ScreenCrayons: Annotating Anything

Dan R. Olsen Jr., Trent Taufer, Jerry Alan Fails
Brigham Young University
Computer Science Department, Provo, Utah, USA
olsen@cs.byu.edu,

ABSTRACT
ScreenCrayons is a system for collecting annotations on any type of document or visual information from any application. The basis for the system is a screen capture upon which the user can highlight the relevant portions of the image. The user can define any number of topics for organizing notes. Each topic is associated with a highlighting "crayon." In addition the user can supply annotations in digital ink or text. Algorithms are described that summarize captured images based on the highlight strokes so as to provide overviews of many annotations as well as being able to "zoom in" on particular information about a given note and the context of that note.

General Terms:
Human Factors

Author Keywords
Annotation, Screen capture, digital ink, image summarization.

ACM Classification Keywords
H.5.2 User Interfaces

INTRODUCTION

This paper describes the ScreenCrayons system for capturing and managing annotations for a variety of tasks using any application. The goal is for the system to be very flexible, lightweight and widely applicable.

The fundamental metaphor for the modern office workstation has been paper. Beginning with the design of the Xerox Star, windows have been modeled as active sheets of paper. The most commonly used word-processing, drawing and spreadsheet applications all use paper as their metaphor. In the days before computers, creating documents was hard (typewriters are not error-friendly). Distributing paper documents was hard. Modifying paper documents was hard. Annotating paper documents by making marks on them was easy. Because of this disparity of labor, the focus of most office tools has centered on the creation and dissemination of documents that can readily be rendered onto paper. In this context the process of annotating the paper received much less attention despite its importance in actual use.

The advent of very cheap storage, cheap communication via the Internet, standard formats such as PDF or HTML and the pervasive availability of computing has caused a shift in our usage of documents. For an increasing number of people, the majority of their reading experience is digital rather than paper. Email has rapidly replaced the paper letter for much correspondence. Scholars increasingly subscribe to digital libraries rather than print journals. Technical manuals and promotional materials increasingly come through the web.

Adler et al [1] have reported that reading occupies 70% of document-related activity. However, for many subjects a substantial amount of reading time occurred in conjunction with writing. In the same study creation and updating of documents constitutes only 18% of writing while reading. On the other hand annotation and note taking consume 48% of the time. Schilit describes this as "active reading" [19] where the user is augmenting, filtering, highlighting, summarizing and organizing the information that they are reading. What we need are widely applicable computer-based tools that support this activity.

We will first provide an extended example of the range of annotations that we expect of our system. We then will review prior work in annotation, followed by the architectural issues with being able to annotate anything and review our notes in a meaningful way. This is followed by a description of the note taking process along with algorithms for associating image regions with our highlight marks. Lastly we discuss how we use these highlights, regions and notes to provide summarized views.

An Example

Consider Fred the biologist. Fred is an expert on nematode genomics. When he starts work one morning, he begins to read his email and finds that one of his students has posted a copy of her thesis on a web site so that he can review it. Fred wants to complete his email so he makes a note of where the thesis is located and continues. He soon finds that he is requested to do an urgent review for a journal paper. Again he makes a note of the paper and continues. He finds a message with a budget for a research proposal. He notes that the budget's travel and fringe benefits are not correct and sends the note back to his colleague. He also finds a message from a student indicating the results of a successful sequencing analysis that he adds to his list. Fred finishes his email, reviews his "todo" notes and decides that his student's experiments are the most important.

Fred then runs a special sequencing program that shows the results of the latest supercomputer run. He is excited by the results but not completely happy with the software settings that were used. He makes a note on those settings and forwards it to his student.

Fred then downloads the thesis and opens it in his word processor. While reading he makes notes of corrections. In the middle of his reading he thinks of a related paper that his student has not seen. He opens a web browser and begins a search for "nematodes." While doing the search he sees a page for a new center for nematode genomics. He makes a note to remember to review the site later. He also encounters the "Nematode Songbook" [17] with a hilarious rendering of an old western tune redone with worms. He notes this in his list of "worm humor" to share at a future conference.

Having completed the thesis reading Fred begins work on the journal paper that came to him as a PDF file. Working through the paper he finds several problems with its references to prior work. He makes notes as he goes along. At one point he finds a relevant paper and makes a note that links a paragraph in the new paper with the place where it should be cited in the one that he is reviewing. Having read the paper Fred goes back through his notes and begins to write the review. He reorganizes his notes into a better structure for the review and then begins to write. When he reaches one of his notes he finds that he needs the actual reference of a paper. He looks at the context of the note to find where the paper came from so that he can tell the author where to look.

There are several points that this example illustrates about note taking.

1. Notes occur spontaneously during work and as with the "Nematode Songbook" they are not always related to the current task.
2. Notes occur in the context of many applications. In our example Fred used email, a word processor, a spreadsheet, a web browser, a PDF reader and a homegrown piece of special software.
3. Notes are frequently a summary or highlighted excerpts from other reading. They serve to focus attention so that in the future the entire document does not need to be reread.
4. Notes are frequently a source for later writing as in the review to be written.
5. Notes can link disparate sources of information where the user has found a relationship, as in the linking of a desired citation to a segment of another paper.

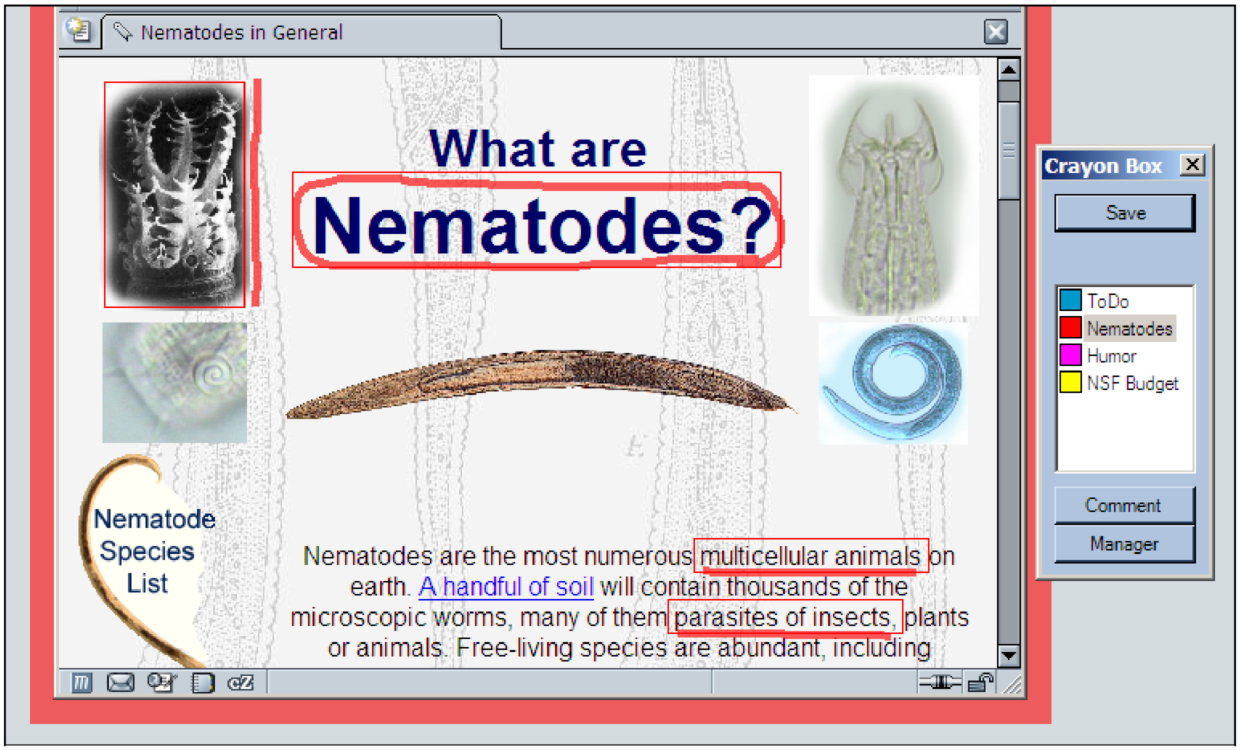
The goal of ScreenCrayons is to provide a lightweight, universal note taking facility that satisfy all of these needs without interfering with other work.

Prior Work

There have been several projects to support the annotation and organization of information. Some systems such as Notecards [14], gIBIS [8] and Aquanet [15] have the user enter their notes in separate structures. These approaches provide structure to the information, but they are self-contained and insular. The user must explicitly enter the desired information into the structure and in doing so much of the context is lost. The note is there but other than a possible URL or bibliographic reference, the rest of the context from which the note was taken is no longer readily available. A problem with notes is that their creation is viewed as work to be minimized, whereas their ultimate use frequently requires much more information. A user is faced with "work more now" to support "possible use later". This work/future benefit tradeoff is usually resolved by a choice to work less now in extracting information for notes. We need a lighter weight (easy to create) model of note taking that preserves as much context as possible so that it can be retrieved. Thus the user gets a great deal of preserved information with little current effort at the time a note is taken.

The XLibris project [19] tries to bring annotation in contact with the reading process through a "reader's notebook" and annotation marks. XLibris provides a rich set of annotations and nice mechanisms for summarizing documents and searching for other documents based on the annotations. The annotation facility, however, is built into a special reading application. This would not work for notes concerning Fred's gene sequencing software. Nobody else in the world has software like Fred's, but he needs to make and share notes about it. Microsoft's discussion and comment facility allows notes to be embedded or attached to Office documents and then shared with others. Adobe Acrobat allows notes to be added to documents.

An important early annotation system was the Dynomite project [20]. The primary data was digital ink and audio. The digital ink would capture the user's intent and comments, and the audio would capture contextual information that was going on in the environment. In many ways we follow this approach except that we use the user's computer screen as context information rather than ambient audio. Dynomite's use of ink and audio do have the nice property of being independent of the user's purpose. We



Nematodes are the most numerous multicellular animals on earth. A handful of soil will contain thousands of the microscopic worms, many of them parasites of insects, plants or animals. Free-living species are abundant, including

Figure 1 - Crayon Highlighting

seek to mirror this property with image rather than audio information.

Most annotation systems only support annotations of specific artifact/document file types. These include the XLibris reading appliance, Microsoft Office, specialized XML formats [13] or HTML [4]. The Watson system [7] provides for "application adapters" that must be uniquely written for each new application. An annotation system that depends upon specific application implementations is awkward to use and is frequently deimplemented by new releases of the software. More importantly it is very painful to learn new, mutually incompatible note systems for each application. The E-Quill[9] annotation system provides nice flowable notes, but can only create such notes using Internet Explorer and can only annotate web pages. One of our goals is to provide annotation facilities that are independent of application implementations or their file formats while still providing a rich capability. We want an "information foraging" tool that is pervasive across all of a user's work.

The Pervasive Annotation Architecture Problem

Creating a pervasive annotation facility presents an architectural challenge. The easiest approach to annotation is to create a special purpose model for all artifacts to be annotated. Anything to be annotated must be translated into this model. That is the approach taken by most prior annotation tools. The virtue of this approach is that the notes can be embedded in the model representation and a variety of views that display content relative to those notes can be designed and implement. The challenge is that the artifact model design restricts what can and cannot be annotated. If we take a document-centric approach like XLibris or a digital ink approach like Dynomite there are many applications that are shut out because our information model is not sufficiently rich to represent them. This also necessitates creating "translators" as in Watson. If no translator exists for your application, then there is no annotation facility.

A second approach is to create a special protocol to which all applications must conform. Such a protocol would include "here is a digital ink stroke, return me an annotation reference", "here is an annotation reference, display it in your application" and "here is an annotation reference, return me its bounding box on the screen". Such a protocol would allow annotations to be attached to any application that conforms to the protocol. The down side of this approach is getting all interesting applications to conform.

The approach used in ScreenCrayons is to annotate exclusively in image space. All GUI applications must render their information as images. All major windowing systems provide the ability to capture screen images. Thus we have a universal medium for annotating any information from any application without requiring the cooperation of that application. This is the heart of making annotations pervasive. The downside is that without access to the model the annotations cannot take advantage of the model's structure to behave intelligently. ScreenCrayons attempts to overcome this by inferring simple structure from the image itself. Another disadvantage is that some of the non-visible context is lost. This would include portions of a document currently scrolled out of sight.

Requirements for an Annotation Tool

In thinking about annotations we find also that they are frequently spontaneous and not always related to the task at hand. One may receive an email, encounter a web page or see a reference in a paper that relates to a task other than the user's current activity. When this occurs, the user experiences a tension between breaking the flow of the current task or potentially forgetting this nugget of information for a different task.

Our requirements for an annotation tool are: 1) the notes must be taken in the context of the user's work rather than in a separate application, 2) taking notes must be a very lightweight task involving very little user effort, 3) the note must preserve the visual context in which the note was taken, 4) the resulting notes must be condensed and summarized to that they can be easily browsed and manipulated later.

There are two fundamental pieces to an annotation system. The first is how notes are created and stored and the second is how the user browses the collection of notes. We will address each of these in turn.

CAPTURING ANNOTATIONS

As the user is reading documents, browsing the web or performing other work they may come upon some information that is of interest to a topic. Note taking involves three activities: 1) indicating the artifacts on the screen to be annotated, 2) indicating the topic to categorize the note and 3) adding optional commentary to indicate what is important about this note.

Consistent with our work on Image Processing with Crayons [10] we associate each of the user's topics of interest with a crayon. Crayons are essentially digital ink dispensers that are kept in a "crayon box." This is similar to the Intelligent Pen [13]. These crayons are similar to the categories in Dynomite [20]. When the user sees information of interest to some topic, they grab the crayon for that topic from their crayon box and use the crayon to draw a highlight on the screen as in figure 1. This "scribble on the screen" metaphor is trivial to learn and completely independent of any application.

Note capture is implemented by performing a screen capture. The Freestyle [11] system and Alias Sketchbook Pro[2] use screen capture for annotation. However, to just file away the screen shots is not sufficient because the results are unusably large. By definition, the user can only view one full screen shot at a time. Sketchbook treats its captures as images to be panned and zoomed rather than as notes to be summarized or expanded. Other than a traditional layered drawing mechanism, Sketchbook provides no convenient mechanism to visually distinguish or organize annotations relative to the image that they annotate. This is extremely awkward as an annotation tool. Microsoft provides a "snipping tool" for the tablet PC that can capture small screen fragments to file away as notes. The problem with capturing selected sections is that small easy to use snippets lose their visual context. Our approach is to allow the user to annotate what interests them and then use that information to summarize what was captured. The process of perusing captured images and their summaries will be discussed in a later section.

After capture, the image is displayed in a borderless window on top of all other windows. To the user it looks as if nothing has changed except that all of the applications are inoperative and mouse gestures will draw highlights on the screen instead of interacting with the application. To show the user that this is a new mode, the active window is bordered in a transparent highlight that is the same color as the selected crayon and all background applications are blended with a light color to deemphasize them. This makes it clear to the user that they are in crayon mode. The "crayon box" also appears on the screen. The crayon box provides several options for selecting other crayons, saving the note, canceling the note or editing the crayons.

Requiring the user to context switch into "crayon mode" may seem cumbersome, but there is no other choice. Without control of the underlying applications, there is no other way to distinguish between annotation inputs and application inputs. Any annotation pen or mouse gesture could also be interpreted as an input to the underlying application. The overlay of the entire screen with an annotation image preserves application context while clearly indicating that input gestures are now annotations, not application operations.

In ScreenCrayons, a note is composed of a name, a screen image, the ink from the crayon highlights and zero or more comments created by the user using either ink or typed text. This representation is usable on any application and is independent of any specific implementation.

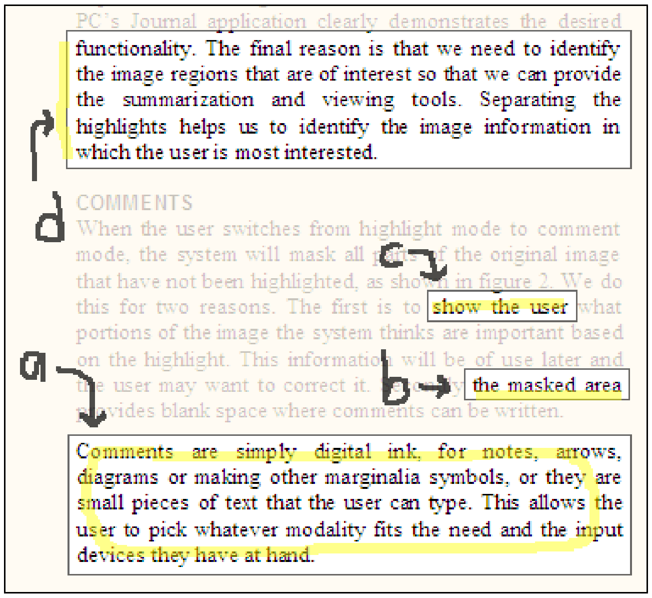
The annotation process has two modes: highlighting and commenting. The user switches between these two modes by a button on the crayon box. When highlighting, the user uses the crayon to indicate those portions of the image that are related to the crayon's topic. The highlight strokes are immediately associated with a region and displayed on the image as a rectangle in the color of the crayon, as shown in figures 1 and 2. This shows the user what portion of the image the system thinks are important based on the highlight and allows the user to make corrections if necessary. The user can make a more precise encirclement stroke to select exactly the right region. We will discuss the stroke/region association algorithm in a later section.

We separate highlights from comments for three reasons. The first is that our annotation system must know which portion of the full screen image actually applies to the selected topic. Capturing notes is only part of the problem. The user will later want to review and reorganize the notes. Presenting a full screen image in such a case will be very cumbersome. The highlights are what guide image summarization. The second reason for the separation is that the user's commentary should be searchable. We did not implement searching handwritten ink, however, the Tablet PC's Journal application clearly demonstrates the desired functionality. The third reason is that although documents have white space margins in which to write notes, most applications do not. Even documents do not provide sufficient white space to handle the size handwriting that most people use with digital pens. When switching to comment mode, the unselected regions of the screen become usable scratch space on which to write notes.

COMMENTS

When the user switches from highlight mode to comment mode, the system will mask all parts of the original image that have not been highlighted and surround the highlighted regions with boxes, as shown in figure 2. The masked area provides blank space where comments can be written.

Comments are simply digital ink, for notes, arrows, diagrams or making other marginalia symbols, or they are small pieces of text that the user can type. This allows the user to pick whatever modality fits the need and the input devices they have at hand.


PC's Journal application clearly demonstrates the desired functionality. The final reason is that we need to identify the image regions that are of interest so that we can provide the summarization and viewing tools. Separating the highlights helps us to identify the image information in which the user is most interested.

COMMENTS

When the user switches from highlight mode to comment mode, the system will mask all parts of the original image that have not been highlighted, as shown in figure 2. We do this for two reasons. The first is to show the user what portions of the image the system thinks are important based on the highlight. This information will be of use later and the user may want to correct it. Secondly the masked area provides blank space where comments can be written.

Comments are simply digital ink, for notes, arrows, diagrams or making other marginalia symbols, or they are small pieces of text that the user can type. This allows the user to pick whatever modality fits the need and the input devices they have at hand.

Figure 2 – Masking unhighlighted regions

We use the regions for each highlight stroke that were calculated in highlight mode to associate a region of the image with each stroke. We then obscure all unhighlighted regions by blending them with a light color. This is similar to the highlighting technique described in [18]. By using a blend, the unselected regions fade into the background. This highlights the selected regions while preserving their context. The blended areas now form a more uniform region where ink and typed notes can readily be seen.

HIGHLIGHT/IMAGE REGION ASSOCIATION

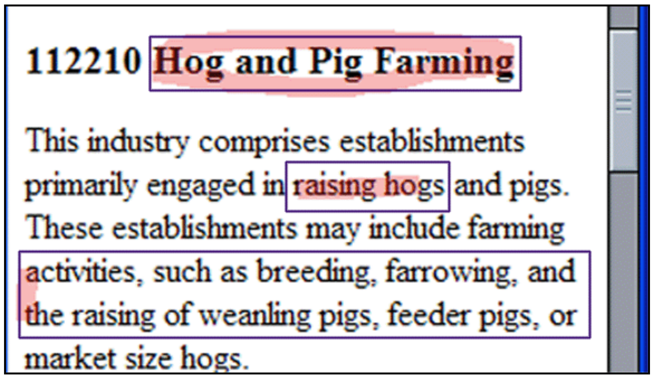
Each highlight mark that the user makes is associated with some region of the image. This association can be challenging because of the various kinds of marks that a user might make and the fact that the annotation system has no control over the underlying applications and how they lay out information. Golovchinsky [12] and later Bargaron [4] identify five types of marks that people make on paper. They are: circles (figure 2-a), underlines(2-b), highlights(2-c), margin bars (2-d) and marginalia. Similar annotation marks are found in XLibris[19]. For our purposes we consider the first four to be highlighting activities that identify areas of interest. The marginalia we classify as comments. Each of these highlights has distinct ways in which they are associated with the imagery being annotated.

The highlight/region problem is one of taking each ink stroke in the highlight, classifying it into one of the four categories and then computing a bounding rectangle for the associated image region. Unlike other stroke/content association techniques, we do not have an underlying model of the information. We must infer all structure from the image with no other knowledge.

Classification of the marks is based on the bounding box for the ink stroke. The vertical/horizontal aspect ratio of the bounding box can be used to detect vertical margin bars and horizontal underline/highlight marks. Anything that is not a vertical or horizontal line is treated as a circle/scribble.

Once the highlight strokes have been classified, we next must associate those strokes with rectangular image regions. This is complicated by several factors. First is the diversity of images that one might highlight ranging from landscape images, floor plans, schematics, documents, spreadsheets and anything else people may want to annotate. Many times there are textures such as web page backgrounds that people treat as uniform when in fact they are not. Lastly there is the fact the people make sloppy marks and are not completely accurate about what they want to highlight.

The basis of our approach is to extract natural boundaries from the image that can then be associated with the marks as in figure 3. Our basis for such natural boundaries is that most applications use long runs of "uniform" color to visually segment their presentations. Such runs might be borderlines, white space between lines of text or paragraphs or other long uniform areas. We must account for the fact that people frequently treat textured or gradient backgrounds as uniform in the sense of requiring attention when in pixel terms they are not uniform at all.


112210 Hog and Pig Farming

This industry comprises establishments primarily engaged in raising hogs and pigs. These establishments may include farming activities, such as breeding, farrowing, and the raising of weanling pigs, feeder pigs, or market size hogs.

Figure 3 – Natural Image Boundaries

Continuity Images

To rapidly recognize these "uniform" runs we create vertical and horizontal continuity images. Continuity images are inspired by the integral images approach to computing features across large areas of an image in constant time. The algorithm for computing a horizontal continuity image is as follows.

```
Forall (X and Y in IMG)
    if (X==0) HCONT[X,Y]=1
    else if (diff(IMG[X,Y],IMG[X-1,Y])<threshold)
        HCONT[X,Y]=HCONT[X-1,Y]+1
    else
        HCONT[X,Y]=1;
```

The nature of the continuity is defined by the *diff* function and its *threshold*. Our implementation uses a difference function that is the maximum of the absolute values of the RGB differences. Our empirical trials found a threshold of 55 in a 0-255 RGB space to perform well. Essentially we are looking for intensity contrast. When people use textured backgrounds, they tend to keep the contrast low so that the foreground detail will stand out. Note also that this algorithm compares the difference pixel by pixel. Gradient backgrounds have small pixel-to-pixel differences even though there may be a large difference from one end of the run to another. We compute the vertical continuity map in a similar manner.

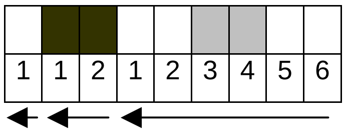


Figure 4 – Run Counting in Continuity Images

As figure 4 shows, each pixel of the horizontal continuity map contains the number of the pixels to the left that are part of the same "uniform" run. If we start from the rightmost pixel, we can find all of the runs in O(R) where R is the number of runs on that line. We can do this because the run length stored in one pixel will tell us where the next run to the left will end. Since we are only interested in regions with a few long runs this algorithm is very efficient. We can discard any line that shows several short runs. The continuity image allows us to evaluate the lengths of various runs in any part of the image in constant time. This is very important to an efficient boundary search algorithm.

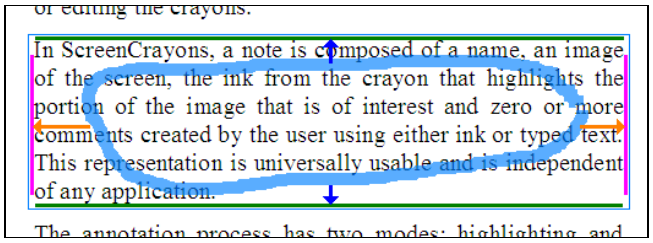
In searching for a boundary, we have a predicted length for that boundary. In the case of an underline, for example, the predicted boundary length is the length of the underline stroke. To find a boundary we search for a run that is greater than 98% of the predicted boundary length. The purpose of the 98% is to handle special cases of borders and other kinds of marks that are naturally in some images. When searching for horizontal boundaries, such as space between lines, we look for single runs of sufficient length. However, when searching for vertical boundaries we require 3 boundary runs together of acceptable length to declare a vertical boundary. This is to prevent accidental detection of aligned character or word spacing as a major boundary.

Stroke/Region Association

Having classified the highlight strokes and computed our continuity maps we now can compute the rectangular region associated with each stroke. Our algorithm searches in each of the four directions to find natural boundaries that correspond to each ink stroke. Stroke/region association has four cases: 1) area marks such as circles and scribbles, 2) horizontal underline, 3) horizontal highlight and 4) margin bars.

Circles and scribbles

Finding the corresponding regions of circle and scribble strokes is the easiest. We start with the bounding box of the stoke. We then shrink this bounding box to account for ink marks that actually surround the desired region rather than highlighted over the top. As shown in figure 5, we search in each direction from the ink stroke's bounding rectangle to find the region bounds.


of editing the crayons.

In ScreenCrayons, a note is composed of a name, an image of the screen, the ink from the crayon that highlights the portion of the image that is of interest and zero or more comments created by the user using either ink or typed text. This representation is universally usable and is independent of any application.

The annotation process has two modes: highlighting and

Figure 5 – Finding Regions for Circles/Scribbles

Underlines

If a horizontal stroke is in a boundary region of the same or greater length, as in figure 6-a, then we assume the stroke is an underline. From the vertical center of the ink stroke (red line) we search up to find the bottom boundary (green line) and then search up from that boundary to find the top boundary (green line). We then shrink the length of the highlight to give starting points for the vertical boundaries. We use the space between the top and the bottom as our predictor for the vertical boundaries. We then search outward to find these boundaries.

Horizontal highlights

Horizontal ink marks that are not in a boundary area are assumed to be highlights, as in figure 6-b, and we search down from the middle of the ink stroke to find the bottom boundary and then up to find the top. After finding the upper and lower boundaries for highlights and underlines we search left and then right to find the other boundaries (pink lines) as with the underlines.

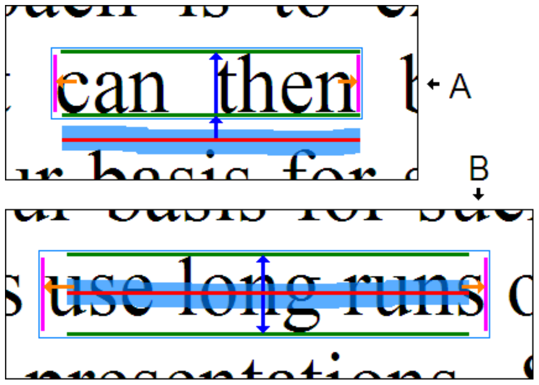
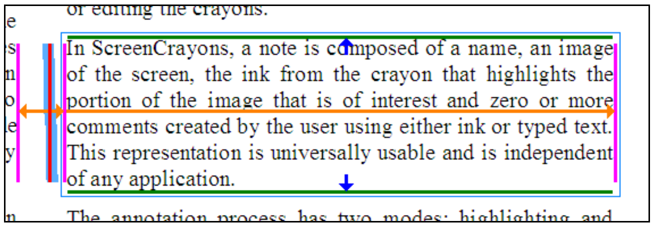


Figure 6 – Finding Regions for Highlights/Underlines

Margin bars

Margin bars are similar to underlines except that they can be in the left margin, right margin or in a gutter between areas of interest. Margin bars are generally drawn in a vertically uniform area. We can quickly decide how wide this area is and then determine whether the bar is closer to the left or the right side of the white space. The bar will generally be drawn nearer to the information being highlighted. In figure 7, we see that the center of the ink stroke (red line) is closer to the region on the right. This boundary then becomes the left boundary of the region we are finding and we continue searching to the right to find the right boundary of the region. Then the top and bottom boundaries of the region are found as before. A similar left searching approach can be used if the ink is closer to the material on the left.


of editing the crayons.

In ScreenCrayons, a note is composed of a name, an image of the screen, the ink from the crayon that highlights the portion of the image that is of interest and zero or more comments created by the user using either ink or typed text. This representation is universally usable and is independent of any application.

The annotation process has two modes: highlighting and

Figure 7 – Finding Regions for Margin Bars

Document Analysis Algorithms

The process of finding white space, text space and images in an array of pixels has long been studied as part of document analysis [3]. There are a variety of techniques for decomposing an image of a page into its components and to discover the document structure. There are techniques using projection profiles to recursively decompose a page. There are "smearing" techniques the blur text into lines and paragraphs. There are techniques for finding the baseline of text[5] as well as techniques for identifying large rectangular blocks of background.

We moved away from such techniques for several reasons. The first is that we are annotating anything, not just documents. The foreground/background approach of document analysis does not always fit. Computer applications regularly swap dark/light for foreground background frequently in the same application. Web pages and other applications introduce gradients, textures and other non-uniform features into the background. In many cases it is not background color that defines the best border. When highlighting a region of spreadsheet cells it is the cell separator lines in foreground color that provide the best boundaries. The second reason was that by annotating anything we could not make document-centric assumptions about how to infer structure. Thirdly we wanted an algorithm that would be interactively responsive. The goal is to amplify the user's intent rather than to understand the document. The driving force is the user input rather than structure inference. We wanted techniques that behaved simply and predictably in a variety of situations with clear mechanisms to override. We did not want users arguing with the system about what had just been highlighted. Lastly we wanted an efficient algorithm that would operate at interactive speeds.

Region Association Algorithm Verification

In order to verify the accuracy of our stroke to region association algorithm, we created a test suite to interactively capture screen images, mark them with highlight strokes, and manually specify the appropriate region for each stroke. The manually specified regions were then compared to regions that the algorithm found to determine accuracy. Our test data included 46 images with a total of 363 ink strokes. Our test images included screen shots from office applications, document readers, web pages, and others. In order to verify how well the continuity images handle finding continuous runs in textures, 15 of the included screen shots have moderate to highly textured backgrounds. The ink strokes correspond to 1,452 region boundaries, since there 4 boundaries per region.

The algorithm was able to correctly identify 1,374 (94.6%) of the region boundaries to within a distance of 3 pixels. The remaining 78 boundaries were checked manually to see if they were still acceptable boundaries even though they were not within the 3-pixel distance. We found that 62 (4.3%) of the boundaries were still acceptable. Even though they were not within 3 pixels of the ideal boundary, the calculated boundary was still acceptable. This is mainly due to the fact that regions without well-defined boundaries were being highlighted such as images with feathered edges, and therefore didn't have a well-defined boundary.

This leaves only 16 (1.1%) of the region boundaries as incorrectly calculated by our algorithm. These can be easily corrected since the user gets immediate feedback on the region associated with each ink stroke.

WORKING WITH CRAYONS AND NOTES

Once notes are created they also must be browsed, augmented, discarded and organized if they are to become useful. The crayon manager, shown in figure 8, provides a simple tree structure for organizing both crayons and their notes.

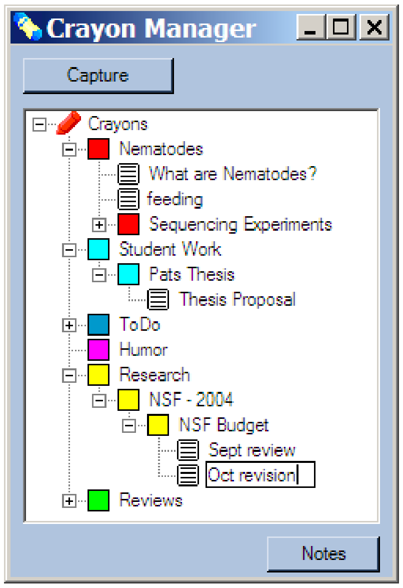


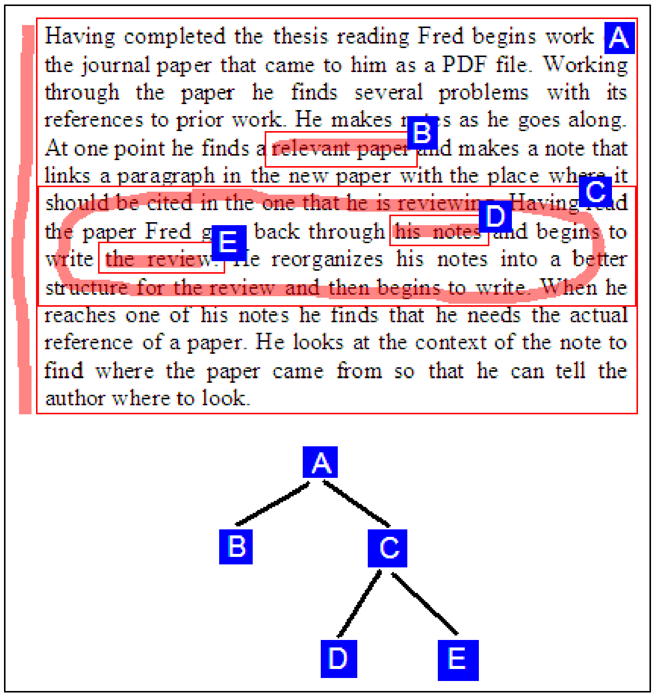
Figure 8 – Crayon Manager

The crayon manager provides the interface for creating new crayons, setting their ink parameters (color, thickness, transparency), organizing them into a hierarchy and providing labels for notes. The capture functions of the crayon box are also possible in the crayon manager. The crayon box is simply a very lightweight mechanism for grabbing frequently used crayons. In essence the crayon box is the "active cache" for the crayon manager.

Viewing Notes

The crayon manager is also the interface for locating and viewing previously captured notes. The user can "open" a crayon and see all of its notes. This task poses a challenge when each note consists of a full screen shot. It is necessary to summarize the images to their essentials so that many notes can be shown at once and the user can browse through them. Because we are working exclusively with images we do not have the information model available in systems like XLibris[19].

The key to our image summarization is the selected rectangles associated with highlight strokes. We take these rectangles and form a containment tree, as seen in figure 9. One rectangle is contained by another if most of its space is within the larger rectangle. Most rather than all is used to allow for user and algorithm sloppiness.



Having completed the thesis reading Fred begins work the journal paper that came to him as a PDF file. Working through the paper he finds several problems with its references to prior work. He makes r s as he goes along. At one point he finds a relevant paper and makes a note that links a paragraph in the new paper with the place whe it should be cited in the one that he is reviewin Having ead the paper Fred g back through his notes and begins to write the review. He reorganizes his notes into a better structure for the review and then begins to write. When he reaches one of his notes he finds that he needs the actual reference of a paper. He looks at the context of the note to find where the paper came from so that he can tell the author where to look.

Figure 9 – Containment Tree

We present all of the notes for a given crayon in a list. Each note can be expanded or collapsed. For expanding and collapsing notes in the list we use the following levels of detail:

1. Note is represented by its label (which the user can change)
2. All leaves of the containment tree. These leaf images are presented in a flow sequence as shown in figure 10. This level gives key phrases from the full image as a means of understanding the note. The highlight strokes themselves are not shown, although comment notes are shown.
3. Direct parents of rectangles presented in step 2 or previous stages of step 3, as shown in figure 12. By organizing highlights, the user controls what these levels of detail will be. The highlight strokes and comments for any descendent rectangles in the tree are shown, but the highlight stroke for the displayed rectangle is not because it is visually redundant. This view level actually represents multiple levels of detail where the user can dynamically expand and collapse individual nodes of the containment tree.
4. The bounding box of all highlights and comments, figure 13.
5. The bounding box of the active window at the time of image capture.
6. The entire captured image.

Any of steps 3 through 6 may be omitted if their bounding rectangle is the same or a trivial extension of the lower level.

Animation of Note View Transitions

All of the node expansion and collapsing operations are animated to allow the user to easily follow highlighted information as it repositions from one view to the next. If a user expands Figure 10, the screen animates through Figure 11 to its final form in Figure 12. Without animation, the expanding and collapsing of nodes make it difficult for the user to follow highlighted segments. Since the layout of images in the views use a flow sequence, view changes can significantly change the position of information. The animation allows the user to follow pertinent information from one view to the next.

Animating the expanding/collapsing of the representation is problematic because there is no inherent structure to the underlying representation of the image itself. Before expansion only portions of the image are shown and they are not in their final positions. The problem is how to present the new material as summary material animates to its final location. Rather than a complex multi-fish-eye warping of the underlying image, we simply fade in the new material using transparency as shown in Figure 11. We then linearly interpolate the previously visible items to their final positions. For collapsing, the process reverses.

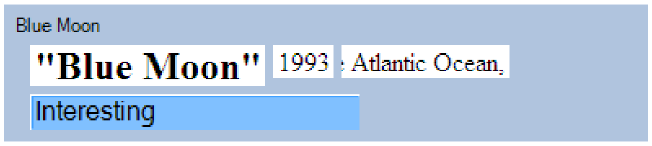

Blue Moon
"Blue Moon" 1993 Atlantic Ocean,
Interesting

Figure 10 – Leaf Images

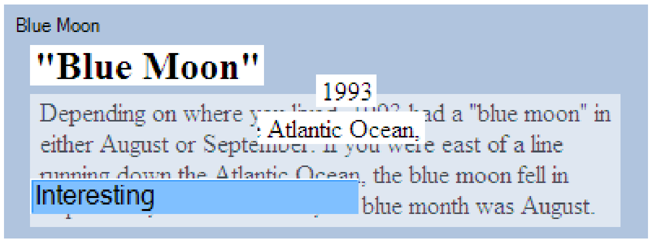


Figure 11 – Expanding Nodes

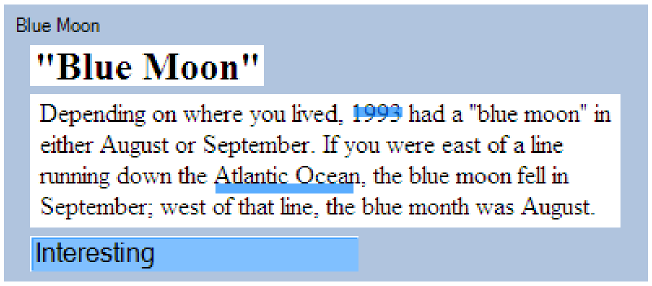
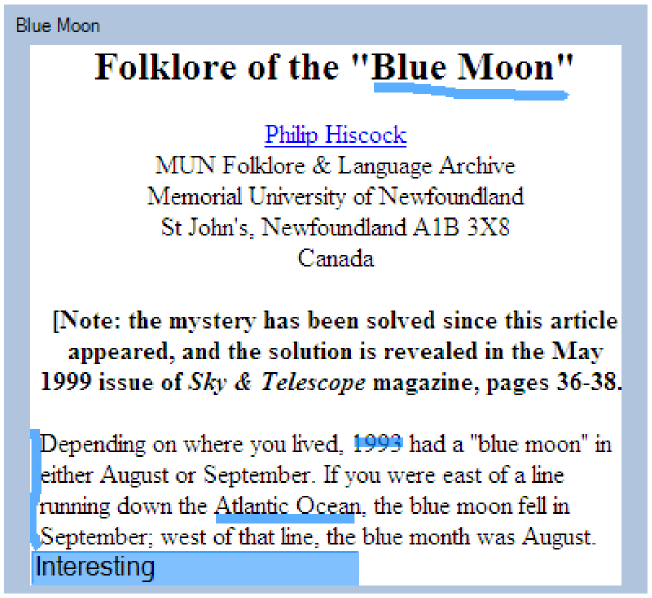

Blue Moon
"Blue Moon"
Depending on where you lived, 1993 had a "blue moon" in either August or September. If you were east of a line running down the Atlantic Ocean, the blue moon fell in September; west of that line, the blue month was August.
Interesting

Figure 12 – Expanded to Containing Region


Blue Moon
Folklore of the "Blue Moon"
Philip Hiscock
MUN Folklore & Language Archive
Memorial University of Newfoundland
St John's, Newfoundland A1B 3X8
Canada

[Note: the mystery has been solved since this article appeared, and the solution is revealed in the May 1999 issue of *Sky & Telescope* magazine, pages 36-38.

Depending on where you lived, 1993 had a "blue moon" in either August or September. If you were east of a line running down the Atlantic Ocean, the blue moon fell in September; west of that line, the blue month was August.
Interesting

Figure 13 – Expanded to All Highlights and Comments

USEFULNESS OF SCREEN CRAYONS

The key advantage of Screen Crayons lies in its simplicity and its universal applicability. A note is a screen image, highlight ink, note ink and text notes. These basic components can be used for virtually any purpose with at most 10 minutes of training. The crayon manager adds simple structure to the notes and the image summarization makes notes manageable with very little effort on the part of the user.

We can now revisit Fred the biologist. Creating a ToDo list while reading email is as simple as creating a ToDo crayon that is placed in the crayon box and always available. Adding an item to the list consists of grabbing the ToDo crayon, underlining the key part of the email message and saving. All of the visual contextual information about the item is implicitly saved and the highlighted information provides a brief summary for the item in the ToDo list.

Handling notes from the email, web page, journal article and budget spreadsheet are all trivial with no file format compatibility problems. They are all just images and all are managed in the same way. This includes Fred's special home-built gene sequencing program. Cross references among disparate documents and applications are handled by bringing all relevant items onto the screen, capturing a note, highlighting the key points and then using digital ink comments, circles and arrows to tie the points together for future use. The fact that multiple programs are being annotated is irrelevant to the image/note tools.

Creating notes is also spontaneous. When "off the wall" items like the Nematode Songbook or travel information appear, they can be rapidly captured under other crayon topics for later use. The notes become an "instant memory" tool. If no crayon exists for a topic the user can create a "Look at Later" crayon for such items. Later the user can review these notes, add more comments, move them to other crayons, email them to friends or throw them away. In the case of the Nematode Songbook, highlighting a song title can jog one's memory. However, because the entire screen was captured, opening up the image further will reveal the URL for that page as well as the scroll bar image that shows how far into the document the song was found. These are natural contextual cues that require no effort on the part of the user.

For extended tasks such as reading a thesis or reviewing a paper a separate crayon for that document can be created. This crayon might be placed under some global "Journal Reviews" or "Pat's Thesis" crayon for tidiness purposes but the crayon for the specific review task can collect all of those notes together. The image summarization capability allows those notes to be reviewed, reordered and reorganized using brief information with the expansion capability allowing recovery of more detail.

Managing little notes using full screen capture may seem like a waste of disk space. We did an informal sample of screen captures of 1600x1200 resolution screens of various topics. We compressed the images in PNG format, which preserves the exact image with all colors retained. PNG is smaller and more accurate than JPEG for these types of images. The average capture was about 300K in size. This means that more than 3000 notes will fit in one gigabyte. With a gigabyte of disk costing less than $15USD in 2003 it seems that space is not an issue.

Our implementation of ScreenCrayons is written in C# and runs under Windows.

REFERENCES

1. Adler, A., Gujar, A., Harrison, B. L., O'Hara, K., and Sellen, A. A Diary Study of Work-Related Reading: Design Implications for Digital Reading Devices. *Proc CHI '98*, 241-248.
2. Alias, Sketchbook Pro, http://www.alias.com/eng/products-services/sketchbook_pro/index.shtml
3. Baird, H.S., Background structure in document images. *International Journal of Pattern Recognition and Artificial Intelligence*, 8(5):1013-1030, 1994.
4. Bargeron, D., and Moscovich, T., Reflowing Digital Ink Annotations, *Proc CHI 2003*, 385-393.
5. Breuel, T.M., Two Geometric Algorithms for Layout Analysis, *Proceedings of the 5th International Workshop on Document Analysis Systems V*, p.188-199, 2002.
6. Brush, A.J.B., Bargeron, D., Gupta, A., and Cadiz, J.J. Robust Annotation Positioning in Digital Documents. *Proc CHI 2001*, 285-292.
7. Budzik, J., and Hammond, K. J., User Interactions with Everyday Applications as Context for Just-in-time Information Access, *Proc Intelligent User Interfaces 2000*, 44-51.
8. Conklin, J., and Begeman, M. L. gIBIS: A Hypertext Tool for Exploratory Policy Discussion. *Proc CSCW '88*, 140-152.
9. E-Quill, http://www.rocketdownload.com/details/HTML/5337.htm
10. Fails, J., and Olsen, D. A Design Tool for Camera-based Interaction, *Proc CHI 2003*, 449-456.
11. Francik, E., Rapid, Integrated Design of a Multimedia Communication System. In *Human Computer Interface Design*, M. Rudisill, et al., (Eds.), Morgan Kaufman, San Francisco, CA (1996).
12. Golovchinsky, G. and Denoue, L., Moving Markup: Repositioning Freeform Annotations, *Proceedings of ACM UIST 2002*, (2002).
13. Gotz, M. G., Schlechtweg, S., and Strothotte, T. The Intelligent Pen – Toward a Uniform Treatment of Electronic Documents, *International Symposium on Smart Graphics*, (June 2002).
14. Halasz, F. G. Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems. *CACM*, Vol 31(7), (July 1988), 836-852.
15. Marshall, C.C., Halasz, F.G., Rogers, R.A., Janssen, W.C., "Aquanet: A Hypertext Tool to Hold Your Knowledge in Place", *Hypertext '91*, ACM, (1991).
16. Marshall, C.C., Price, M.N., Golovchinsky, G., and Schilit, B.N., Designing e-books for Legal Research, *Proc International Conference on Digital Libraries (2001)*, ACM, 41-48.
17. Nematode Songbook, http://mgd.nacse.org/hyperSQL/squiggles/songs.html.
18. Olsen, D. R., Boyarski, D., Verratti, T., Phelps, M., Moffett, J. L., and Lo, E.L., Generalized Pointing: Enabling Multiagent Interaction. *Proc CHI 98*, 526-533.
19. Schilit, B.N., Golovchinsky, G., and Price, M. N. Beyond Paper: Supporting Active Reading with Free Form Digital Ink Annotations. *Proc CHI '98*, 249-256.
20. Wilcox, L., Schilit, B., and Sawhney, N., Dynomite: A Dynamically Organized Ink and Audio Notebook. *Proc CHI '97*, (1997).