User-Directed Analysis of Scanned Images

Steven J. Simske, Jordi Arnabat
Imaging Systems Laboratory
HP Laboratories Palo Alto
HPL-2003-233
November 18th , 2003*

user interface, segmentation, classification, zoning, bottom-up analysis, preview display, scanning, click and select

Digital capture (scanning in all its forms, and digital photography/video recording), in providing virtually free temporary memory of captured information, allows users to "over-gather" information during capture, and then to discard unwanted material later. For cameras and video recorders, such editing largely consists of discarding images or frames in their entirety. For scanners (and high-resolution camera/video), such editing benefits from a preview capability that provides quick and reliable user interface tools for selecting, filtering and saving specific portions of the input. Appropriate preview user interface (UI) tools ease the accessing, editing and dispatch to desired destination (archive, application, webpage, etc.) of captured information (text, tables, drawings, photos, etc.). In this paper, we present several different means for the user-directed "rapid capture" of portions of a scanned image. Specifically, we review past, present and future preview-based UI tools that allow efficient and accurate means of capture to the user. The bases of these tools, as described herein, are user-directed zoning analysis, known as "click and select", which incorporates a bottom-up zoning analysis engine; and statistics-based region classification, which allows rapid reconfiguration of region identification and clustering. We conclude with our view of the future of UI-directed capture.

* Internal Accession Date Only Approved for External Publication
To be published in and presented at ACM Symposium on Document Engineering (DocEng 2003) 20-22 November 2003, Grenoble, France

User-Directed Analysis of Scanned Images

Steven J. Simske
Imaging Systems Laboratory
Hewlett-Packard Laboratories

Jordi Arnabat
Barcelona Imaging and Color Team
Imaging and Printing Group, Hewlett-Packard
October 10, 2003

ABSTRACT

Digital capture (scanning in all its forms, and digital photography/video recording), in providing virtually free temporary memory of captured information, allows users to "over-gather" information during capture, and then to discard unwanted material later. For cameras and video recorders, such editing largely consists of discarding images or frames in their entirety. For scanners (and high-resolution camera/video), such editing benefits from a preview capability that provides quick and reliable user-interface tools for selecting, filtering and saving specific portions of the input. Appropriate preview user interface (UI) tools ease the accessing, editing and dispatch to desired destination (archive, application, webpage, etc.) of captured information (text, tables, drawings, photos, etc.). In this paper, we present several different means for the user-directed "rapid capture" of portions of a scanned image. Specifically, we review past, present and future preview-based UI tools that allow efficient and accurate means of capture to the user. The bases of these tools, as described herein, are user-directed zoning analysis, known as "click and select", which incorporates a bottom-up zoning analysis engine; and statistics-based region classification, which allows rapid reconfiguration of region identification and clustering. We conclude with our view of the future of UI-directed capture.

Keywords: User Interface, Segmentation, Classification, Zoning, Bottom-Up Analysis, Preview Display, Scanning, Click and Select.

1 Introduction

A variety of scanning/digital capture devices benefit from having a preview capability that allows the user to select, or filter, specific information, from the overall image. Auto-cropping, for example, results in the removal of extraneous background (scanbed, table top, etc.). However, in many cases, it may not be clear what is to be selected or filtered *a priori*. In these cases, a user interface (UI) is required to give the user the choice. This paper focuses on several user interface techniques that allow the user to restructure the data they capture. The following topics will be discussed:

1. Document understanding: top-down vs. bottom-up
2. Click and select motifs for capture
3. Statistical model for region classification and clustering
4. User-friendly user-interface (UFUI) features
5. Comparative analysis

Throughout this paper, we will argue that "document capture" is not a singular process, but rather a continuum between complete automation on one end and complete user control (UFUI control) on the other. As such, the following continuums are also of interest:

1. Manual control vs. automated quality assurance (QA)
2. High speed (performance, efficiency) vs. high accuracy
3. User interfacing vs. artificial intelligence
4. Region growing vs. scene analysis

5. Multiple tools vs. multiple algorithms
6. Image display vs. command-line automation

In each case, we explore how the UFUI philosophy favors the left-hand side of these continuums, particularly for speed and efficiency of information capture, classification, clustering and purposing. We also note how processes such as commercial scanning, book capture and print-on-demand publishing favor the right-hand side of these continuums. We shall make these comparisons throughout, showing in this paper the tools necessary to provide the user with efficient, "bottom-up" control of her capture experience.

2. Document Understanding: Top-Down Vs. Bottom-Up

Document understanding implies that (at least) the following data has been obtained for a document image: (a) zoning analysis, which includes segmentation (formation of separate regions) and classification (typing the separate regions; e.g. as "text", "photo", "table", etc.); (b) clustering analysis, in which like regions are combined into higher-level "derived" regions, such as columns of text, composites of backgrounds and overlying regions (termed "multiple Z-order regions" and discussed in the next section); (c) meta-data analysis, which includes optical character recognition (OCR), drawing recognition (e.g. conversion of line drawings to Scalable Vector Graphics, or SVG [1]), region representation in XML [2], and page format description (this can be, for example, the PDF, HTML, or other end format); and (d) if the document is within a larger corpus, it has been appropriately tagged, sequenced, clustered, assigned, archived, etc. We focus here primarily upon the creation aspect of document understanding; that is, upon (a) and (b). A top-down approach considers "page-wide" or "edge-in" metrics to segment a document. An example of "page-wide" metrics is described in [4]: segmentation using "the ratio of edges to image region; and…the ratio of strong to weak edges. It has been determined that these two ratios operate so as to 'cluster' the segmented regions into desired categories for differential rendering, such as, for example, clusters corresponding to the text, graphics, and picture regions." Wahl, Wong and Casey's landmark zoning engine [5] uses "page-wide" rules; that is, thresholding, run-length smearing and other decisions are made at the page-wide level. Another type of top-down zoning analysis is that based on cuts [6], which use spacing between regions to assist in segmentation. A combination of these page-wide population statistics, region-classification heuristics and use of cuts and projection profiles [7] was used to perform the top-down zoning analysis of the entire document in Figure 1.

A "bottom-up" approach (Figure 2), on the other hand, begins at one point in the image, uses local statistics, and applies different rules, procedures, etc. based on the type of region its statistics indicate it to be finding. This click and select approach is "bottom-up" because it starts at a single point, and more area is added to the region dynamically, based on that initial point.

While this is more efficient for a single region (and for continually providing the user with feedback in the UI window), it should be noted that, in general, click and select will actually take substantially longer overall to generate regions than will a top-down approach. The reasons for this are straightforward: (1) certain information (thresholds, projection profile, regions, etc.) will be duplicated from one click to the next (especially when the regions are non-rectangular), and (2) information by definition will be calculated outside the boundaries of each clicked region (to know where it ends), causing further duplication. In fact, for the Figures shown, the entire page was analyzed in 0.63 sec using the top-down approach (mean for 200 documents), while 8 successive clicks were required to select all of the regions, taking 1.33 sec of processing along with the time for the user to view and click. This may, however, seem faster to the user due to the nearly instantaneous formation of each region and the responsiveness of the UI. (Performance was evaluated using a Pentium P-III, 1.13 GHz, 256MB RAM, Windows 2000 system)

In addition to overall (and apparent) timing differences, top-down and bottom-up approaches will have differences related to how the analysis is performed. For example, in many top-down approaches, there is smearing to help form regions. In the bottom-up approach, since only one region is formed at a time, no smearing is required; instead, regions can be formed using boundary expansion.

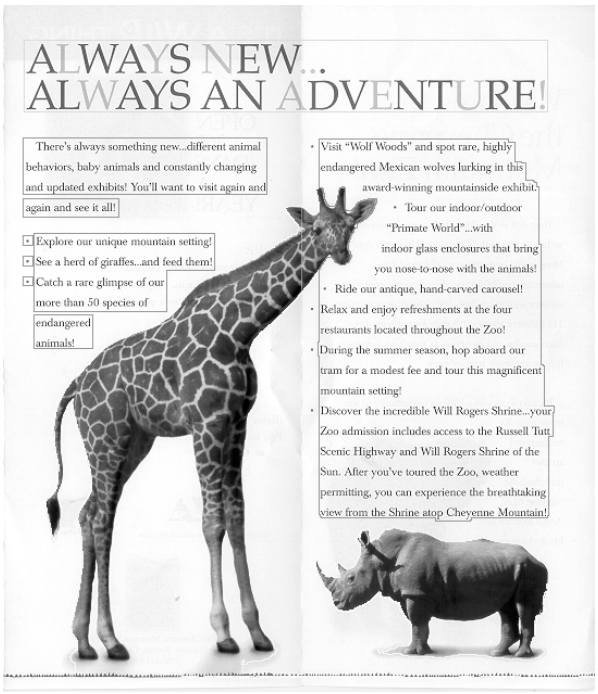

Figure 1. Document [3] regions formed using top-down document understanding (there is no "clicked" point—the entire page is analyzed at one time).

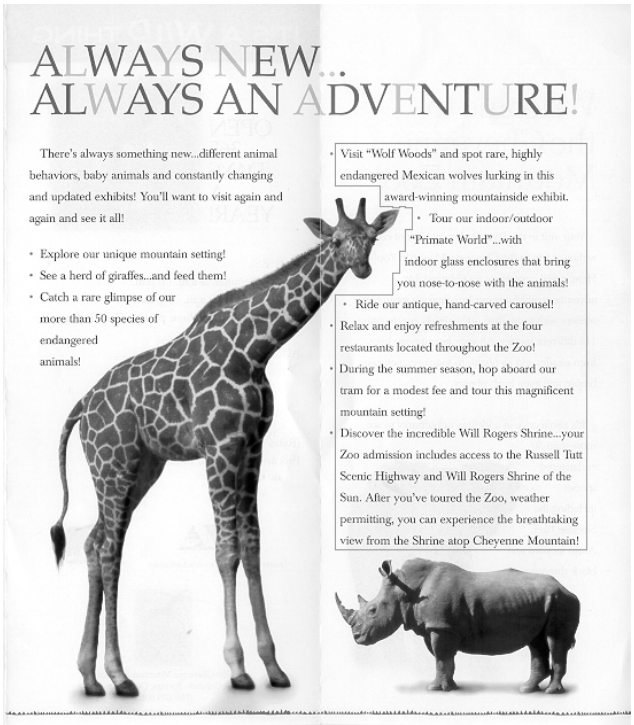

Figure 2. Document [3] region formed using bottom-up document understanding ("clicked" point was in the middle of the text region outlined).

In comparing the two figures (Figure 1 and 2), it can be seen that the top-down approach (Figure 1) has the right-hand side text column "grow" right up to the borders of the giraffe. In the bottom-up approach

(Figure 2), because subsequent lines of text are added incrementally to the region (this can be seen in the "stairstep" nature of the region's boundary to the right of the giraffe's head), no such adjoining of the text and "giraffe" region occur. As can also be noted from the comparison, however, the two regions contain the same text information: the only difference between the two regions is in the whitespace bordering the text.

3. Click and Select Motifs for Scanning

The "bottom-up" approach described above provides the means for the "click and select" capability for region capture [8]. Click and select means that the user activates a zoning analysis engine directly by clicking the mouse. Click and select has several manifestations, including: (a) single-region formation (Figure 2); (b) multiple-region formation (sequential clicks without destroying the previously-clicked regions) (Figure 3); (c) polygon-based region formation (large text region in Figure 3); (d) "enforced-rectangular" regions (Figure 4); (e) recursive click and select, in which case click and select can be applied to an already existing region (e.g. to click on text above a photograph); and (f) "pre-analyzed" click and select, in which the top-down analysis has already occurred, but no regions are presented to the user until she clicks on points in the UI (this allows the user to have all the regions "at the ready", but still be able to control the regions selected).

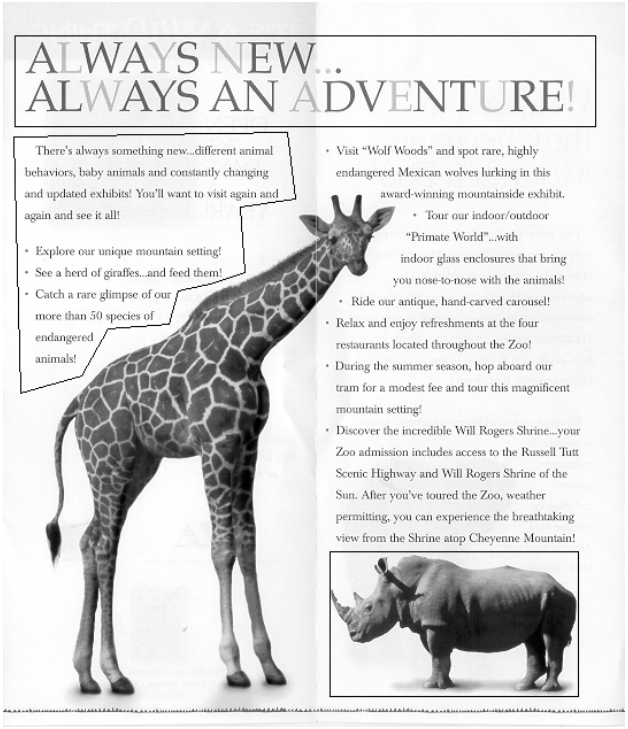

Figure 3. Document [3] regions formed using three consecutive "click and selected" points: two "grown" regions were rectangularizable, the other was not and so required polygonal bounds (shown here) or overlap with one or more other regions (not shown) for its definition.

The fundamentals underlying click and select are the following:

(1) Thresholding. The raster image is first thresholded to binarize the image, so that background (usually white) is represented with a "0" in the thresholded map, and all other (non-background) pixels are represented with a "1" in the thresholded map. Many different thresholding algorithms have been tested for this step [9-13]; we have generally used a modified form of [9] for implementation, subject to the constraint

that the primary background is determined beforehand. If the primary background is not on one side or the other of histogram, then it is instead subtracted out (made "0"'s in the thresholded map) from the surrounding non-background pixels. In this way, we accommodate non-white predominant backgrounds.

(2) Next, the area (usually 0.5 square inches to start, with sequential doubling of this area as the region around the pixel is grown) around the clicked pixel is analyzed for, among other parameters, its percent thresholded map "1" pixels. Text and line drawings will tend to have low (<50%) values for this parameter, while photos and color drawings will tend to have high (>80%) values for this parameter.

(3) If the parameters measured match the profile for a "thick" region, then the region boundaries are obtained. If the region is nearly rectangular or "rectangular" click and select is desired, then the smallest rectangular outline of the region is saved as the region (it can be corrected for skew by measuring the skew angles of its thresholded map boundaries).

(4) Otherwise, the region is analyzed for its skew (using standard Hough transform techniques, for example), if any, and projection profiles are accumulated to see if it is text or a line drawing. If the former, additional projection profiles are obtained to see how many evenly-space lines can be clustered into a text column. Otherwise, run-length smearing is used to form a single region for the line drawing.

Because it is a "bottom-up" algorithm, click and select is fast, efficient and generally reliable. It is, however, sensitive to the thresholding value, since low thresholds can lead to "breaking up" of large text regions (due to whitespace "cuts" extending inadvertently throughout the entire column). Work-arounds to this include (a) using multiple thresholds and selecting the smallest of those determined, (b) forming multiple "clicked" regions and OR'ing them together, and (c) aggressively narrowing the background range in the histogram (adjusting the threshold).

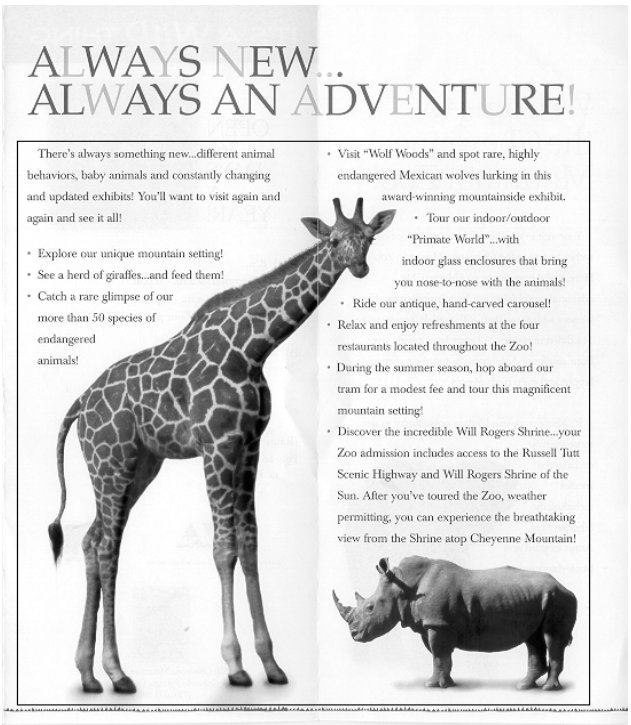

Figure 4. Document [3] region formed using "enforced-rectangular" region when clicking on the text left of the giraffe's neck. Since the smallest rectangle drawn completely in whitespace must enclose both the giraffe and the text due to geometry, the rectangle ends up "expanding" to that shown. This method ensures a rectangular file is captured, but does not ensure a single region type has been captured.

4. Statistical Model for Region Classification and Clustering

Next, we consider how regions can be represented statistically so as to enable efficient filtering of the content to be obtained from the UI [14]. The example we show is for text, although it can readily be extended to any other region type, based on the statistics in Equations 1-2 below.

Click and select is possible due to a bottom-up model for the zoning analysis. Additionally, a statistical model for all regions is used to allow dynamic updating of the region classification as well as its segmentation. For example, suppose one has clicked on a table. It is possible that one has clicked on the text in a given row and/or column. Thus, initially the region will be classified as "text." Alternatively, one may have clicked on a line in the table, so that the region will be originally classified as "line drawing".

As the region grows, however, it should be clear that it is a table due to its row/column structure and ordered text and lines. How can this be accomplished? Our approach has been to implement a classification set spanning statistical approach to region typing (classification). If the set of all region classification types is C, where "text" = C_1, "drawing" = C_2, "photo" = C_3, "table = "C_4", etc., then for all $C_1 \ldots C_N$ (N=number of region types possible), each region $R_j \varepsilon \Re$ where $\Re$ is the set of all M regions formed during segmentation, is assigned probabilities $p(C_i)$, such that:

$$\forall R_j \varepsilon \Re_{j=0\ldots M}, \Sigma_{i=0\ldots N} p_j(C_i) = 1.0 \quad \textbf{Eq. 1}$$

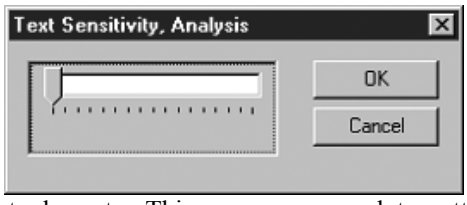

Figure 5. Text sensitivity set to lowest. This may correspond to setting only those regions with a differential probability of more than 0.75 for (Text | all other region types) to text in the UI (Figure 6).

That is, the given region R_j has a summed probability of 1.0, which represents its relative probabilities over all region types. In the following example, we illustrate the use of these statistics where N=4, the C_i are defined as above, and M=13 (counting backgrounds, rules, etc.). Starting with Figure 6, the text column in the upper right may have the following statistics: $p_1(C_1) = 0.90$, $p_1(C_2) = 0.00$, $p_1(C_3) = 0.00$ and $p_1(C_4) = 0.10$. Thus, region 1 can be viewed as having a differential statistic for text of $p_1(C_1) - p_1(C_4) = 0.80$. In general, the differential statistic, DS, for region x with respect to classification y is given by Equation 2:

$$DS(x|y) = p_x(C_y) - \max(p_x(C_i))_{i \neq y} \quad \textbf{Eq. 2}$$

Thus, in Figure 6, the regions {x}outlined all have a DS(x|1) >= 0.75, since that is the Text Sensitivity setting in Figure 5.

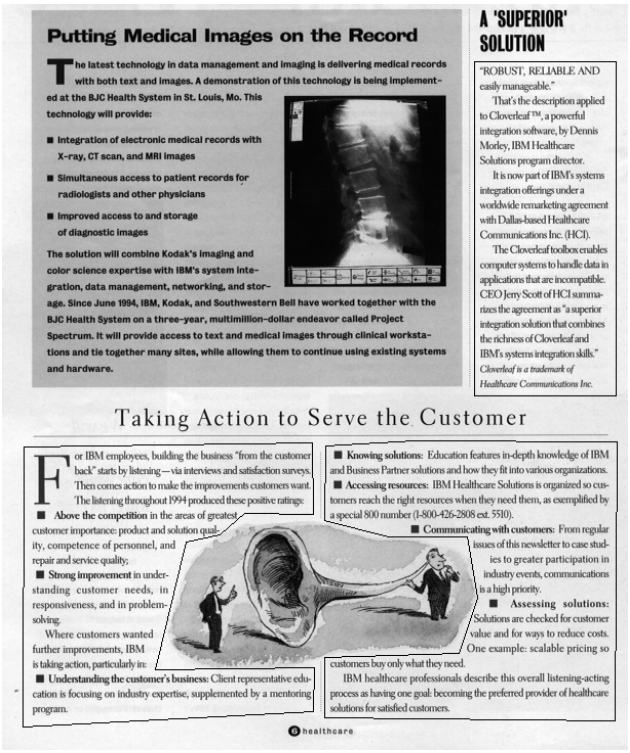

Figure 6. Regions defined as text with text sensitivity set to its minimum (Figure 5). For this document [15], there are three such text regions, corresponding to text columns that are over white backgrounds.

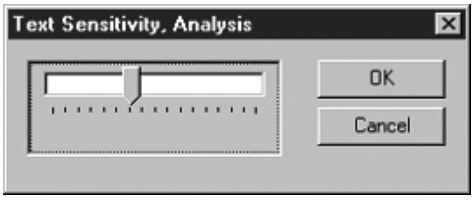

Figure 7. Text sensitivity set to 0.35 (range from 0.75 at far left, -0.25 at far right). This now corresponds to setting only those regions with a differential probability of more than 0.35 for (Text | all other region types) to text in the UI (Figure 8), which now includes the large text region over a shaded background.

Next (Figure 7), we move the "Text Sensitivity" slider to the right, changing its value from 0.75 to 0.35 and increasing the sensitivity of the UI to text as follows:

1. Capturing the setting from the "Text Sensitivity" UI tool. Supposing the "Text Sensitivity, Analysis" tool's range is from 0.75 to –0.75, then in Figure 7 this setting is now 0.35.

2. Compare the value of DS(x|1) for all regions x. If DS(x|1) >= 0.35 for any region x, then this region is now selected in the UI as shown in Figure 8.

We continue this process in Figures 9-10. For Figure 9, we have set "Text Sensitivity" to 0.0, which now means any regions for which DS(x|1) >= 0.0 are typed as text. These are all of the regions that would default to text at the end of zoning analysis. Note that in this particular application (Hewlett-Packard scanning software that shipped from 1996-1999) we defaulted larger font size, single line text regions to "drawing" region type, so that a raster-to-vector (i.e. SVG [1]) conversion of this text was generated (the OCR engines did not, in general, reproduce such fonts well).

In Figure 9, regions that will otherwise default to "text" in the UI are outlined. As a final example, the "Text Sensitivity" slider is moved all the way to the right of its range; that is, to –0.25. This means that, in effect, any region for which the text probability, $p_x(C_1)$, is within 0.25 of any other region classification probability will now be outlined in the UI (and thus classified dynamically as text).

The dynamic classification of regions has several advantages. First, the user can select only large blocks of text from the image, suitable for "cutting and pasting" into other documents. This activity can be facilitated by "deleting" the previously clicked regions after they have been sent to their destination (e.g. a text-based file).

Additionally, the user can define the page for how it is to be repurposed. If the user wants the OCR engine to perform OCR on all of the text (since text-based searching may be important to his end-application), for example, he may choose the highest sensitivity (Figure 10). Third, the user can use the same statistical techniques described above for region clustering ("click and cluster").

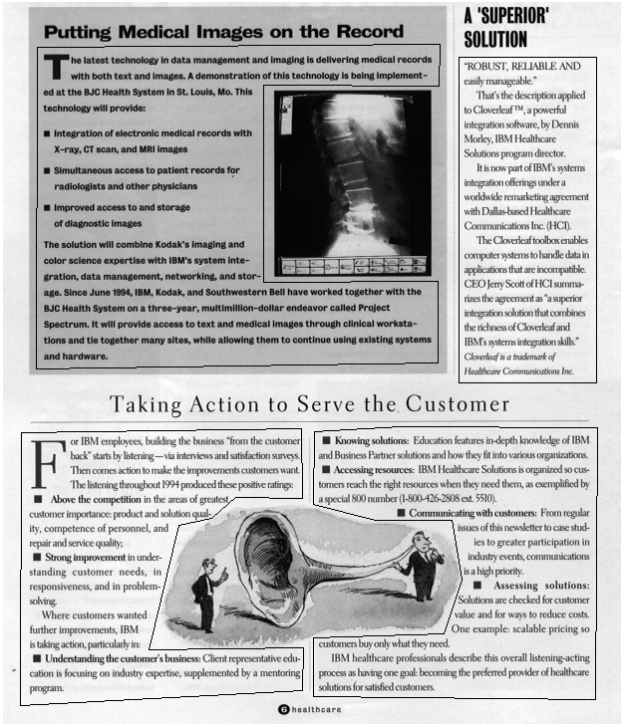

Figure 8. Regions defined as text with text sensitivity set to 0.35 (Figure 7). For this document [15], there are four such text regions, corresponding to all text regions with 3 or more lines of text.

Region clustering consists of two separate types of region combination. The first, simple clustering, involves combining two or more regions with the same current classification. This classification is termed "like classification" since regions of the same class are clustered (Figure 11). The second, "shared background clustering", clusters regions that share a common background (Figure 11). The third, "vicinity clustering", simply groups regions based on their (relative and/or absolute) proximity to one another (Figure 11). Various schema for switching between these clustering types can be used. For our instantiation, we made the composite region consist of the region clicked as well as the best set (1 or more) of candidate regions to create a larger, unified region. The following choices were prioritized: (A) like clustering (e.g. paragraphs created from clicking on single lines of text, or a single large photo created from a set of nearly adjoining photos), (B) compound region clustering (the creation of compound region types, such as business graphics, tables, cartoons, equations, etc., from their component regions types such as text, rules, line drawings, etc.), (C) shared background clustering, and (D) vicinity clustering.

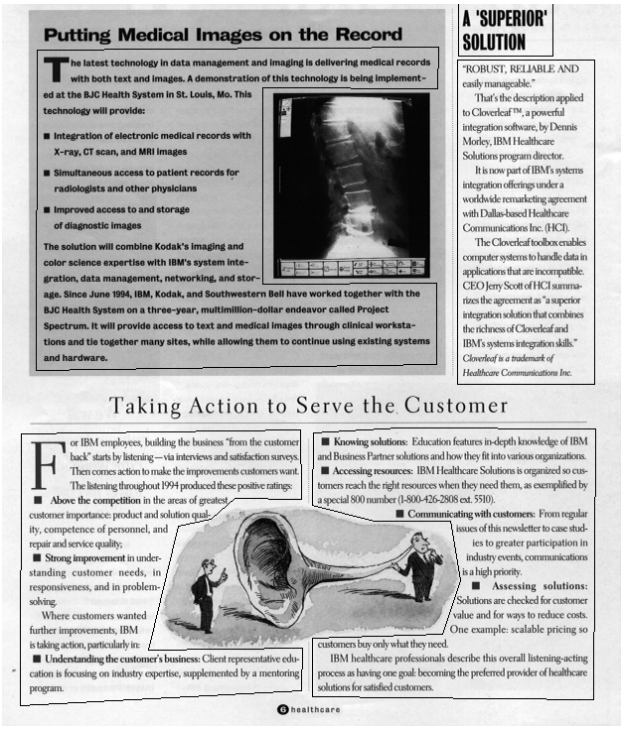

Figure 9. Regions defined as text with text sensitivity set to 0.0 (3/4 of the way right on Figure 7). For this document [15], there are five such text regions, corresponding to all text regions with 2 or more lines of text.

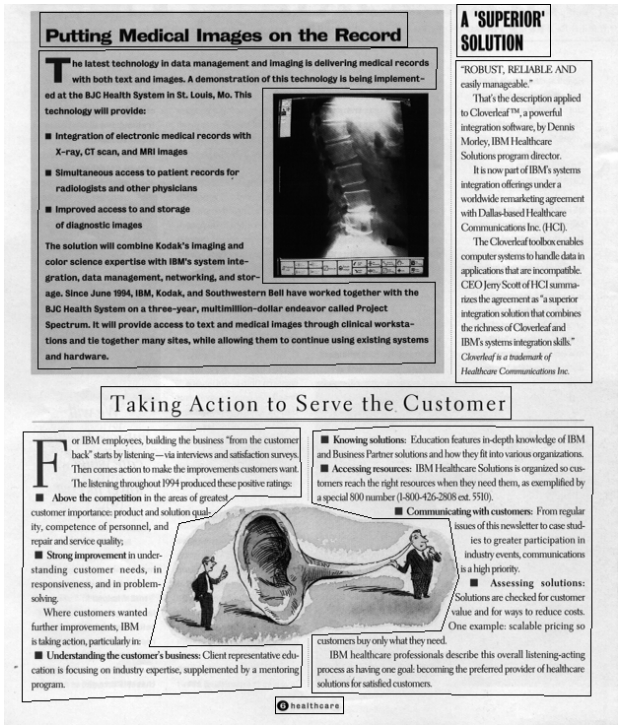

Figure 10. Regions defined as text in the UI with text sensitivity set to -0.25 (all the way right on Figure 7). For this document [15], there are eight such text regions, which encapsulate all real text in the document, including some line art and text in fonts that will not be well-reproduced by some OCR engines.

Aside from the potential benefits in clustering, such a click-initiated clustering has another benefit: that of simplifying the zoning analysis regions presented in the UI window. This is because each successive click is guaranteed to cluster two or more regions into one region, thus continually simplifying the layout of the regions shown.

5. UFUI (User-Friendly User Interface) Features

The click and select features for region filtering, classification and clustering described in the previous section can be viewed as a UFUI feature. A UI is user-friendly only if it provides useful functionality with convenience. Complicated menu dialogs and sub-dialogs are, in general, less friendly than mouse-initiated clicks, hotkeys or "hot buttons" (such as those shown in Figures 13-15). One simple, user friendly motif is the use of a highlighting tool for text capture and text emphasis [16]. This tool (Figure 12) allows the user to draw a highlight (usually in yellow or pink "fluorescent" color) over the selected text that they wish to convert (with OCR) or otherwise use.

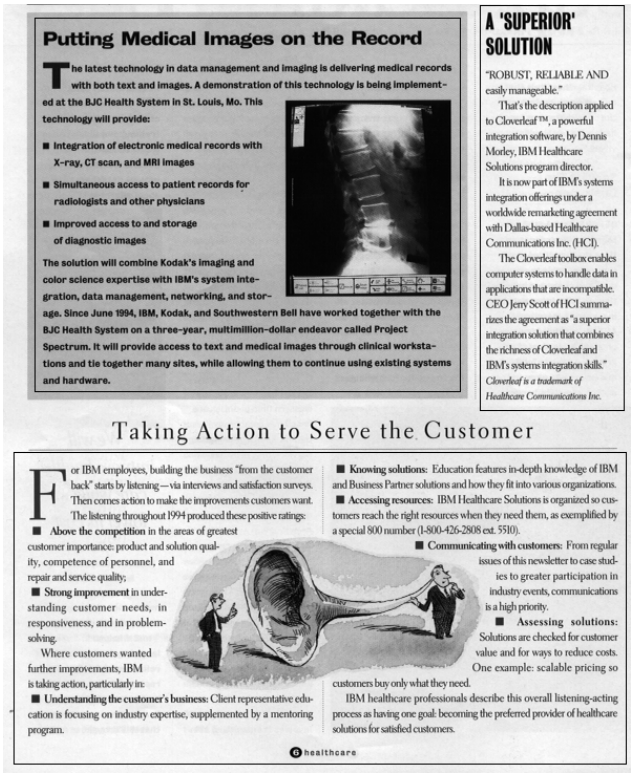

Figure 11. Statistically-based clustering ("click and cluster"). For this document [15], there are several types possible, including like classification (upper right), like background (upper left) and vicinity (lower), as the three outlines evince.

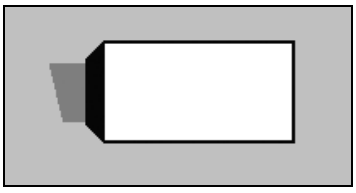

Figure 12. The icon for the "highlighter" UI tool. Running the pen over text "highlights" the text with 100% saturated yellow (i.e. sRGB = {255,255,0}) color.

Other simple UFUI tools include cropping (rubberbanding) tools, fill tools, etc., most of which are available in image editor software. They should be used sparingly in scanning/capture software, and only when they speed the process between scanning and reaching the end application/destination (e.g. photo editor, word processor, printer). It is worth noting that the cropping and filling tools can benefit from underlying zoning analysis: once the user has zoned the document, he can just select the crop/fill tool and start cropping/filling the then-visible boundaries of the clicked region.

6. Comparative Analysis

The "click and select" algorithm and user interface tool [8] has shipped for several years as part of the scanner, multifunctional printer (MFP) and all-in-one device software at the authors' company. The earliest incarnation was in the Precision Scan Pro software, as shown in Figures 13-15.

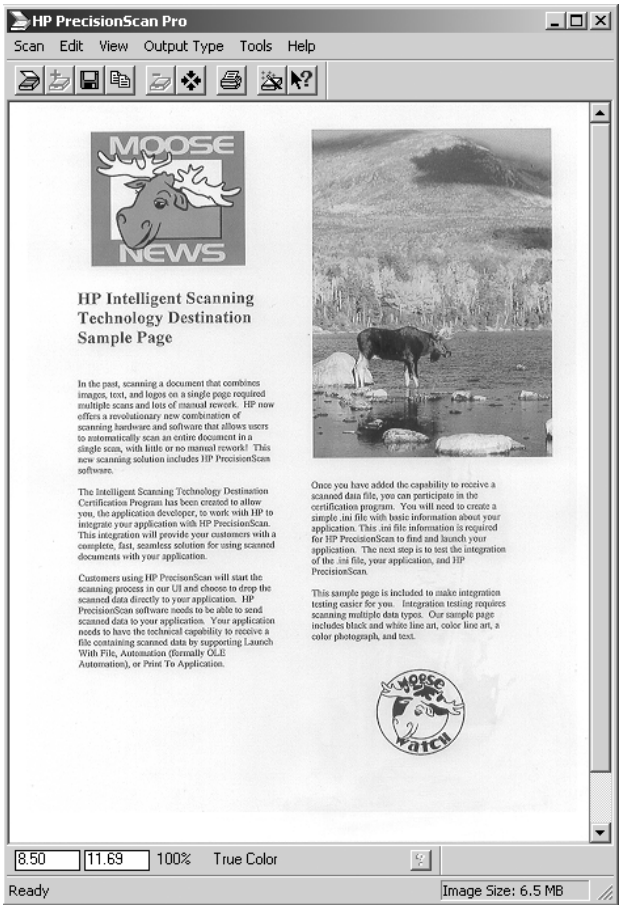

Figure 13. The PrecisionScan Pro user interface, with its simple menu and "hot button" configuration, provides click and select using the (left) mouse button. The simple UI is deceptively powerful.

In Figure 13, the PrecisionScan Pro UI is shown. Menus for Scan, Edit, View, Output Type, Tools, and Help are clearly visible, along with "hot buttons" for scanning, saving and printing. In Figure 13, the preview scan is shown. This is usually presented at screen resolution (that is, at 72 pixels/inch, or "ppi"), which is adequate for click and select analysis. Click and select is accurate between 30-100 ppi, and operating at this low ppi provides nearly-instantaneous performance, low memory overhead, and rapid preview. Clicking (left mouse button) on the "Moose News" drawing in the upper left of the preview

window sets the click and select segmentation+classification into motion, and within a few tenths of a second, the selected region is indicated (Figure 14). Rectangular regions are used in the product that ships to consumers, since most of their output files (photos, text, etc.) are preferably rectangular. The region is also automatically typed as a color drawing, so that optimal bit depth (e.g. 24) and resolution (e.g. 200 ppi) are selected for the output (destination). If the user accepts this region, the selected box—and not the whole scanbed—is rescanned at the appropriate bit depth and resolution and shipped to its fulfillment destination.

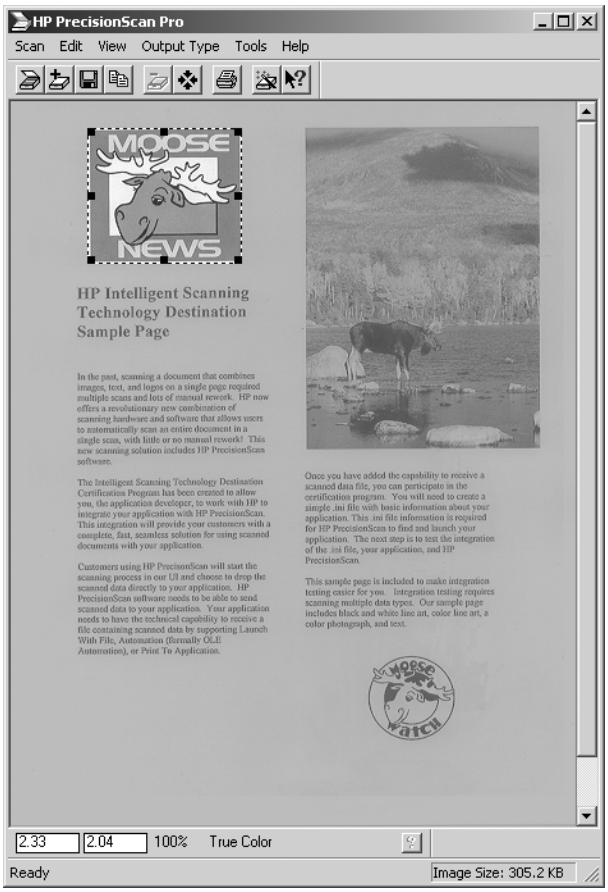

Figure 14. The PrecisionScan Pro user interface, after the "Moose News" drawing has been clicked on. On the display screen, the region has also been typed as "color drawing" as part of the click-instantiated region find.

In Figure 15, the user has instead clicked over a text region in the central right-hand side of the image. The region has been typed as text (1-bit, 300 ppi settings automatically chosen), and this will govern the settings of the final scan.

Advantages of a preview window and click and select include a low resolution preview scan that lets the user see the scanned document/image quickly. Click and select allows the user to decide what parts of the scanbed she wishes to actually save, print, insert into another file, etc. The preview + click and select motif is thus advantageous in the following situations:

1. The user wishes to only capture part of the scanned document/image (filtering).
2. The user wishes to control the region classification, segmentation, clustering or order (re-purposing).
3. The user wants a specific region structure (layout editing).

4. The user is scanning multiple images and wants a continual reminder of which document is currently on the scanner (automatic document feeder mode).

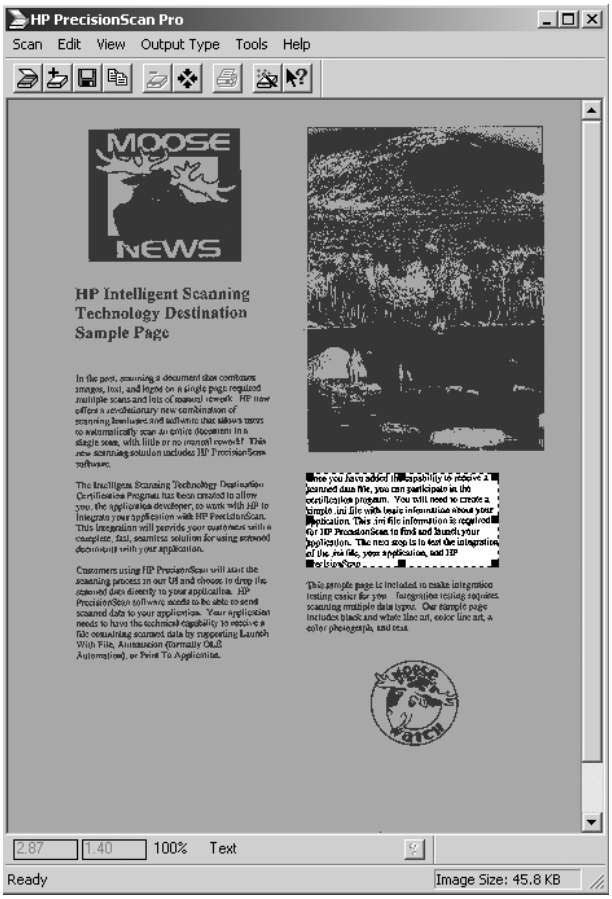

Figure 15. The PrecisionScan Pro user interface, with its simple menu and "hot button" configuration, provides click and select using the (left) mouse button. The simple UI is deceptively powerful.

On the other hand, click and select is not always preferable to "top-down" or "whole-document" zoning analysis. Instances where the latter is preferred include:

1. Large document processing jobs, in which only minimal time is available for correcting each page.
2. Automatic modes, wherein user intervention is either not possible (remote application) or not preferred (due to costs, etc.).
3. Most of the document is to be used in the end-application or destination. In these cases, it is more efficient to show the user the default regions and let him delete or cluster the unwanted regions.
4. During UI "idle times", full document analysis can be performed in the background so that any potential future selection consists of selecting already formed (but hidden to the preview window) regions.

7. From UFUI to Full Automation

Here, we explore how the UFUI philosophy favors the left-hand side of several continuums, particularly for speed and efficiency of information capture, classification, clustering and purposing. We also note how processes such as out-of-print book capture, commercial scanning and print-on-demand publishing favor the right-hand side of these continuums.

Manual control vs. automated quality assurance (QA)
As mentioned above, the UFUI tools described herein ("click and select", "click and cluster", "highlight tool", etc.) require the user to operate. Thus, they are manual. Click and select can be fully manual (regions are formed after the click) or semi-manual (regions already formed are shown after the click). Manual control of QA is 100% accurate, assuming the user understands the issues behind QA. This means the user is responsible for understanding bit-depth, resolution, sharpening, auto-exposure and a host of more arcane imaging concerns. This is contrasted to automated QA used for large-scale projects such as the paper-to-PDF conversion of the entire out-of-press MIT Press book collection [17]. In the automated case, the user need understand nothing about imaging, but instead need simply supply files to the system and archive the output files (PDF's, in this case).

High speed (performance, efficiency) vs. high accuracy
UFUI techniques are typically a few times faster than full page zoning analysis techniques. On a per-area basis, however, they are slower, for reasons discussed above ("information duplication" and computation on areas outside of the borders, or "processing overlap"). However, there is often a subtle difference in how the user perceives time. While waiting for a preview or for the automated set of regions to appear, the user is often impatient. After the preview has shown up and the user has clicked, however, she may not notice at all the processing time, even when it is comparable to that of the top-down, whole-page processing. Typically, the accuracy (e.g. defined as $2pr/[p+r]$ where p=precision and r=recall) is comparable for top-down and bottom-up (UFUI) algorithms, so that the UFUI offers advantages in correcting mistakes, since the user is already interacting with the UI window.

User interfacing vs. artificial intelligence
Manual techniques, such as UFUI techniques, require user familiarity with the application's UI (preview/editing window). This is a distinct requirement from the imaging knowledge the user must have in Section 7.1. The user must know how to use menu, hotkey, "hot button" and mouse commands effectively, how to correct or undo mistakes, and how to save their results. User familiarity enables a repertoire of applications ("re-purposing") from a single window, if the UI is properly designed and connected. Automated systems, on the other hand, require some level of "artificial intelligence", from the most rudimentary (static pipelines) to the most advanced (agent-based, adaptive and dynamic workflows that account for bandwidth, processing and memory availability). Building automated systems to perform all of the same tasks as users can control from a menu is no easy task, especially since well-architected UI's allow the user some measures of creativity.

Region growing vs. scene analysis
With UFUI tools, the user can click and select regions, click and cluster them, and otherwise control the nature of their region set. Usually, this means starting with a blank set of regions and "growing" the set they want to use in their end-application or destination. Essentially, this is a filtering process: the user does not save everything that is scanned/captured, but only the material that is important to her. This contrasts with automated methods of capture (including auto-cropping), which capture what the software thinks is important. Presumably, the user knows with 100% certainty what is important to her. The software cannot. For example, in Figure 8, the user has captured a large amount of text over a non-white background. The user otherwise have wished to collect the entire background with the text and the X-ray image over it. Either of these two is a "legitimate" requirement for the user: the former perhaps for text archiving, the latter for saving a digitized patient record. How is the software to store the region…with text settings? photo settings? both?

On the other hand, user-directed capture is clunky when all that is required is scene analysis. Most descriptions of scenes can be automated with image segmentation algorithms that the user is unlikely to understand or be able to reproduce with manual entry of meta-data. Here, UFUI tools are unlikely to benefit the user.

Multiple tools vs. multiple algorithms
With a UFUI, multiple tools are available to the user to filter, edit and re-purpose his content. The number of tools should be limited to those essential for ensuring usable downstream data (image quality) and proper capture (image quantity). Since users of UI are used to quick responses to menu, key, button,

mouse, etc. commands, it is important that the UFUI algorithms are fast and relatively accurate (highly inaccurate UI tools lead to immense wasting of the user's time). Automated processes, however, can be run off-line, which allows for combinations of algorithms to be used to improve accuracy (usually at the expense of performance). For example, multiple OCR engines can be used, together with a voting scheme, to improve the accuracy of text output after raster analysis.

Image display vs. command-line automation
Obviously, UFUI tools benefit from the image, or preview, display. Command-line automation, however, is possible for automated tools. This improves scheduling possibilities (i.e. the processing can be scheduled to occur while the user is away from the office), and also increases the ease of distributed computing appropriate to truly large-scale projects.

8. Conclusions

We have overviewed several tools in the UFUI toolbox. The UFUI approach is best suited to situations in which the scanning/capture device is a conduit and filter for information, and the user has further ("downstream", or destination) applications that will process/use the information captured. We focused on one extreme of the continuum, the rapid capture and purposing of the scanned information extreme. On the other end of the extreme is a fully-automated system, such as for converting a large corpus of documents from raster to PDF format [17], with the only possible user intervention being after-the-fact use of a UI-based correction application [18] for any documents not meeting quality assurance levels. Between these two extremes lie most of the popular image-based applications: image editors, document management systems, optical character recognition systems, copiers, and others.

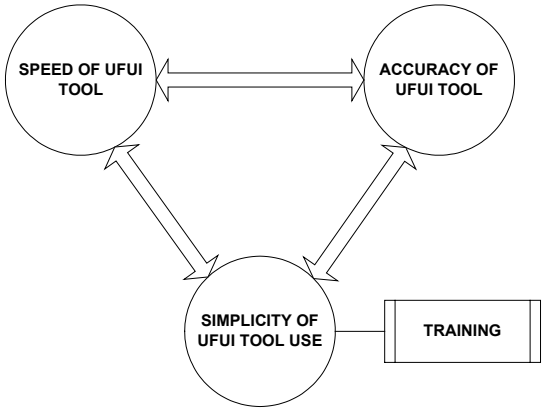

Figure 16. Trade-offs involved in designing user-friendly user interface (UFUI) tools for user-directed analysis of scanned images.

Figure 16 summarizes many of the key trade-offs to consider in designing UFUI tools for capturing scanned data. In general, the UFUI tools should be efficient/fast, accurate and simple to use (or to train on in using). The faster an algorithm is, however, the generally less accurate it is. This is a difficult trade-off to override. However, the other two trade-offs can be addressed, and in doing so we gain some insight into addressing the efficiency-accuracy trade-off. (1) Simplicity of UFUI tool use can be obtained, for example, by relegating the least-commonly used commands to the least-obvious UI buttons, menu locations, hotkeys, etc. This "scales up" the complexity of UI use to those UI tools least likely to be used, minimizing the mean time spent on each task. An analogous approach can be taken with respect to efficiency vs. accuracy. That is, the default processing invoked by the tool assumes a simple data model, and this is overridden only if the data proves more complex. An example is using "click and select" to generate polygonal regions. A fast method for doing this is to initially assume that the polygon is actually a rectangle, and then add more vertices only as the non-rectangular nature of a region asserts itself. (2) Accuracy of UFUI tool can be

addressed in many ways, but one particularly successful approach is to use two or more independent analysis methods and combine (via voting or more complex "meta-algorithmics", or combinatory algorithmics) what they output to get a more accurate result. Since this usually causes a deleterious effect on speed/efficiency, instead we apply this principle as follows: use a fast first algorithm and allow the user obvious, efficient editing tools for cases in which this simple-but-fast algorithm fails. An example is illustrated in Figures 14-15, wherein the clicked region can be easily edited by "dragging" any of the four borders of the rectangle shown in the UI window.

With these strategies, we have shown that it is possible to provide UI tools that are simultaneous efficient, accurate and simple. These tools help make the capture of scanned data more accurate, encouraging the continual enrichment of the digital world with ready-to-use "analog" data.

The methods we describe here provide input to meta-data analysis, and can be used as part of larger QA/publishing systems [17-18]. An important aspect of the UI tools described here is in shortening the time it takes to "tag" the input data, so that the user can spend more time on fulfillment (photo editing, printing, archiving, etc.) than on the mechanics of capture. Also, intuitively simple UI tools make the novice-to-medium-skilled user feel intelligent. This encourages them to adopt more and more digital-only solutions for their document processes.

9. Acknowledgements

The authors thank HP colleagues (in particular, Jeff Lee and Patty Lopez) for their insights, support and collaboration in many related projects over the past nine years. Special thanks also go to Jay Veazey, Sheelagh Hudleston, John Burns, Julie Dawe, Archie Carrington, David Auter, Virgil Russon, Ted Neff, Rick Lesser, Mark Carleton, Jeremy Cook, Margaret Sturgill, Ray Smith, Phil Cheatle and Scott Baggs.

10. References

[1] SVG home page, http://www.w3.org/TR/SVG/.
[2] Simske, S. "The Use of XML and XML-Data to Provide Document Understanding at the Physical, Logical and Presentational Levels," In Proc. of the ICDAR99 Workshop on Document Layout Interpretation and its Applications, Sept. 1999.
[3] Cheyenne Mountain Zoo, http://www.cmzoo.org/.
[4] Revankar, S.V. and Fan, Z. "Image segmentation system", U.S. Patent 5,767,978, January 21, 1997.
[5] Wahl, F.M., Wong, K.Y. and Casey, R.G. "Block segmentation and text extraction in mixed/image documents," Computer Vision Graphics and Image Processing, Vol. 2, pp.375-390, 1982.
[6] Shi, J. and Malik, J. "Normalized cuts and image segmentation," IEEE Trans Pattern Analysis Machine Intelligence, Vol 22, no. 8, pp. 888-905, 2000.
[7] Zramdini, A. and Ingold, R. "Optical font recognition from projection profiles," Electronic Publishing, Vol 6, no. 3, pp. 249-260, Sept. 1993.
[8] Lee, J.P., Lopez, P.D., and Simske, S.J. "Click and select user interface for document scanning," U.S. Patent no. 6,151,426, Nov. 21, 2000.
[9] Kittler, J. and Illingworth, J. "Minimum error thresholding," Pattern Recognition, Vol 19, no. 1, pp. 41-47, 1986.
[10] Otsu, N. "A threshold selection method from gray level histograms," Pattern Recognition, Vol 9, no. 1, pp. 62-66, 1979.
[11] Kurita, T., Otsu, N. and Abdelmalek, N. "Maximum likelihood thresholding based on population mixture models," Pattern Recognition, Vol 25, no. 10, pp. 1231-1240, 1992.
[12] Kittler, J., Illingworth, J. and Fõglein, J. "Threshold selection based on a simple image statistic," Comp. Vision Graph. Image Proc., Vol 30, pp. 125-147, 1985.
[13] Wilkinson, M.H.F. "Optimizing edge detectors for robust automatic threshold selection: coping with edge curvature and noise," Graph. Models Image Proc., Vol 60, pp. 385-401, 1998.
[14] Simske, S.J. and Lee, J.P. "System and method for manipulating regions in a scanned image," U.S. Patent 6,263,122, Jul. 17, 2001.

[15] IBM Healthcare web page, http://www-1.ibm.com/ industries/healthcare/.
[16] Simske, S.J. and Lesser, R.R. "User interface high-lighter function to provide directed input for image processing," U.S. Patent 6,385, 351, May 7, 2002.
[17] http://mitpress.mit.edu/ main/feature/classics/MITPClassics_release.pdf, MIT Press Classics Web page.
[18] Sturgill, M. and Simske, S.J. "A proofing, templating and purposing engine in Java and C#/.NET," Hewlett-Packard Technical Report HPL-2002-272, 25 pp., 2002.