reliable OCR and data extraction is small fonts. When the fonts are large enough, almost any threshold will preserve the topology and geometry of the underlying font characters. When the font size gets small, it is hard to find a good threshold such that readability of the text regions are preserved. It is important to identify small font regions before thresholding, rather than after. Depending on the font size, there are different recommended methods for converting the greyscale or color text region into bitonal prior to OCRing the region.

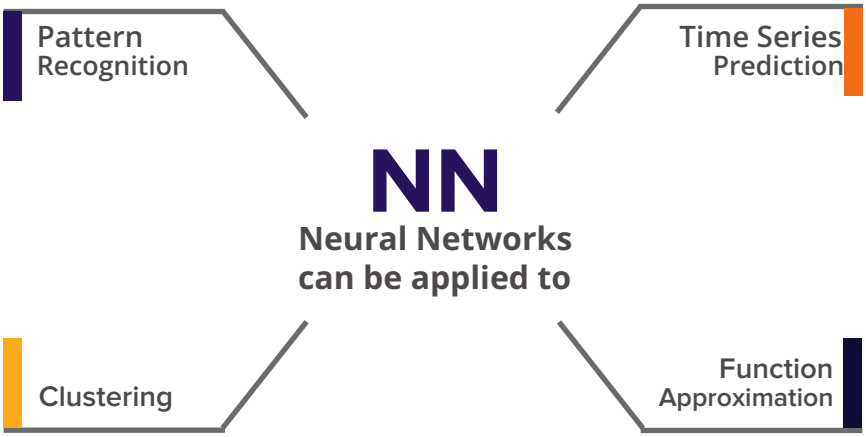
Certain font sizes (e.g., font size 10, 300 dpi scan) will still support a semi-static region threshold and preserve topology. It is also important that, to some extent, geometric properties of the characters be preserved as well. We do need to recognize these characters in an OCR sense, and maybe allow for JBIG2 style font matching. So for certain font sizes and text region backgrounds, a semi-static threshold will do. If the background has a lot of texture, then even though the font size is reasonably large, some preprocessing (before thresholding) may be necessary.

For very small fonts, that fall below the Nyquist sampling rate, standard thresholding is not adequate. These fonts must be handled very carefully or readability will be lost. These small font text regions can either be given directly to the OCR engine or upsampled to a higher resolution space and then thresholded. It might be preferable to sharpen the greyscale text region before thresholding.

OCR, NEURAL NETWORKS AND OTHER MACHINE LEARNING TECHNIQUES

Introduction

There are many different approaches to solving the optical character recognition problem. One of the most common and popular approaches is based on neural networks, which can be applied to different tasks, such as pattern recognition, time series prediction, function approximation, clustering, etc. In this section, we'll review some OCR approaches usingNeural Networks (NNs).



A Neural Network (NN) is a wonderful tool that can help to resolve OCR type problems. Of course, the selection of appropriate classifiers is essential. The NN is an information processing paradigm inspired by the way the human brain processes information. Neural Networks are collections of mathematical models that represent some of the observed properties of biological nervous systems and draw on the analogies of adaptive biological learning. The key element of an NN is its topology. Unlike the original Perceptron model, shown by Minsky and Papert to have limited computational capability, the NN of today consists of a large number of highly interconnected processing elements (nodes) that are tied together with weighted connections (links). Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons.

This is true for NNs as well. Learning typically occurs by example through training, or exposure to a set of input/output data (pattern) where the training algorithm adjusts the link weights. The link weights store the knowledge necessary to solve specific problems. Originating in the late 1950s, neural networks didn't gain much popularity until the 1980s.

Today NNs are mostly used for solution of complex real world problems. They are often good at solving problems

that are too complex for conventional technologies (e.g., problems that do not have an algorithmic solution or for which an algorithmic solution is too complex to be found) and are often well suited to problems that people are good at solving, but for which traditional methods are not. They are good pattern recognition engines and robust classifiers, with the ability to generalize in making decisions based on imprecise input data. They offer ideal solutions to a variety of classification problems such as speech, character and signal recognition, as well as functional prediction and system modeling, where the physical processes are not understood or are highly complex. The advantage of NNs lies in their resilience against distortions in the input data and their capability to learn.

Backpropagation NNs

A popular and simple NN approach to the OCR problem is based on feed forward neural networks with backpropagation learning. The basic idea is that we first need to prepare a training set and then train a neural network to recognize patterns from the training set. In the training step, we teach the network to respond with the desired output for a specified input. For this purpose, each training sample is represented by two components: possible input and the desired network's output given that input. After the training step is done, we can give an arbitrary input to the network and the network will form an output, from which we can resolve a pattern type presented to the network.

Example of OCR-Based NN Font Learning Using Bitmaps

For example, let's assume that we want to train a network to recognize 26 capital letters, represented as images of 16x16 pixels. One of the most obvious ways to convert an image to an input part of a training sample is to create a vector of size 256 (for our case), containing a "1" in all positions corresponding to the letter pixels and "0" in all positions corresponding to the background pixels. In many NN training tasks, it's preferred to represent training patterns in a so called "bipolar" way, placing into the input vector "0.5" instead of "1" and "-0.5" instead of "0". This sort of pattern coding will often lead to a greater learning performance improvement.

For each possible input we need to create a desired network's output to complete the training samples. For the OCR task at hand, it's very common to code each pattern as a vector of size 26 (because we have 26 different letters), placing into the vector "0.5" for positions corresponding to the pattern's type number and "-0.5" for all other positions.

After having such training samples for all letters, we can start to train our network. But, the last question is about the network's structure. For the above task we can use one layer of neural network, which will have 256 inputs corresponding to the size of input vector and 26 neurons in the layer corresponding to the size of the output vector. At each learning epoch, all samples from the training set are presented to the network and the summary squared error is calculated. When the error becomes less than the specified error limit, then the training is done and the network can be used for recognition.

Example of OCR-Based NN Font Learning Using Feature-Based Classifiers

The approach described above works fine, but is limited in its extensibility. There are some issues that a generalized, robust NN-based OCR system needs to handle, which include font and scale variations. Giving an NN system bitmaps as input is somewhat problematic since humans don't see characters at the pixel level, nor is the "essence" of a character font conveyed by this pixelized representation. When there are considerable bitmap variations in the definition of each font character, a better set of inputs to represent the data would be a set of classifiers, computable from the bitmap images, such that these classifiers are invariant to changes in font and point size.

Such classifiers might include topological characteristics, such as Euler number, compactness, and geometric properties, e.g., concave up. Of course, these features now need to be computed from the input images and given as input to the NN system. In addition, the system is invariant to changes in font and point size, so it cannot classify beyond labeling an input bitmap as say an "e", when we may want additional information such as the font and point size, e.g.,"e", point size: 12, font: Times Roman. The point is that features typically provide some level of invariance, but at the same time, limit the degree of recognition.

In this case, since there is wide variation in font definitions, we could first have an NN-based OCR system that is invariant to font and scale to recognize the character. Once we know it's an "e", we can match it against all "e" font definitions in our font database to establish the exact font and point size.

OCR, CRYTORITHMS, CRYPTOGRAMS AND SUBSTITUTION CIPHERS

Cryptorithms

Cryptorithms are puzzles where the digits in an arithmetic computation are replaced with letters. The puzzle is presented with the letters, and the object is to find out what the corresponding digits are. A famous example is:

```
SEN D + M O R E
---------------
MONEY
```

Another, somewhat simpler example is given by:

PYX +
PYX

YYP

Here, the trick is, like in a crossword puzzle, to start where the puzzle is "easiest" to break. In this example, we note that in the 2nd column we have that Y + Y + {0,1} = {0,1}Y. But Y cannot be 0 as YYP is the sum of 2 numbers whose leading digits would be non-zero. But then X + X must involve a 1-carry since otherwise Y+ Y + 0 cannot equal Y, for non-zero Y. This forces Y to be odd, since odd + odd + 1 is odd, but even + even + 1 is not even. But then Y must equal 9 since only 9 satisfies the constraint that 9 + 9 + 1 = {0,1}9. Since Y = 9, we also have the constraint from the leftmost column that P + P + 1 (1-carry from column 2) = 9. So that P = 4. We know from our analysis thus far that X > 4 since X + X results in a 1-carry to column 2. But then X + X = 14, so that X = 7. This is how a simple cryptorithm is solved.

Of course, they can get more complicated. Try the first cryptorithm problem, given above. Analysis there would again start from the easiest letter to break, leading us to conclude M = 1.

Cryptograms and OCR

In cryptography, a substitution cipher is a method of encryption by which units of plaintext are substituted with ciphertext according to a regular system; the "units" may be single letters (the most common), pairs of letters, triplets of letters, mixtures of the above, and so forth. The receiver deciphers the text by performing an inverse substitution. A cryptogram is defined as a short piece of text encrypted with a simple substitution cipher in which each letter is replaced by a di□erent letter. To solve the puzzle, one must recover the original lettering.

Here is a simple example: **CAEEAEEAOOA Answer: Mississippi**

We can, quite naturally, view certain OCR problems in a similar vein. Of course, in analyzing scanned documents we cannot always assume that each connected component in the image corresponds to a symbol in the alphabet. We have to deal with oversegmented and undersegmented images. In the oversegmented case, more than one model is required to comprise certain letters in the alphabet.

This can happen if the document is not thresholded correctly, or if composite topological structures, such as "i" and "j", are not combined into single models. In the undersegmented case, one component comprises more than one symbol in the alphabet. This happens often with certain letters such as "fi" and "th". For best OCR results, these undersegmented cases need to be broken.

HUMAN AND MACHINE READABILITY & OCR

There is a gap between human and machine readability. What does this mean exactly? Well, consider the websites that rely on "CAPTCHA" to distinguish between humans and bots. These websites are relying on the fact that there exist images where the text is human readable, but not machine readable.

What is CAPTCHA? A CAPTCHA is a type of challenge-response test used in computing to determine whether the user is human or not. "CAPTCHA" is an acronym for "Completely Automated Public Turing test to tell Computers and Humans Apart", trademarked by Carnegie Mellon University. A CAPTCHA involves one computer asking a user to complete a test. While the computer is able to generate and grade the test, it is not able to solve the test on its own.

Because computers are unable to solve the CAPTCHA, any user entering a correct solution is presumed to be human. The term CAPTCHA was coined in 2000 by Luis von Ahn, Manuel Blum, Nicholas J. Hopper (all of Carnegie Mellon University), and John Langford (of IBM). A common type of CAPTCHA asks the user to type in the letters of a distorted image.

Typical document scanning takes place in the 200-300 dpi range. In that range, basic topological and geometric properties are preserved, even after thresholding (i.e., converting scan document to black and white). At low resolution scan rates, however, machine OCR systems run into trouble. Some of the reason for this disparity is that humans are adept at reconstructing the shapes of characters even if multiple characters share a pixel. For computers, if distinct characters are not separate in the image after thresholding, there is often a sharp decrease in recognition rates. Usually, an image is adequately sampled if each letter is at least two pixels in thickness; the same applies to white space. When sampling is below the Nyquist sampling rate, such that this constraint is clearly not satisfied, machine recognition fails entirely, while human recognition remains intact until perhaps 25 dpi.

Wherein lies the difference between human and machine readability? For example, there is an explosion in cell phone use worldwide, with the expected number of units to exceed one billion by the end of 2008. Many of these users will have the ability to capture images, including documents. For OCR to work effectively at these cell scan rates, which for documents is well below 50 dpi, there need to be fundamental improvements in OCR technology.

> **"For computers, if distinct characters are not separate in the image after thresholding, there is often a sharp decrease in recognition rates.**
>
> **—Ari Gross**

RELATIONSHIP BETWEEN OCR & JBIG2

There is a clear connection between OCR and the new ITU bitonal JBIG2 standard. In particular, an important aspect of JBIG2 is font learning. Whereas in the previous CCITT4 TIFF image specifications there was no notion of fonts, or font learning, it is a very important part of the JBIG2 compression specs and is one of the main reasons that JBIG2 compression rates are as high as 10:1 with respect to TIFF G4 compression.

Of course, font learning is important for OCR performance as well. When a font is “learned” it imposes constraints on all the connected components that map to that font character. One of the aspects of JBIG2 is font models, another aspect is global models, and a third is composite model. Each of these is not only useful for compression purposes, but also for effective OCR rates. Models, assuming a perfect font matcher, impose intra-page node constraints, but do not impose any constraints between nodes on different pages. Global models impose inter-page constraints on nodes linked to the same global font model.

OCR & NOVEL FONTS

In classical OCR, the recognition systems were trained on a very specific set of fonts. If these fonts varied in any material way, recognition rates would fall o accordingly. Today’s systems are much more robust and can handle the myriad of novel fonts that are used in publishing and available on the Web. What becomes more relevant for modern OCR systems is adaptability. If the shapes of characters in a new font is fairly unpredictable, what can be relied upon? It would be nice if, at the very least, topological properties, e.g., Euler number, are preserved. But often even this property is not invariant either due to novel fonts that modify basic character topology or because of scanning noise that introduces or eliminates holes.

As a result, what has become more prevalent in recent OCR technology is “shape- free” OCR. These algorithms seek to find the appropriate mapping between learned font symbols and the symbol alphabet. These newer methods seek to solve the OCR problem relying heavily on order statistics. Among the methods used, numbered strings that make use of the word structure to limit or uniquely identify the correct mapping. Obviously, the longer the document being analyzed, the more relevant the document statistics (such as K-tuples) will be.

It would seem that shape-only OCR systems have somewhat limited in applicability. Such systems want to solve the OCR puzzle strictly from the shape of a component image. This method can also be referred to as context-free, since no neighboring context is required to solve for the correct ASCII mapping. Similarly, OCR methods that are highly statistical can be thought of as contextsensitive, as these methods want to first compute order stats, or k-tuples, and only then infer the ASCII mapping. A combination of context-free and contextsensitive methods, incorporating geometric and topological properties of each component in conjunction with shape-free statistical methods, is probably most likely to yield accurate OCR results.

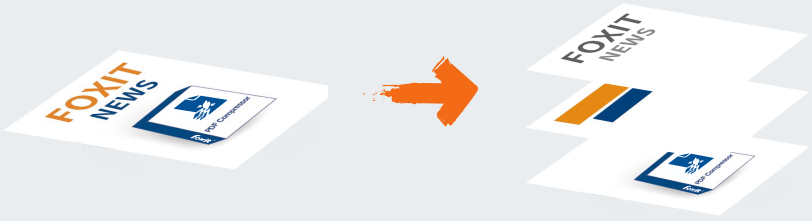
OCR, MRC & JPEG2000

One of the novel aspects to JPEG2000 is that, although an algorithm for segmenting color images is not specified, the JPEG2000 spec does support segmentation-based coding. In fact, the most effective rates for color compression are obtained by analyzing, understanding, and reversing the page layout process. It becomes important in JP2 segmentation-based coding to be able to separate foreground from background, and more specifically, text regions from non-text regions.

Unlike many OCR systems, which can "tolerate" missing text regions that are not aligned horizontally or vertically, color compression using Mixed Raster Content (MRC) coding or based on JPEG2000 part 6, is much less forgiving. The basis of MRC coding is separating out the high frequency signal information from the low frequency information. Usually, the high frequency information in an image is text-related. Of course, there is also line art, edge structures, and other objects that may degrade when kept at low resolution. But text regions must be recognized and lifted for MRC-based compression to be non-degrading. This necessitates finding all text regions, regardless of skew, rotation, etc.

Accurate detection of all text regions is very helpful in improving OCR accuracy. In this way, a system that supports MRC-based or JPEG2000 compression coding (like Foxit) will benefit with respect to improved OCR as well. The detected text regions can be fed directly to the OCR engine to be sure that no image text is left unsearchable. Accurate OCR and perceptually lossless compression for color images both rely on robust, precise page segmentation of foreground text from background image. This segmentation leads to the best color compression rates and the most reliable OCR.

Mixed Raster Content (MRC)



REVERSE VIDEO & OCR

Reverse video detection is important for high fidelity OCR recognition. Many color documents have text that is lighter (rather than darker) than its background.

There are many OCR engines that rely on certain assumptions, like text being darker than background. This is a limitation in several commercial OCR products that are fairly highly rated, yet do not perform well given a reverse video recognition test.

In the reverse video recognition test, a set of color image documents where the text is consistently darker than the background is given to an OCR engine. With this "video" set of document scans a baseline recognition rate is established for the OCR system. Next, the reverse color mapping (aka color complement) of the original images is computed. This 2nd image set is now considered and the engine's OCR accuracy for these reverse video documents is determined. If the OCR's system performance on the reverse video documents is statistically worse than on the original set then this OCR engine is not invariant to reverse video transformations and is biased towards relatively well-behaved dark text on light background. Obviously, an OCR recognition system that has no such bias (e.g., Foxit's OCR) is more highly rated.

LOCATING MULTIDIRECTIONAL TEXT WITH OCR

Multidirectional text is one area where current commercial OCR systems fail. In most commercial systems, a dominant text direction is found. Text is then OCRed along this dominant direction, but not along any subdominant directions. If text on the same page occurs in multiple directions, such as horizontal and vertical text occurring at the same time, it is generally recognized only in the predominant direction. Many documents, e.g., patent litigation, have important text running in multiple directions. The ability to detect these text regions, in any direction, and OCR them is a good litmus test for any commercial OCR system.

Foxit OCR (PDF Compressor) passes this test. Most commercial systems do not.

OCR & HOW THEY RELATE TO MFPS (MULTIFUNCTIONAL PERIPHERAL DEVICES)

MFPs, short for Multifunctional Peripheral and also known as multifunction printers, are devices that provide several functions including printing. Multifunction printers can act as printers, scanners, fax machines, and photocopiers.

These devices are becoming a popular option because they're less expensive than buying three or four separate devices.

The downsides to combining all these functions in one device are:

i. If the device breaks, you may lose all of its functions at the same time;
ii. You can only do one operation at a time. For example, you can't print a document and receive a fax simultaneously;

As many companies integrate MFPs into their workflow, the notion of document capture is changing. Whereas documents might have been sent to a scanning center in the past, with MFPs a company can capture documents and optionally insert them into a database in a totally distributed environment. This has the benefit of making document capture much more routine at the corporate level, but also requires that any document capture process be as foolproof as possible.

Many MFPs have support for OCR in some form. This support may be directly on the MFP device, but more likely it is actually on a nearby server that can communicate with the MFP. For example, Foxit's PDF Compressor handles MFP files through its watched folder mechanism.

What is desired generally at the MFP level is a simple way to allow the user to route, name, and modify a document, with possible insertion into a database. Modification of the document might include OCR and metadata insertion into OCR & How They Relate to MFPs (MultiFunctional Peripheral Devices) the scanned PDF file. These functions are often controlled at the MFP, either using the MFP control panel or via a cover sheet. The cover sheet has the advantage of also serving as a file delimiter, allowing many files to be processed at the same time. It also has the advantage that a complete specification, including routing the document, and emailing to a group of people, can be specified on the cover page. The control panel, on the other hand, requires no advance preparation, no stops at the printer, and is easy to use.

OCR & UNDERSAMPLED TEXT

Undersampled text is a serious problem for OCR engines. Although people have no problem with typical undersampled text, machines do have a recognition problem. Most OCR engines do NOT handle undersampled text well, and this is currently an area where there is considerable disparity between human and machine recognition rates.

If there is some control over the document capture environment, it is highly advisable to scan at 300 dpi. With new compression formats available (JBIG2, JBIG2 PDF, MRC-coded PDF, JPEG2000), there is very little reason not to scan to higher resolution. TIFF G4 compression increases linearly with the scanning resolution, so that a 300 dpi scan is about 2x the size of a 150 dpi scan. With JBIG2-encoded perceptually lossless PDF, however, a 300 dpi scan is actually smaller than a 150 dpi scan. This is because the font library is minimal, with no topologically false connections or disconnections.

In any event, if there is control over the document capture process, current OCR methods being what they are, a higher dpi is strongly recommended (e.g., 300 dpi) for attaining most accurate OCR results. The file size will not go up (using JBIG2) and the OCR recognition rates will be significantly better than at lower scanning resolution.

If one has documents already sampled at low resolution scanning rates, proper (post) processing of these files is also necessary to achieve good recognition results. First, if these files are currently in greyscale or color formats (e.g., JPEG) then we would strongly suggest not thresholding prior to OCRing. Rather, these documents should be upsampled to 300 dpi, preferably using bicubic splines. These upsampled greyscale or color image documents should then be presented directly to the OCR engine.

> **"Most OCR engines do NOT handle undersampled text well, and this is currently an area where there is considerable disparity between human and machine recognition rates.**
>
> **—Ari Gross**

DICTIONARY LOOKUP & OCR

In comparing and rating OCR systems, recognition rate is usually the most significant feature. Recognition rates can be improved often by using dictionary lookups to aid in the recognition process. Essentially, when a dictionary is used to help in the OCR process, the system will bias its recognition in favor of words that are in the dictionary. Of course, there are many strings in a document that might not be in any dictionary, like an invoice number. So even if the system is using dictionary lookup, it must allow for strings that are not in the dictionary.

When using dictionary-aided OCR, there are different functions to determine which word in the dictionary is closest to a given string. A very typical distance metric that is used to measure word distance is edit distance. The edit distance is defined recursively, and can be implemented effciently in a dynamic programming (DP) framework.

A commonly-used bottom-up dynamic programming algorithm for computing the Edit distance involves the use of an $(n + 1) \times (m + 1)$ matrix, where n and m are the lengths of the two strings.

Here is pseudocode for a function EditDistance that takes two strings, s of length m, and t of length n, and computes the Edit distance between them:

```
int EditDistance(char s[1..m], char t[1..n])

    // d is a table with m+1 rows and n+1 columns
    declare int d[0..m, 0..n]

    for i from 0 to m
    d[i, 0] := i

    for j from 1 to n
    d[0, j] := j
    for i from 1 to m
    for j from 1 to n
        if s[i] = t[j] then cost := 0
else cost := 1
        d[i, j] := minimum(
            d[i-1, j] + 1, // deletion
            d[i, j-1] + 1, // insertion
            d[i-1, j-1] + cost // substitution
            (
return d[m, n]
```

Intuitively, we are interested in understanding how close a given string A is to a known dictionary word B. One way to measure the distance between any two strings is to compute the number of keystrokes or edits required to transform string A into string B. If this value can be computed e□ciently for every string B in a given dictionary, then we can determine the dictionary word B that is closest to some string A. If this word is sufficiently close to the desired string, we assume the correct ASCII mapping for this string is B.

Dictionary-based OCR is usually advantageous and yields overall better recognition rates. Among the drawbacks: the system tends to run slower, and sometimes technical terms are “wrapped” into dictionary words.

RATING AN OCR SYSTEM

Speed Accuracy

Rating an OCR system is a little tricky. Very often, one OCR system might excel on one type of document and another OCR system might excel on a 2nd document type. The best OCR test, unless one is testing for Consumer Reports, is to test on the company's own documents. Some documents are first generation, clean scans. Other corporate documents are 3rd generation re-scans. While one can analyze a system generally, on a broad spectrum of documents, the most relevant testing is done on company-specific documents.

Features that minimally need to be rated are generally OCR accuracy and processing speed. Of course, in any such test there also needs to be a familiarity with the OCR controls. Many OCR systems allow the user to control speed vs. accuracy. The fast setting can often be 3x-9x faster than the slower, most accurate setting. So any such test, needs to be "apples to apples".

OCR accuracy can be tested programmatically, by running the OCR output through a dictionary lookup. It can also be tested manually, by having a human go and match the OCR output against the original documents. The programmatic system tends to be a lot faster, but some accuracy might be lost (e.g., valid OCR responses to items like invoice number or part number, or non-English phrases).

TWEAKING THE SYSTEM TO OPTIMIZE OCR PERFORMANCE

Within the domain of OCR testing and evaluation is controlling the document formation process. There are times, for example, when the OCR recognition rates are poor due to document imaging conditions that can be changed. For example, sometimes the background invoice color is not white, but maybe a textured blue. The foreground is black.

In the black and white space, which is how the invoice might be captured, the texture elements are partially lifted which interfere with the recognition rates. A solution here may involve modification of the document capture process to allow for color capture and thresholding in the color space so that the texture and text are properly delineated.

Another example might be that there is a form (in black) and foreground text (also in black), and there is interference between the form and foreground text. This might include text stuck to grid lines of the form. If the form is predictable, or from a set of forms known a priori, then this form can be solved for by the system and can be "removed" prior to the OCR process.

OCR TO CREATE SEARCHABLE PDFS

In the new "paperless" office, there is no more paper. Any file, contract, meeting notes, etc. you need are found instantly with Google-style searching. To make all this happen, paper needs to be made searchable. Pure OCR is usually not the answer for a variety of reasons. Among them pure paper to electronic conversion is not a reliable process. For most corporate documents, before searchability is a factor there must be a guarantee of accuracy in the conversion process. Any process that can modify the look and feel of the document is not acceptable.

Paper to electronic format conversion is not reliable for purposes of record management. The OCR engine attempts to understand the document logically, and then reconstruct it electronically. Any error on the part of the OCR engine in interpreting the document will result in perceptual errors in the electronic reconstruction. Instead, what has become heavily used in the document management industry is that of searchable image documents. Within searchable image documents, the most popular format is that of PDF image + hidden text. In this format, what you "see" is just the imaged document. What you can search on, however, is the fully OCRed version of the file. Since the OCR layer is hidden, mistakes and substitution errors at the OCR level are not visible on the document.

Searchable PDF is an ideal format for several reasons. One of the main reasons for searchable PDF is database portability. When porting between databases, it is very good practice for documents to be self-contained. This essentially means that OCRed text and any document-related metadata should be available within the document. With PDF this is easily realizable. With other formats, such as TIFF, it is virtually impossible. Of course, other information including headers, footers, Bates Stamping, and security can easily be added to a PDF file.

OCR & LOGICAL DECOMPOSITION

Many OCR users want basic search from an OCR engine. This means that they want to find the needle in the haystack. They need to search a database and find all files that contain a certain expression. This type of OCR does not depend on a logical decomposition of the document image. It is sufficient to get back all the text associated with

each page of a document image and feed the OCR text to a full-text search database engine. The database will then index on the full-text and allow general text-based database queries, e.g., proximity search.

There are times, however, when a logical decomposition of the document is required. This happens when part of the document is to be used in composing another document. In this case, the document needs to be logically understood, including word readability order, tables, and graphs, so that an excerpt can be utilized as part of another document. Certain OCR processes need this logical decomposition, and looking at OCR word accuracy is not sufficient in evaluating OCR systems for these applications.

ELECTRONIC FILE CONVERSION & OCR

Electronic file conversion refers to converting a file into electronic form. The look and feel of the original file is hard to preserve when converting to pure electronic format. For the purpose of records retention, such electronic documents are usually unacceptable. Having made these disclaimers, electronic file conversion is generally more compact than image based documents. For files that start off in electronic format, such electronic conversion is considered more acceptable.

BAR CODES, OCR & ICR

Bar Codes and ICR are both types of data that can be recognized in a commercial OCR system. However, they lie at opposite ends of the image recognition spectrum. Bar codes are considered more reliable than conventional machine printed OCR. As such, they are used in many commercial applications, like UPC codes, and considered very reliable. Of course, this reliability comes as a result of a very constrained alphabet, i.e., straight lines of varying widths.

On the other end of the spectrum is ICR, which is an acronym for Intelligent Character Recognition. ICR is really analogous to OCR except that OCR is for machine printed text while ICR is for handwritten text. While many consider the OCR problem solved, the $20 billion spent by industry on field coding suggests otherwise - namely, that reliable recognition and data extraction is far from solved. On the other hand, general ICR is a very hard problem where there are almost no commercial systems today that run reliably in this environment.

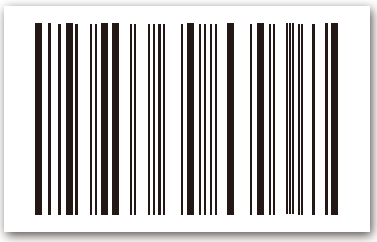
Viewed another way, bar code recognition is a more constrained problem than OCR. While OCR seeks to understand all printed characters in a document, bar code recognition seeks to understand only characters that are bar codes, which are lines of varying thickness. ICR is more general than OCR, seeking to recognize hand printed alpha-numeric characters, and certainly machine printed ones.

While ICR is not generally viable as a commercial technology, there are applications where this technology might be viable. If the string to be recognized has to fit a rigid profile, such as a date or social security number, then recognition rates are much higher. ICR recognition rates for strictly numeric data are generally much higher than for mixed alphanumeric data. Also, ICR recognition is much higher when there is data redundancy, as in check deposit slips.

OCR:

ICR:

Bar Code:



OCR & FORM RECOGNITION

Form recognition is an area very relevant to the document imaging industry. In fact, it is one way for a company to save money by automating processes that are now done manually. It is estimated that industry spends as much as $20 Billion annually on field coding, which means taking information that people have written or typed on form documents, invoices, etc., and keying this information into a database. Certainly, some of this work could be automated - the question is, how much?

There are multiple problems involved in “form recognition”. The most straightforward forms recognition to solve is to recognize a “fixed form”, where the form always has the exact same appearance. Even in a fixed form environment, where the form type can be detected with almost absolute certainty, there can be problems. Assume the task at hand is to identify the precise form type and then extract certain fields. It’s possible that the fields are handwritten, or even if machine printed, that OCR rates are less than 100%. So what needs to be learned is not just the form characteristics, but also the constraints on the different fields to be extracted, e.g., date field. For handwritten documents, ICR is less than reliable so redundancy can also be a key factor in reliability. If there are multiple fields on the form that give the same or database-related pieces of information, these can be combined to yield a much higher recognition accuracy.

There are forms that are not fixed. Examples can include bank transaction statements that resemble business letters and differ based on issuing bank. There are Dept. of X files on an individual, where X could be Housing, Corrections, Employment, Education, etc. These documents may differ based on State of issue, and within each State, differ by County. The forms again may not be fixed, but may vary in structure. The field information may be embedded somewhere in the document.

Most form recognition problems where companies could potentially see serious ROI with a fully- automated or semi-automated recognition system, are beyond the capabilities of current off-the-shelf OCR, form recognition, and data extraction systems. That does not mean, however, that a solution cannot be engineered to a company’s specifications based on a company’s unique set of forms to be processed, data to be extracted, possible data redundancy, and other factors. Any system that involves 3 or more full-time data entry personnel, from menial data entry to more complex data entry and analysis is a candidate for automation (or at least semi-automation). Consulting a company with the right expertise in the area of form recognition (e.g., Foxit) can make all the difference.

DATA EXTRACTION WITH OCR

The data extraction problem is very closely coupled with form recognition. Usually, when a company needs data extraction it is in the context of form recognition. This means that one cannot extract meaningful data in the absence of recognizing the form type. This is distinct from the general OCR problem. The general OCR problem is to extract as much meaningful text from an image document as possible. There is no assumption about prior knowledge with respect to this document, other than perhaps what language the document is in. So OCR is ideal for full-text search where a database index needs to be constructed to allow for arbitrary text-based queries. But general OCR is not ideal for field coding, when certain fields need to be precisely coded into the record of a database and they must be entered correctly or there may be no way to find this document later.

For a reliable automated or semi-automated data extraction / field coding system to work, characteristics of the application need to be known ahead of time. These are aspects the system needs to train on to have effective recognition rates. For example, if the system being constructed or maintained is a University database then the

fields necessary for each student record must be known a priori. In addition, whatever constraints are available for each record field must either be explicitly entered, e.g., XML-based file, or learned by the system during training. So the data extraction system, if looking to code a social security field per student, should know that the field is numeric, consisting of exactly 9 digits (with possible embedded "-"'s).

FORMS, DATA CONSTRAINTS AND REDUNDANCIES: CHECK DEPOSIT

There are many factors to solving the data extraction problem correctly. Among them are form-based constraints, data constraints, and data redundancies. For example, these three factors are all very useful in accurate coding of check deposit information. When you drop your checks off for deposit, there are usually some checks and a deposit slip. The deposit slip is usually handwritten, Data Extraction with OCR though company related information like account number may already be printed on the deposit slip. The checks themselves are either handwritten or machine printed. Routing and bank branch information are encoded on the bottom of each check using special numeric symbols that are easily recognizable.

One of the issues in solving this problem reliably is that the check deposit process usually contains handwritten data, and handwritten data recognition is still considered largely an unsolved problem. However, there are some data redundancies and form & data constraints that make the problem largely solvable. In particular, on the deposit slip, which is basically a form, there are boxes for each numeric character. This does not allow the user to write unconstrained cursive for the dollar amount. It also handles the difficult segmentation problem, as each numeric character has already been isolated. In addition, the state of numeric handwritten character recognition is significantly higher than unconstrained handwritten character recognition. Furthermore, the check deposit slip asks for each check amount, even though it is already on the check, and the check deposit total is requested twice. So there is redundancy with respect to each check amount, redundancy with respect to the total deposit, and additional redundancy in that the sum of all the checks (and other deposits) must ADD up to the total deposit amount. The semi-automated check deposit system in place at many large banks takes advantage of all these constraints and redundancies and, as a result, processes the average check for considerably less cost than 10 years ago, i.e., pre-automation.

I KNOW THAT I DON'T KNOW THAT I KNOW...

What is very important in the design and implementation of any field coding / form learning / data extraction systemis to know what you know. And what you don't. The reason for this is simple: If an automated system performs correctly the Company saves money and see ROI (return on investment). If the automated system makes mistakes that go UNDETECTED, even occasionally, it could cost the company a lot more in correcting the situation than the automation saved.

Going back to the semi-automated check deposit example: if the system correctly recognizes all the dollar amounts, on both checks and deposit slip, 90% of the time, is this a win for the Bank or not? The answer is totally dependent on whether the system knows what it knows. Meaning, if the system knows when a numeric value MAY be incorrect because all the numeric information, which is heavily redundant, is not in sync then any such case can be shown to a human operator without penalty so that the automation is a win for the bank. If the system, however, does not have the controls in place to verify the correctness of the extracted data then this system is probably not commercially viable, unless each transaction is shown to a human for the purpose of verification.

BUSINESS PROCESS AUTOMATION AND HOW IT RELATES TO OCR

Business Process Automation (BPA), also known as office automation, is the field concerned with identifying applications in a business that can be automated, then designing and implementing a solution. Unlike other areas in business, it is usually easy to make a business case for business process automation as any successful installment of such a system will reduce a Company's manpower costs. Thus, with BPA it should not be difficult to show a return on investment (ROI) in some reasonable time.

So if the challenge in office automation is not justifying the installation of an automated system from an ROI perspective, where do the difficulties lie?

There are at least 3 issues that typically get in the way of a company deploying an application-specific automated system (ASAS):

i. engineering-specific obstacles to overcome;
ii. integration into existing workflow;
iii. modification of existing processes.

Let's review each of these issues briefly:

i. Engineering-specific obstacles: Office Automation is generally not trivial and out-of-the- box solutions usually do not work. It is easy to get frustrated when your IT department can't get the problem solved. Fact is that most automation solutions are complex, push the envelope with respect to current technology, and need to be engineered by experts.

Q: But if the automation venture has a strong upside but involves some risk, how do you minimize your Company's exposure and come out on top?

A: Have the specifications for the working, automated system very precisely defined before approving the project. This way there is no ambiguity as to how the system will operate. If the system operates to specifications, then the Company's IT department is sure the system will end up going online, and paying for itself. In addition, the Company should leave most of the risk with the automation engineering company, so that there is very little financial exposure until the system is up and running. On the other hand, the automation company (e.g., Foxit) will take the job only if it's sure it can get the job done to spec. In addition, there needs to be a significant upside for automation company once the system is operational for taking the risk in design, development, and implementation of the automation system.

ii. Integration into existing workflows: Of the 3 issues listed above, this one is probably the easiest to overcome. In most companies, an automated system would replace some system already in place. As a result, it must be integrated into an existing workflow. This is usually straightforward engineering, with no research component and little risk, if any. Thus, if the automated system will show a clear ROI to the Company once operational, the integration component should not get in the way of making the project viable.

iii. Modification of existing processes: Sometimes, in order for an automation application to be successful, certain processes within the application need to be redesigned. In our check deposit example, this was true for many Banks. For example, a financial institution had to redesign their check deposit forms to support automation by requiring that the deposit total amount be entered twice. Similarly, many automation projects require some aspect of redesign in order to satisfy some of the specifications for the project to be feasible. This part of an automation project is a little tricky since often more than just one group of the Company may need to get involved once issues like form redesign come up.

The relevant decision factor here is, and should be, return on investment. If someone at the Company is convinced that the automation project will yield the Company significant ROI long term, and the specs have been carefully put together, then process re-engineering should not get in the way. In particular, no re-engineering (other than on a prototype basis) should be done until the automation group produces a running system that is shown to satisfy the system specifications. Once this prototype has been developed, such that the risk factors have been eliminated (or greatly reduced), then it is time to reengineer the process workflow as needed for the system to go into production.

OCR-BASED ROI

What do we mean by OCR-based ROI? It is the ability of a Company to show a clear return on investment from converting its paper documents to searchable electronic files. This metric is not always easy to compute. Unlike automation, it does not necessarily involve reducing the workforce, but may instead make the work time of existing employees more productive. Having a Google-like ability to find documents at a law firm, bank, financial institution, or insurance company should greatly reduce a Company's cost of locating documents and make the related workers more productive. In the long term, this should involve a clear ROI for the Company, but might be difficult to quantify precisely.

If introducing fully text-searchable documents will reduce the size of the Company's file room staff or mailroom staff then the ROI case becomes rather straightforward. Of course, if conversion to searchable documents, say from TIFF to PDF image + hidden text format, is combined with compression than it becomes easier to show tangible ROI. If the document file size after converting to compressed, searchable, web-optimized PDF is 10x smaller than before then, searchability notwithstanding, one can make a clear ROI argument based on reduced storage requirements, reduced document-related transmission bandwidth requirements, and reduced web-hosting fees.

TOWARDS THE PAPERLESS OFFICE

Are Companies Moving Towards the Goal of a Paperless Office?

The answer seems to be both YES and NO.

If the issue being considered is: Are companies putting systems in place such that, eventually, all corporate documents will be fully searchable? The answer seems to be YES. Electronic Content Management (ECM), making sure all company information is available and searchable electronically, is an effective way for companies to manage their documents and information. And companies seem to be clearly moving in this direction.

The related paperless office question is: Are companies getting rid of office paper? The question here appears to be unequivocally NO. All indications are that offices are printing record amounts of paper. The amount of paper being used on the corporate level shows no indication at all of slowing down.

Why is it, if companies are moving in the direction of Electronic Content Management, that paper use is up? There are several reasons given to explain the elusive paperless office. One reason for increased paper use is that a lot more corporate searching and research is going on at the Internet level, but this is only for finding documents, not for reading them. If one finds a relevant research report, white paper, manual, etc. available online they are very likely to print it in its entirety, rather than reading it off the screen. Then if this document is sent to a co-worker, they are also liable to print it. In the pre Google days, there was a greater likelihood the physical paper copy would be passed around the office rather than copied several times.

Another reason why office paper use is on the rise, even with ECM, is that many documents are now kept electronically in the office. For example, a contract would no longer have a physical paper copy. On the upside, the corporate file room has been eliminated, downsized or moved offsite as relevant documents are all available electronically. In addition, the time to find relevant documents has been drastically reduced. The flip side of this is that document generation has becomes relatively easy such that these corporate files are generated from electronic copy "on demand". This streamlines many document related processes, but also creates an increase in company print utilization as paper documents are generated dynamically.