To Appear in UIST '03, Nov 2-5, Vancouver, Canada

Perceptually-Supported Image Editing of Text and Graphics

Eric Saund, David Fleet, Daniel Larner, James Mahoney
Palo Alto Research Center
3333 Coyote Hill Rd.
Palo Alto, CA
{saund,fleet,larner,jvmahon}@parc.com

ABSTRACT

This paper presents a novel image editing program emphasizing easy selection and manipulation of material found in informal, casual documents such as sketches, handwritten notes, whiteboard images, screen snapshots, and scanned documents. The program, called *ScanScribe*, offers four significant advances. First, it presents a new, intuitive model for maintaining image objects and groups, along with underlying logic for updating these in the course of an editing session. Second, ScanScribe takes advantage of newly developed image processing algorithms to separate foreground markings from a white or light background, and thus can automatically render the background transparent so that image material can be rearranged without occlusion by background pixels. Third, ScanScribe introduces new interface techniques for selecting image objects with a pointing device without resorting to a palette of tool modes. Fourth, ScanScribe presents a platform for exploiting image analysis and recognition methods to make perceptually significant structure readily available to the user. As a research prototype, ScanScribe has proven useful in the work of members of our laboratory, and has been released on a limited basis for user testing and evaluation.

KEYWORDS: ScanScribe, rough document, WYPIWYG, perceptual document editing, foreground/background, lattice grouping, bitmap image

INTRODUCTION

Computer editing tools can be characterized along the dimensions shown in Figure 1. The horizontal axis represents the type of imagery the editor is designed to handle on a dimension ranging from richly complex to highly constrained. Three major categories are photorealistic scenes, 2D graphics, and text. The vertical axis represents the degree of structure available to a computer program. This ranges from unstructured, when individual pixel intensities are represented without any coherent object identities relating them, to highly structured objects such as 3D CAD-type models, vector graphics, and ascii text.

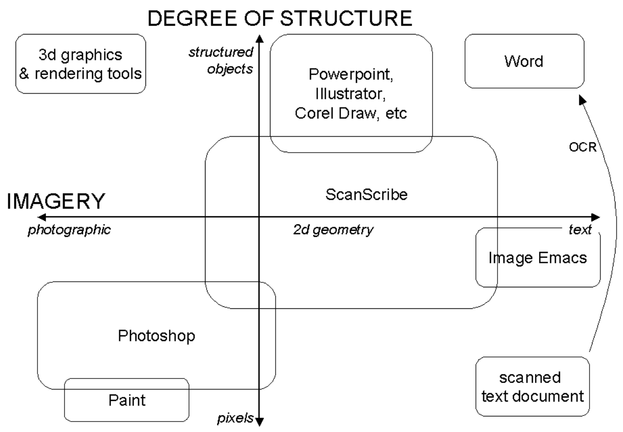

Figure 1: A two-dimensional conceptual view of computer editing tools.

The class of *structured* image editors includes ascii text editors such as Gnu Emacs, Microsoft Word, and Wordperfect, and structured vector graphics editors such as Microsoft Powerpoint, Corel Draw, and Adobe Illustrator. These tools allow the import of bitmap images as independent objects, and some are increasingly integrating tools for manipulating the content of these bitmap images.

The class of unstructured pixel-based editors includes "paint" style image creation programs such as Microsoft Paint and JASC Paint Shop Pro. Other programs are aimed toward editing images of existing bitmap images, such as Adobe Photoshop and GIMP. These are targeted largely to editing images of natural photographic imagery. As such, they provide a host of features and options for choosing colors, selecting image regions, for painting and filling, organizing collections of pixels on different layers, and applying image processing filters. A significant trend is to provide intelligent tools whose behavior is governed by the underlying image material, such as "smart" scissors that follow lightness or color boundaries, and "red-eye removal" tools that find the pupils of peoples' eyes and remove photographic flash artifacts. The number and complexity of features and options in sophisticated photographic image editing programs tends to render these programs difficult to learn and cumbersome to use. For targeted tasks and imagery of interest, we propose that a simpler and more accessible model is available.

We suggest that an unexplored region of the Structure/Imagery space lies between photographic bitmap image editors and structured document editors. This is a class of editors targeted toward what may be broadly construed as document images, that is, bitmap images derived from written textual or graphical material. This includes, but is not limitied to, scanned or rendered representations of formatted text.

One neglected class of document image can be referred to as *rough documents*. These are casual, informal documents such as handwritten notes, sketches, scribbles, doodles, annotations, diagrams, line art, graphics, and figures. Rough documents are often associated with creative, informal work processes. They are found on note pads, whiteboards, post-its, index cards, books, magazines, and on the backs of envelopes. Increasingly, means are becoming available for capturing rough documents not only via inexpensive flatbed scanners but through digital cameras, whiteboard scanners, tablet computers, and instrumented pens. Yet tools for editing rough documents—excerpting, cleaning, rearranging, and combining image material—are lacking.

Despite their diversity, images of rough and formal documents are substantially more constrained in their stylistic appearance and semantic content than general photographic imagery. By and large, documents consist of relatively dark markings, or foreground, set on a relatively light background (although the dark/light foreground/background relationship is sometimes reversed). Furthermore, by virtue of their purpose as conveyors of visual patterns for human communication, documents typically contain just a few kinds of markings, namely text, graphics, and photographs (although there are certainly exceptions and blurring between these classes). These constraints imply that editing tools can be designed to facilitate selection and manipulation of this kind of imagery specifically, and that such tools might serve some users' purposes more faithfully than general photographic image editors. Such is a principal motivation for ScanScribe.

One of the most critical properties of any editor is the facility it provides for *selecting* material. Once selected, standard manipulation operations include translating, rotating, scaling, duplicating, deleting, and copying to a clipboard. Vector graphics editors typically provide for selection by two means, by clicking the mouse on an image object, or by dragging a rectangle enclosure around one or more objects. Paint and photographic image editors provide a palette of enclosure-based selection tools, including typically rectangle drag, lasso, and polygon. Some programs also provide color based selection. Depending upon the pointing device and the user's dexterity, selection by dragging an encircling or rectangle can be marginally to substantially more cumbersome than simply pointing and clicking. Therefore we believe that greater facility is provided to the user to the extent that they can perform selection by direct means such as pointing and clicking.

Our long term goal is WYPIWYG image editing: "What You *Perceive* Is What You Get." The tool should maintain representations of the image objects and visual structures the user is likely to perceive as sensible chunks or objects to select, and make these readily available via the interface. This entails image analysis and recognition which can become arbitrarily complex. ScanScribe's architecture provides a framework in which recognition algorithms can be brought in incrementally, and are not required to work perfectly to provide useful results.

The paper is organized as follows. Section 2 reviews related work. Section 3 describes ScanScribe's design and novel features, including the maintenance of image objects, foreground/background separation, overloaded drag selection, and flat grouping model. Section 4 discusses our current and ongoing efforts in recognition of document image structure in casual line art and written text. Section 5 reviews our experiences using ScanScribe and deploying it to users. Section 6 concludes with directions for future work.

RELATED WORK

Over the past decade a number of groups have contributed to the notion of "smart" editors that reflect awareness, at some perceptual or semantic level, of the content of the image material being manipulated [1, 3, 6, 12, 13, 17]. Implementations have focused on online pen/stylus-based systems, largely because recognition of markings on a stroke-by-stroke basis is more tractable than recognizing text and graphics in static images. For example, the SATIN toolkit for building pen-based applications provides for modular interchange of digital ink stroke recognizers, whose output can then be routed as command gestures or content strokes [7]. But this system does not attempt to interpret gestures with respect to any underlying content image layer. Indeed, while accurate OCR of scanned, printed text has become commonplace, recognition of scanned engineering graphics remains on the edge of research, and recognition of rough documents, such as handwriting and sketched drawings, lies today beyond the capabilities of computer vision and document image analysis methods.

Commercially, stylus-based smart sketching tools are mainly limited to stroke-by-stroke conversion of digital ink input into vector graphic objects such as straight lines, circles, and rectangles. Research systems extend these capabilities to snapping objects to critical points and constrained configurations [5, 8] and to recognition of objects consisting of several strokes [1, 3]. Both research and commercial systems also display capabilities for parsing handwriting into a logical hierarchy of words, lines, and columns of text [18, 20]. In yet further work, logical relations among collections of digital ink strokes are maintained through recognition of characteristic devices of visual language such as encirclings, linear separators, and arrows [11].

As mentioned above, photographic image editors such as Adobe Photoshop are powerful but are overly complicated for many purposes and are not designed to facilitate editing of document image material in particular. Their notion of image layers provides an organizing principle for the critical function of grouping collections of pixels together and operating on them independently from other collections. ScanScribe's conceptual model of divisible image objects can be viewed as a variant of layers, but one that operates fully automatically and beneath the level of users' awareness.

Close in spirit to the present work is the Image Emacs system [2], which performed spatial analysis of scanned images of formatted printed documents in order to support structured editing of the bitmap itself. In this system, the bitmap is carved into independent image objects consisting of connected components of foreground (printed) characters, organized into groupings representing words, lines and columns reflecting the document layout. Editing operations consists of selecting and then manipulating the spatial positions of these objects.

SCANSCRIBE DESIGN AND FUNCTION

As a package, the ScanScribe document image editor combines new user interface techniques, a novel arrangement of extant methods, and newly developed image processing algorithms, amounting to a uniquely distinct user experience and a framework for enhancing this experience as document image analysis techniques improve over time.

Selecting Image Material

At the heart of ScanScribe lies the ability for the user to easily select whatever image material they intend. Once selected, the standard operations of dragging, cutting, duplicating, scaling, rotating, as well as others, all apply. In ScanScribe, users may select image material by any of four methods:

- Drag a rectangle.
- Drag a lasso.
- Drop the vertices of an encircling polygon.
- Click on an object.

The first three of these are always available. The last, clicking on an object, becomes available any time image objects have become established within the program. If the user selects a collection of pixels by rectangle drag, for example, and then moves it, these pixels automatically become an independent image object that can subsequently be selected simply by clicking on it.

The standard method for providing the user with an array of selection functions is through a tool palette; clicking a tool icon puts the system in that tool's mode, often reflected by the cursor shape. For instance, one tool would put the system in Rectangle Drag mode, while another puts the system in Lasso mode.

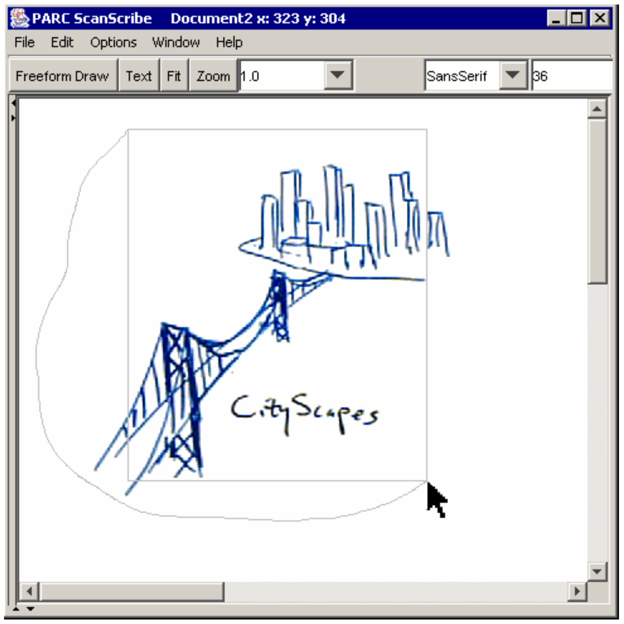
PARC ScanScribe Document2 x: 323 y: 304
File Edit Options Window Help
Freeform Draw Text Fit Zoom 1.0 SansSerif 36
CityScapes

Figure 2: Overloaded lasso/rectangle mouse drag selection technique obviates the need to choose between these selection modes with a prior toolbar selection.

To facilitate fluid interaction with image material, ScanScribe aims to minimize the necessity for prior mode setting and tool palettes. Accordingly, we have found power in an alternative novel *overloaded mouse drag selection* technique. When the left mouse button is pressed on a defined image object, that object becomes selected and is so indicated by a highlight halo. But when mouse-left is pressed over freespace (background), it simultaneously initiates a lasso selection path and a rectangle selection path. As the mouse is dragged, these are both displayed in an unsaturated color (light grey), making it easy for the user to visually focus on the either the rectangle or the lasso and ignore the other. See Figure 2. If the selection path proceeds to substantially close on itself, then the rectangle disappears. If the user releases the mouse button while the selection rectangle is visible, then the material enclosed by the rectangle is selected. If on the other hand the user releases the mouse button when the path has closed enough that the rectangle has been discarded, then the image material selected is that enclosed by the lasso. In other words, the program infers by the user's selection path whether they are attempting a rectangle selection or a lasso selection.

A fourth, *polygon* selection option is also available, again without resorting to a tool palette. This is invoked by double left clicking the mouse. Polygon selection is useful when the user would like to be able to visually adjust the boundaries of a selection region as they go. Vertices are dropped by left mouse clicks, and a left double click completes the polygon selection gesture.

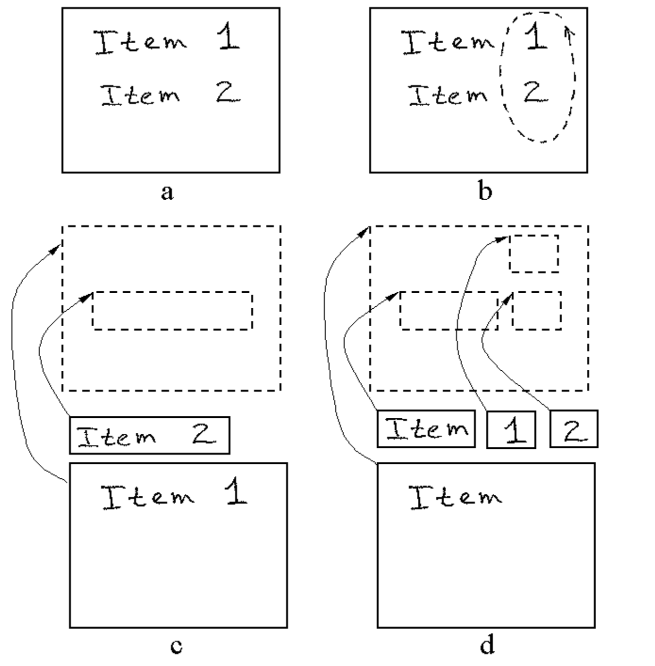

Figure 3: Representing a scene in terms of an arrangement of bitmap image objects. a. A graphic image whose underlying representation consists of two image objects positioned as shown in c. When the user performs a selection gesture, b, the image objects are carved into smaller pieces, d. This figure is an abstraction; issues of depth ordering, transparency, and the dimensions and effective shapes of the bitmap objects comprising a rendered scene are addressed later in the text.

Image Objects

At the start of an editing session, an image loaded onto the ScanScribe canvas consists of an undifferentiated bitmap of pixels. Once a collection of pixels are selected by an encircling operation (rectangle, lasso, or polygon), however, a new object is created representing this image material. In implementation, the selected pixels are copied into a new bitmap whose dimension spans the selected region and whose location is displaced to align with the original selected pixels, which are erased from the original bitmap. See Figure 3. The user is then free to drag this new image object around by holding the left mouse button. When the mouse is released, the image object remains a separate object, unlike a paint program which pastes selected pixels back into a common flat canvas layer. Once this image object has been created, it may subsequently be selected by positioning the mouse over any of its foreground pixels, and clicking left. Subsequent drag selection operations carve the existing bitmap objects into yet smaller pieces. This organization is akin to image layers, but without any demand on the user's conscious attention, and without consuming screen space with complex layer control apparatus. One subtlety of this interaction pertains to the distinction between foreground and transparent background pixels. This is elaborated below.

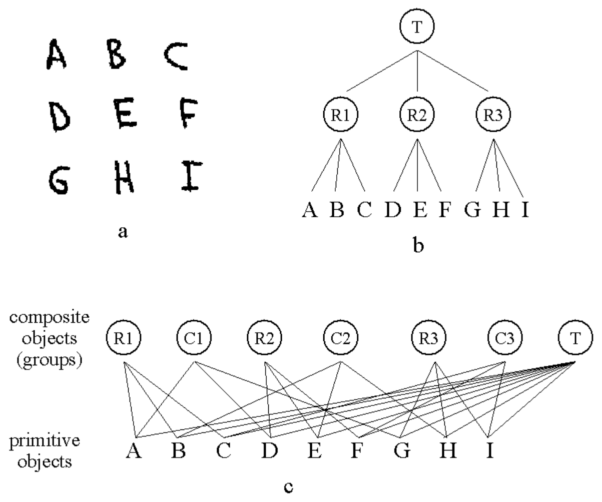

Figure 4: Hierarchical tree versus lattice grouping models. a. A scene which may be conveniently segmented into nine primitive bitmap objects. b. A hierarchical tree structure is capable of representing groups corresponding to, for example, the row structure and the entirety of this tabular image, but not both the row and column structure. c. A flat lattice grouping model permits simultaneous representation of rows, columns, and the entire table, because primitive objects may be linked to more than one parent composite object.

Lattice Grouping

An important facility in image editors is the ability to form groups, or temporary bindings of primitive objects, to which translation, scale, copy, or other operations will apply in common. The standard model for grouping is a hierarchical tree structure, as illustrated in Figure 4b. Most users of PowerPoint are familiar with the tedious ungroup-ungroup-ungroup:change:regroup-regroup-regroup procedure required to modify one object nested at the bottom of a grouping hierarchy. Also unfortunately, the hierarchical grouping model imposes the severe constraint that an object can belong to at most one group. This prohibits the simultaneous existence of multiple overlapping yet meaningful groups.

ScanScribe departs from the hierarchical tree grouping model and instead follows a flat lattice grouping structure as proposed in [17]. Any primitive image object may belong to any number of groups, or composite objects, as illustrated in Figure 4b.

The user interface problem raised by the flat grouping model is, how to invoke any given target group that may have been established. Related to this question is, how to display to the user what groups have indeed been established so they can choose among them. One possibility is to pop up a menu of choices; another is to overlay graphical visualizations of selectable image structures. These approaches could face problems of visual clutter and scalability, but they bear investigation and creative exploration. ScanScribe settles on a simple technique: clicking the mouse on an image object selects just that object. Clicking again selects the set of objects repre-

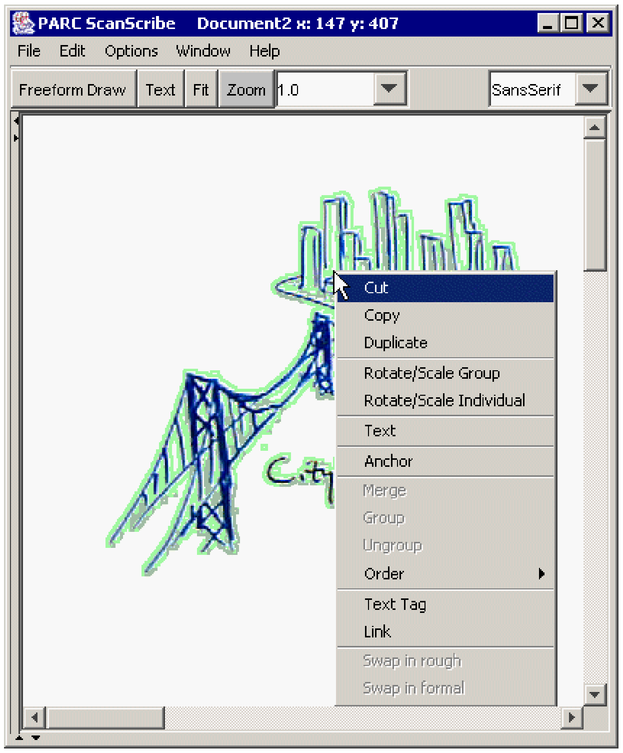

Figure 5: Mouse-right pop-up menu showing locally available edit operations.

sented by the first group, in a sequence of groups, that are associated with the primitive object clicked. Subsequent mouse clicks cycle through the groups in the sequence.

Each primitive image object maintains its own ordered list of groups that it belongs to. Each time any group is operated on, for example moving or rotating it, that group is promoted to the front of the group list for each constituent object in the group. In this way each primitive object maintains its participant groups in most-recently-used order.

Groups may be established by an explicit "group" command, or they may be created automatically. Clicking the mouse-right button on any selected object(s) invokes a pop-up menu of commands, as shown in Figure 5. One available command is to form a group, assuming the selected set contains more than one primitive object. Note also the availability of a merge command which permanently binds a selected set of primitives into a single bitmap image object. A group is established automatically whenever the user selects and then operates on two or more primitives. And, an object is automatically expelled from a group any time it is moved far from that group's spatial locus.

Because primitive image objects can be carved into smaller objects at any time by drag selection, and sets of primitives can be merged into a single object, some bookkeeping must go on to maintain consistent and coherent grouping structure as editing proceeds. For example, if an object belonging to a group is divided by selection into two smaller objects, both

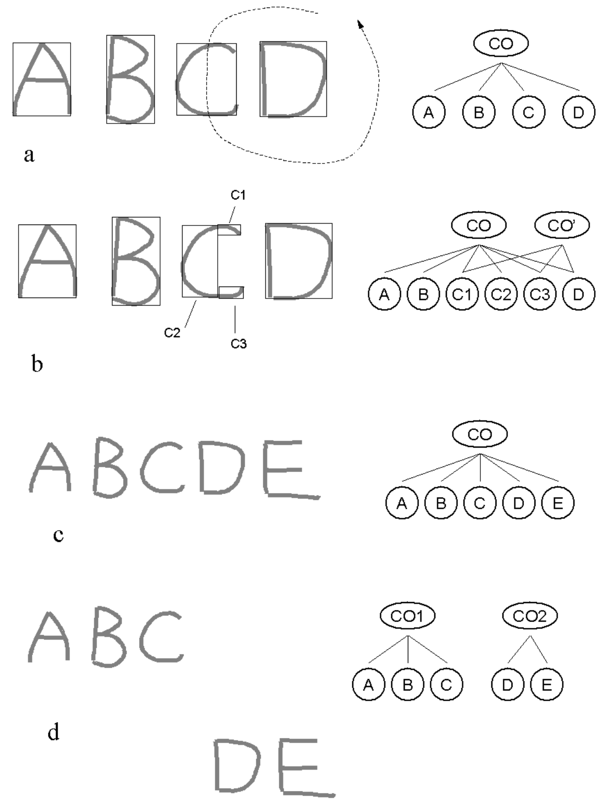

Figure 6: Illustrations of representative logic for maintaining consistency of grouping relations through editing operations. a. An image containing a group (Composite Object) of four primitive image objects is cut into smaller pieces by a selection stroke. b. The resulting grouping structure must reflect the fragmentation of this image object, "C". A second group is formed by virtue of the encircling selection. c. A grouping of five primitive image objects. d. When two of these are dragged away, the single group is destroyed but the resulting groups should preserve sensible remnants of the original bindings.

of these new objects must inherit the grouping links possessed by the original. Some of the major bookkeeping requirements are illustrated in Figure 6.

The result of this model is a very intuitive interaction with document image material. One can quickly individuate semantically significant image objects by mouse dragging, and once created, select them again by clicking. The shift key permits editing of the selection, that is, adding and removing objects from the highlighted selected set. Once any collection of objects have been operated on as a group, this group may be readily revisited by clicking the mouse some number of times on any member of the group. One shortcoming of this interaction design, which is also common to PowerPoint and other programs with hierarchical grouping, is that it is not visually apparent what groups are actually in existence. In the most common condition, however, users seem to maintain a background awareness of the groups they have been using recently on the basis of constituent objects' visual layout and their semantic roles in the scene. This matter bears further investigation through in-depth observational studies.

Foreground/Background Separation

Although ScanScribe's editing functions can be applied to any bitmap image, the program is designed toward editing of rough and formal document images in which foreground markings are arrayed against a white or light background. Important purposes for editing in this image domain include cleaning up image clutter and noise, rearranging text and graphics, and combining material from different sources. Only the foreground pixels are significant in these regards, and it is important that background pixels never occlude foreground pixels as image objects are moved around. Therefore it is normally critical that background pixels be rendered transparent.

Modern photographic image editors provide “magic wand” facilities for selecting collections of pixels on the basis of color or lightness, which can then be set transparent. Depending on the image source, this can be a tedious process. For example, users in one field study we have undertaken report that using Photoshop it takes on average 20 minutes to clean up the background in images of graphic charts captured with a digital camera. A recent commercial product, called WhiteBoard Photo, addresses this problem by providing image processing that normalizes image lightness and contrast across an unevenly lighted document image in a one-step process, leaving the background a uniform white.

ScanScribe includes an image processing module that performs this kind of document image color normalization and setting of a transparent background. One shortcoming of Whiteboard Photo and other techniques employing high-pass filtering to distinguish relatively compact dark markings against white or light background [15], is that they corrupt the interiors of larger foreground objects. See Figure 7. For ScanScribe, we have developed a foreground/background separation algorithm that uses an iterative technique to estimate and interpolate the color distribution of light background across the scene. To distinguish foreground from background we employ a combination of high-pass filtering and distance measures on pixels' HSB values. Figure 7c shows a representative result. Foreground/background separation can be applied selectively to entire images or regions of images that have been loaded into ScanScribe, or not at all. Under one user option this processing can also be applied automatically whenever an image is loaded.

Anchoring Image Objects

Once foreground/background separation has been achieved, the mouse behaves differently when positioned over visible foreground or transparent background, and the cursor changes accordingly as it passes across the image. When pressed on a foreground image object, the mouse selects that object. Holding the mouse button and dragging permits moving of that object. Repeated clicking cycles through the groups that object belongs to. When the mouse is pressed over transparent background, drag selection is initiated via the overloaded rectangle/lasso technique described above. This division of function works well under the assumption that drag selection op-

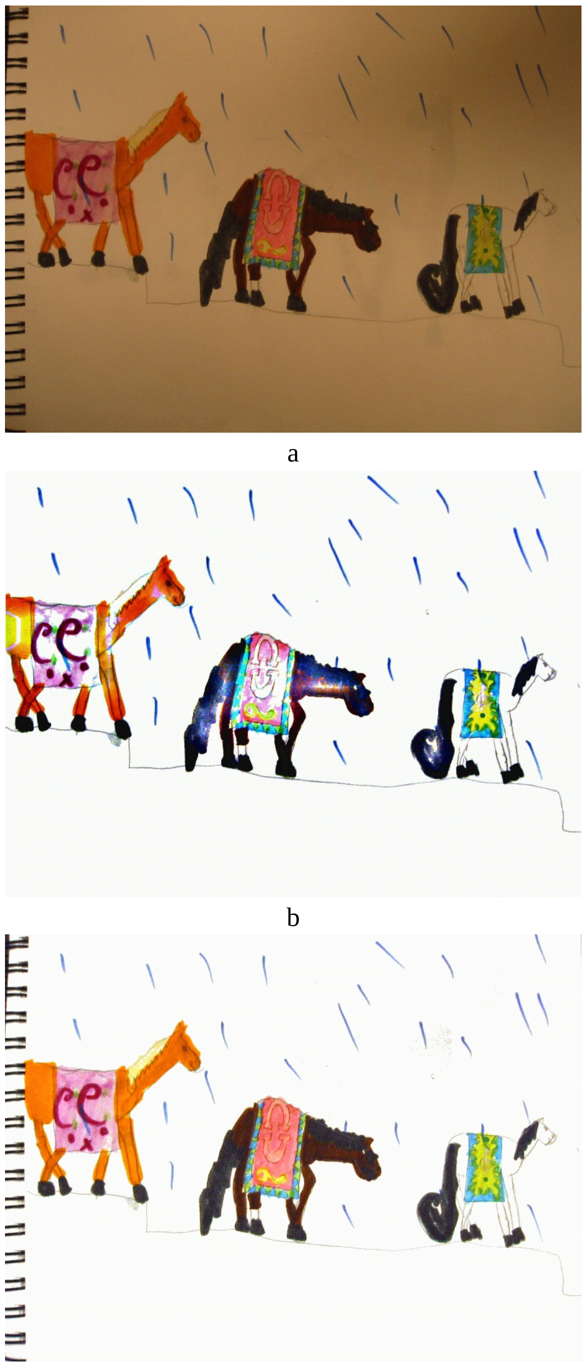

Figure 7: a. Original digital camera image. b. Processing by the commercial product, Whiteboard Photo. Note degradation of the solid color regions. c. Result of our algorithm which detects large foreground regions.

erations will be used to select visible foreground image objects mostly or entirely surrounded by transparent background.

On occasion, however, it is necessary to perform drag selection initiated on foreground pixels, for example to perform lasso selection on a person's face in a photograph. For these instances, it is possible to *anchor* an image object, via the right-mouse invoked pop-up menu. When anchored, an image object cannot be selected by positioning the mouse and clicking on it. Instead, a mouse press on an anchored image object invokes the standard drag selection facility. Image objects that are anchored may be un-anchored by completely enclosing them with a lasso, rectangle, or polygon encircling selector, or via the Edit menu items, "Select All" or "Select Anchored Image Objects."

In our experience, the notion of anchoring is the single most confusing aspect of ScanScribe to novice users. To understand the rationale for the concept requires understanding the notion of differentiated image objects, the dual click/drag modes of selecting, and the distinction between visible foreground and transparent background. The novice user does not sit down with these concepts immediately to hand. The greatest danger of confusion occurs when editing a document containing pure white background, but for which foreground/background separation has not been applied. Then, no background pixels have yet been set transparent, yet background pixels do not visibly appear. Drag selection results in large regions of white pixels which occlude other objects when moved around. One approach which we have yet to explore is to render transparent pixels in some visible manner, preferably less visually distracting than the checkerboard pattern which has become a standard in modern applications.

Text

One purpose for ScanScribe is the cleanup and organization of sketchy, scribbled, handwritten notes and graphics. Often the results of a meeting or collaborative session consists of handwritten items amongst drawn figures. A common practice is to document these results by transcribing handwritten text into a formatted document such as an ascii email message or web page. Current tools do not well support carrying sketches, figures, and diagrams through this process. ScanScribe seeks to address this problem by providing means not only to extract and organize graphical material, but to also replace scribbled written material with typed text as well as enter new text.

Text entry is initiated anywhere on the canvas by positioning the mouse where the text is to begin, and typing keystrokes. A text entry region then appears superimposed on the canvas, and standard text editing features (e.g. drag select, cut, copy) become available. To render this text to the canvas, the user may press the End key or click the mouse anywhere outside the text entry region.

To replace handwritten text with typed text, the user selects the bitmap region containing handwritten text, and commences typing. When text entry is completed by pressing the End key, the selected bitmap object is removed from the canvas and replaced with a typed-text object. ScanScribe does not currently employ handwriting recognition software but it is straightforward to augment text replacement-by-typing with this functionality to the degree it becomes technically available. In this way, ScanScribe is in accord with the Multivalent Document framework [14] which suggests interaction with static image content via different semantic layers built by automatic processing steps.

Typed text objects are a subtype of the standard bitmap object. They are rendered as bitmaps on the canvas, and can be carved into smaller bitmap pieces and rearranged. But they also retain the ascii representation of the character string entered, so they can be selected and re-edited as typed text.

Miscellaneous Features

ScanScribe supports a number of other miscellaneous features. Although we have not addressed the potential for entering vector graphic objects, there is a freeform drawing tool for freehand sketching with the mouse.

Images edited in ScanScribe can be written in standard bitmap formats such as JPEG and PNG. Also, images can be published as HTML for viewing in web browsers. Via the right button pop-up menu, it is possible to establish hyperlinks from image objects which then become HTML links in resulting web pages. The HTML Cascading Style Sheet protocol is used to position image objects.

ScanScribe is implemented in Java and makes use of the system clipboard facilities. One of the common uses for ScanScribe is as a clip-art collection tool, in conjunction with a search engine. Having browsed to an intersting web site, the user can employ the computer's screen hardcopy facility to grab an image, then paste into ScanScribe and proceed to extract and borrow desired graphic imagery for re-purposing in other documents.

AUTOMATIC STRUCTURE RECOGNITION

As described thus far, ScanScribe is a very functional and useful bitmap image editor that facilitates the manipulation of foreground markings in rough and formal document images. Beyond this, it a platform for introducing increasingly sophisticated image structure recognition tools as they become available.

The hook for this is ScanScribe's lattice framework for maintaining groups, which in the current design is accessed through the repeated-click cycling method for point-and-click selection of grouped primitives, as described above. Image recognition occurs in two stages. First, the original undifferentiated bitmap is segmented into primitive image objects by an automatic mechanism. Second, sensible groupings of these primitives are automatically formed and established as groups, or composite objects, in the lattice grouping structure. In other words, image recognition becomes a power assist on the grouping paradigm already available to the user through manual selection.

Procedures for accomplishing these steps are the subject of ongoing research. One question is, at what level of abstraction is image structure made available as groups? We envision a spectrum of recognition algorithms possessing knowledge of specific document domains and graphical communication language constructs, eventually specifically facilitating editing of image content found in engineering diagrams, schedules, calendars, charts, graphs, mathematical notation, and so on.

ScanScribe's initial capabilities are focused at the level of visual perceptual organization. This type of structure roughly aligns with the Gestalt laws of visual perception, which include proximity, smooth continuation, feature similarity, and figural closure [4, 9, 10, 19]. We have implemented fragmentation algorithms to decompose a document image into primitive objects labeled as curvilinear "strokes", which typically comprise line art, and compact "blobs", which typically correspond to characters of handwritten or printed text. Then, grouping algorithms assemble these into composite groups based on two kinds of criteria. Stroke primitives are grouped according to rules following the principles of curvilinear smoothness and closure, as illustrated in Figure 8. Blob primitives are grouped into composite objects reflecting words and lines of text based on spatial proximity and curvilinear alignment. Examples of these are shown in Figure 9. Note that the lattice organization for composite objects is important because any given image primitive may belong to multiple perceptual groups that may not necessarily bear subset/superset relationship to one another.

Structure recognition is invoked in ScanScribe through an Edit menu item, and can be applied to only a selected region of the image. In our current implementation this processing takes on the order of tens of seconds, depending on complexity of image material, so it must be used judiciously. Because structure recognition is used in an interactive editing context that permits the objects included and excluded from a selection set to be modified, even imperfect results can be useful even if they happen to include or exclude some amount of image material that to a human eye does or does not appear to belong to perceptually coherent and salient visual objects.

USE EXPERIENCE

ScanScribe has been deployed to approximately 30 people during its evolution, both within and outside our research center. Feedback has been mainly anecdotal, while more systematic testing and evaluation remains to be done. We have encountered great variability in the ease with which people are able to learn to use ScanScribe. Some require deliberate step-by-step training, while to others it just makes sense right away. One issue is that the user interface model for selecting and creating persistent bitmap objects and groupings of objects is novel, and one is not lead to understand it through inspection of a tool palette or other obvious visual indicators. We provide help in the form of video tutorial snippets accessible from the Help menu. Some users find these very satisfactory instruction, while others are quite reasonably unwilling to spend any time deliberately learning but expect the affordances of the program to teach itself.

The range of things people have used ScanScribe for is quite varied, including cleanup of notes from whiteboard sessions; merging lists spread across multiple scanned pages onto a single page; arranging an office floor plan by positioning scanned furniture templates; and producing casual cartoons. Our commercial test study site has incorporated ScanScribe into their work flow because of its superior quality of its color normal-

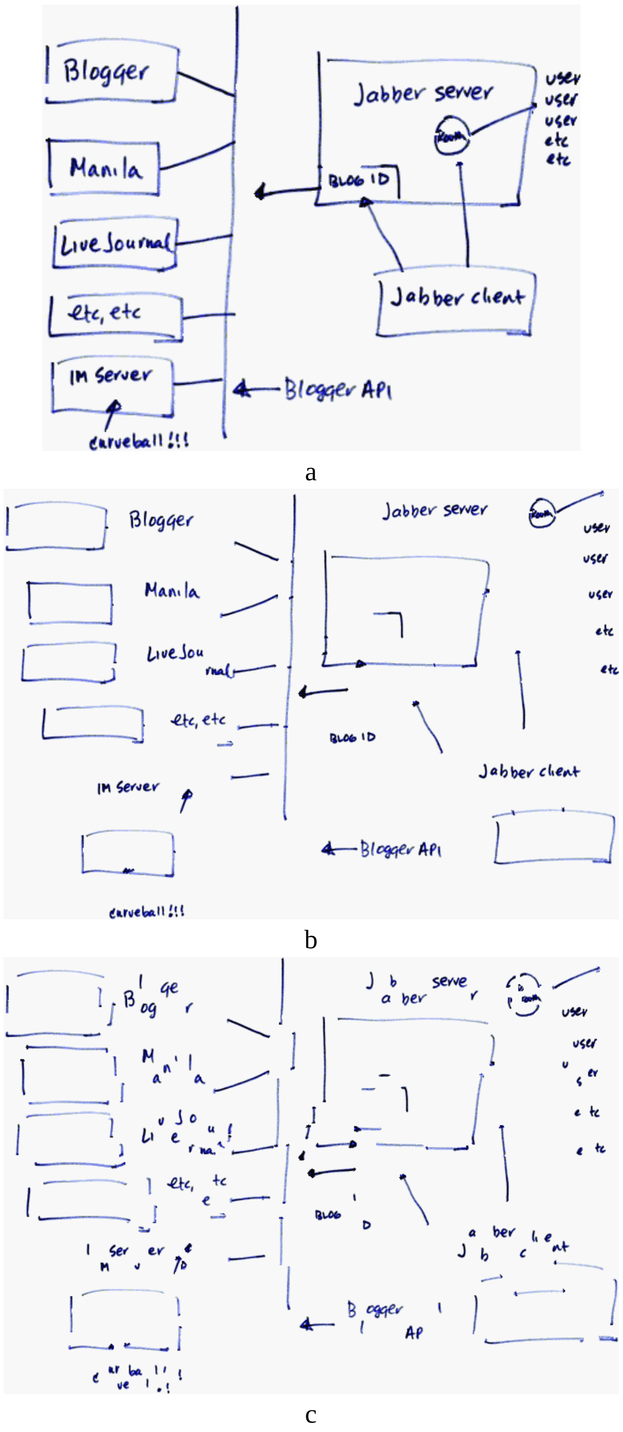

Figure 8: Visual structure found in a hand-drawn graphic figure. a. Original image after foreground/background separation. b. "Exploded view" of perceptually salient structures reflected as groups (composite objects) found by ScanScribe's automatic structuring processes. Each of these was selected for moving from its original location by pointing and clicking the mouse (at most four mouse clicks). c. Exploded view of bitmap image objects comprising the full set of image primitives. These objects' spatial relationships were analyzed by the structure recognition algorithms to form the groups reflected in b.

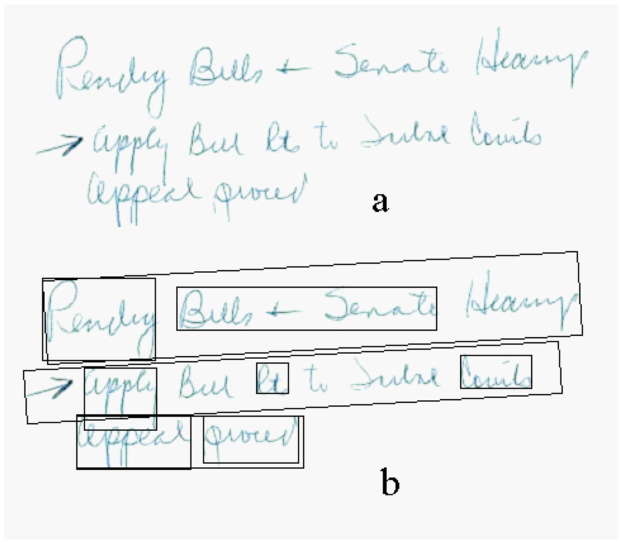

Figure 9: Visual structure found in handwritten text. a. Original image after foreground/background separation. b. Boxes indicate groups found by ScanScribe's automatic structuring processes. Because this is cursive handwriting, in many cases full words were represented by individual primitive bitmap objects (not shown by bounding rectangles).

ization on digital images of graphic charts.

CONCLUSION AND FURTHER WORK

We feel that future progress in increasing the capability and ease of use of editing programs will come from making their designs more closely reflect the nature of human perception. ScanScribe illustrates how the design of an editor for rough material can respect at a foundational level the perceptual phenomena of figure/ground separation and visual grouping. Automated foreground/background separation mirrors the natural human perceptual figure/ground interpretation of most documents presented to the system consisting of dark markings on a light background. ScanScribe's built-in support for constructing and manipulating groups departs from the strictly hierarchical group organization of conventional editors and enables flexible, overlapping grouping structures more commensurate with the phenomena of human visual grouping.

The application of perceptual principles in the design of an editor's core capabilities has strong implications for its user interface design. For example, ScanScribe's lattice grouping capability requires specialized interaction techniques for accessing the various groups the user may wish to access. We have explored one point in the design space, namely multiple clicking to cycle through groups associated with an object. Others interaction modes are possible, for example, the rough shape and size of selection gestures[17], and speech or other multimodal interfaces.

The ScanScribe user interface is designed for mouse interaction. We have also developed a prototype application, called *InkScribe*, sharing the ScanScribe architecture but designed to be used with a pen or stylus. The fundamental problem here is providing seamless means to switch between selection and editing of image material (command mode), and marking with digital ink (draw mode).

A remaining issue regarding the potential for widespread adoption of image editors especially targeted toward rough, or informal documents, is ease of image capture. We believe that the increased availability of low-cost, high-quality digital cameras will help to bridge the physical/virtual barrier for written and graphical communication. The ScanScribe document image editor offers a helpful tool for making use of formal or printed sketches, notes, and diagrams that have been brought into in the electronic world.

ACKNOWLEDGEMENTS

We thank Thomas P. Moran for substantial contributions in the conceptual and initial design stages of this project. We are also indebted to the testers and users of ScanScribe for their comments and feedback.

REFERENCES

1. Alvarado, C., and Davis, R.; [2001]; "Resolving ambiguities to create a natural computer-based sketching environment," *Proc IJCAI*, Seattle, Vol.2: 1365-1371.
2. Bagley, S., and Kopec, G.; [1994]; "Editing Images of text," *Comm. ACM*, Vol. 37, No. 12, 63-72.
3. Forbus, K., Furguson, R.W., and Usher, J.M.; [2001]; "Towards a Computational Model of Sketching," *Proc. IUI '01*, Santa Fe.
4. Green, C.; [2000]; "Introduction to: 'Perception: An introduction to the Gestalt-theorie' by Kurt Kaffka (1922)", "http://psychclassics.yorku.ca/Koffka/Perception/intro.htm".
5. Gross, M.; [1992]; "Graphical Constraints in CoDraw," *Proc. IEEE Workshop on Visual Languages*, Seattle, 81-87.
6. Gross, M.; [1996]; "The electronic cocktail napkin - computer support for working with diagrams," *Design Studies*, 17(1):53-70.
7. Hong, J.I., and Landay, J.A.; [2000]; "SATIN: A Toolkit for Informal Ink-based Applications," *Proc. ACM UIST*, San Diego, 63-72.
8. Igarashi, T., Matsuoka, S., Kawachiya, S., and Tanaka, H.; [1997]; "Interactive Beautification: A Technique for Rapid Geometric Design", *Proc ACM UIST*, Banff, 105-114.
9. Kanizsa, G.; [1979]; *Organization in Vision: Essays on Gestalt Perception*, Praeger, New York.
10. Koffka, K.; [1922]; "Perception: An introduction to Gestalt-theorie. *Psychological Bulletin*, 19: 531-585.
11. Li, Y., Guan, Z., Wang, H., Dai, G., and Ren, X.; [2002]; "Structuralizing Freeform Notes by Implicit Sketch Understanding," in *AAAI Spring Symposium on Sketch Understanding*, Stanford University, AAAI TR. SS-02-08.

12. Landay, J.A., and Myers, B.A.;[2001]; "Sketching Interfaces: Toward More Human Interface Design," *IEEE Computer* V. 34. No. 3, March 2001, 56-64.

13. Pedersen, E., McCall, K., Moran, T, and Halasz, F.; [1993]; "Tivoli: An electronic whiteboard for informal workgroup meetings," *Proc ACM CHI*, 391-398.

14. Phelps, T.A., and Wilensky, R.; [1996]; "Multivalent Documents: Inducing Structure and Behaviors in Online Digital Documents," *Proc. 29th Hawaii International Conference on System Sciences*, Maui, 144-152.

15. Pilu, M., and Pollard, S.; [2002]; "A light-weight text image processing method for handheld embedded cameras," *Proc. British Machine Vision Conference*, Cardiff University.

16. Saund, E.; [2003]; "Finding Perceptually Closed Paths in Sketches and Drawings," *IEEE TPAMI*, V. 25, No. 4., 475-491.

17. Saund, E. and Moran, T.; [1994]; "A perceptually supported sketch editor" *Proc ACM UIST*, Marina del Rey, 175-184.

18. Shilman, M., Wei, Z., Sashi, R., Simard, P., and Jones, D.; [2003]; "Discerning Structure From Freeform Handwritten Notes," *Proc. Int. Conf. Document Analysis and Recognition*, Edinburgh.

19. Wertheimer, M.; [1923]; "Laws of Organization in Perceptual Forms", in Ellis, W., ed, *A source book of Gestalt psychology*, Routledge & Kegan Paul, London, 1938.

20. Wilcox, L., Schilit, B.N., and Sawhney, N.; [1997]; "Dynomite: A Dynamically Organized Ink and Audio Notebook," *Proc. ACM CHI*, Atlanta, 186-193.