SCREENCRAYONS:
USING SCREEN CAPTURES FOR ANNOTATION AND RESEARCH

by
Trent Taufer

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science
Brigham Young University
April 2007
This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date

Dan Olsen, Chair

Date

Phillip Windley

Date

Kevin Seppi
As chair of the candidate's graduate committee, I have read the thesis of Trent Taufer in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

Dan Olsen
Chair, Graduate Committee

Accepted for the Department

Date

Parris Egbert
Graduate Coordinator

Accepted for the College

Date

Thomas Sederberg
Associate Dean, College of Physical and Mathematical Sciences
ABSTRACT

SCREENCRAVONS:
USING SCREEN CAPTURES FOR ANNOTATION AND RESEARCH

Trent Taufer
Department of Computer Science
Master of Science

In a world full of digital information we should be able to easily collect, organize, annotate, and leverage information from many different sources. This should be easy to do and not interrupt our normal workflow. A system to support information collection and organization should be user-friendly and as unobtrusive as possible, while still allowing for flexible and intelligent annotation. It should also be able to leverage the inherent information content of a collection of annotated information. We present a system that will demonstrate how these ideas can come together to make information collection easier and more productive. The system facilitates collection, organization, and annotation of information using screen captures, and leverages the information content of the annotated collection to automatically summarize the information and find additional related information via searchable document repositories.
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. Annotations the user makes on the screen captures are automatically associated with regions of text that are then used to formulate queries to a search engine. The results of these queries are filtered and ranked based on their similarity to the original annotations. The system then presents links to these documents to the user. We also describe an experiment that shows that the documents found by the annotation system are generally found to be more relevant to a user's topic than the users own queries using the search engine.
# TABLE OF CONTENTS

**Chapter 1 - Introduction** ................................................. 1  
1.1 System Basics .......................................................... 4  
1.2 ScreenCrayons at Work ............................................... 6  

**Chapter 2 - Review of Previous Work** ..................................... 11  
2.1 Annotation ............................................................. 12  
2.2 Information Layout .................................................. 15  
2.3 Document Analysis Algorithms ..................................... 15  
2.4 Information Retrieval ............................................... 16  
2.5 Queries through Digital Annotations ............................... 17  

**Chapter 3 - Screen Capture and Annotation** ............................... 19  
3.1 Annotations in Image Space ....................................... 19  
3.1.1 The Pervasive Annotation Architecture Problem ............. 21  
3.1.2 Using Screen Captures ......................................... 22  
3.2 Highlighting and Commenting ..................................... 25  
3.2.1 Comments ....................................................... 27  

**Chapter 4 - Highlight to Region Association** .............................. 31  
4.1 Finding Regions ....................................................... 33  
4.2 Continuity Images .................................................... 36  
4.3 Associating Strokes with Regions .................................. 40  
4.3.1 Circles and scribbles .......................................... 40  
4.3.2 Underlines ...................................................... 41  
4.3.3 Horizontal highlights ......................................... 42  
4.3.4 Margin bars .................................................... 43  
4.4 Verification of Algorithm .......................................... 44  

**Chapter 5 - Organizing & Viewing Annotations** ............................ 47  
5.1 Organizing Annotations ............................................. 47  
5.2 Summarizing Full Screen Images .................................. 49  
5.2.1 Hierarchical Containment Trees ................................ 50  
5.3 Viewing and Animating Annotation Summarizations ............... 51  
5.3.1 Animation of Note View Transitions .......................... 54
# TABLE OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Information flow while finding information</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>ScreenCrayons’ annotation of a spreadsheet</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Highlighted Items</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>Crayon box and annotations</td>
<td>23</td>
</tr>
<tr>
<td>5</td>
<td>Before and after capture</td>
<td>24</td>
</tr>
<tr>
<td>6</td>
<td>Crayon box</td>
<td>24</td>
</tr>
<tr>
<td>7</td>
<td>Highlights and associated regions</td>
<td>26</td>
</tr>
<tr>
<td>8</td>
<td>Comment mode</td>
<td>28</td>
</tr>
<tr>
<td>9</td>
<td>Annotation regions</td>
<td>32</td>
</tr>
<tr>
<td>10</td>
<td>Annotation types</td>
<td>34</td>
</tr>
<tr>
<td>11</td>
<td>Natural boundaries of regions</td>
<td>34</td>
</tr>
<tr>
<td>12</td>
<td>Continuity image algorithm</td>
<td>37</td>
</tr>
<tr>
<td>13</td>
<td>Continuity image details</td>
<td>37</td>
</tr>
<tr>
<td>14</td>
<td>Continuity image graphical example</td>
<td>39</td>
</tr>
<tr>
<td>15</td>
<td>Finding regions for circles and scribbles</td>
<td>41</td>
</tr>
<tr>
<td>16</td>
<td>Regions for underlines</td>
<td>41</td>
</tr>
<tr>
<td>17</td>
<td>Regions for highlights</td>
<td>42</td>
</tr>
<tr>
<td>18</td>
<td>Regions for margin bars</td>
<td>43</td>
</tr>
<tr>
<td>19</td>
<td>Crayon manager</td>
<td>47</td>
</tr>
<tr>
<td>20</td>
<td>Hierarchical containment tree</td>
<td>50</td>
</tr>
<tr>
<td>21</td>
<td>Label view of annotation</td>
<td>51</td>
</tr>
<tr>
<td>22</td>
<td>Leaf view of annotation</td>
<td>52</td>
</tr>
<tr>
<td>23</td>
<td>Expanded view of annotations</td>
<td>52</td>
</tr>
<tr>
<td>24</td>
<td>Combined view of all annotations</td>
<td>53</td>
</tr>
<tr>
<td>25</td>
<td>Bounding box of active window</td>
<td>53</td>
</tr>
</tbody>
</table>
Chapter 1 - Introduction

The fundamental metaphor for the modern office workstation has always been paper. Beginning with the design of the Xerox Star, windowing systems 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 personal computers, creating documents was hard (typewriters are not error-friendly). Distributing paper documents was hard. Modifying paper documents was hard. Annotating paper documents by marking on them was easy. Because of this disparity of labor, the focus of most computer based 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 documents received much less attention despite its importance in actual day to day use.

The advent of very cheap storage, cheap communication via the Internet, standard document formats such as PDF or HTML and the pervasive availability of computers has caused a shift in our usage of documents. For an increasing number of people, a large portion of their reading experience is in a digital format 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 [ADL98] have reported that reading occupies 70% of document-related activity. However, for many people a substantial amount of reading time occurred in conjunction with writing. In the same study, creation and updating of documents constitutes only 18% of time spent writing while people are reading. On the other hand, annotation and
note taking consume 48% of the time. Schilit describes this as “active reading” [SCH98] 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 “active reading”.

During “active reading,” people annotate, summarize, and highlight by marking directly on the paper they are working with. The marks they make are used to indicate important information, cross-references, and supplementary notes. These annotations simplify future use of information and support organizing thoughts and ideas related to the task at hand. The concept of reading and annotating is ubiquitous and independent of the contents or source of the paper.

This same annotation metaphor is very useful in the digital world. Much of the information we deal with is currently in digital form. In order to apply traditional annotation techniques to this information we must print it. By pulling the information out of the digital world we lose many of the inherent advantages of a digital medium. The primary advantage we lose is that any marks made on the paper printout are unavailable in the digital world. These annotations and notes contain information that can be used to our advantage. Among other things, we can use this meta-data to summarize the information for future referencing as well as to find related information. By incorporating these annotations into the digital world, we can more easily use this information to its full potential.

One way to utilize annotations is by using them to find related information. Such research can be anything from a 2nd grader finding information about George Washington for a report, to a physicist delving into the inner workings and theoretical background of a
newly developed nuclear reactor. This process consumes a great deal of time [IST02]. We must look for information, read the information to find the relevant parts, and mark the important portions of the information. These steps can be time consuming and finding information that is relevant can be difficult. Much of this searching is performed using various search engines with keywords, but finding exactly the right keywords to use can be difficult.

The finding and marking steps in this process are generally treated as two separate but related tasks. The flow of information is primarily in one direction: information is found, then read and annotated, and then relevant portions are utilized in the task at hand (See figure 1). Unfortunately this unidirectional flow leaves out an important source for finding more information: the reader’s markings and notes. These annotations indicate the relevant portions of the information and are generally used to speed up referencing the information at a later time.

The system we will present, called ScreenCrayons, is used to annotate information presented on screen. The system is based on screen captures in order to eliminate the need to interface with each and every application. ScreenCrayons can capture and manage annotations for a variety of tasks using any application. It is able to organize the collected information in an easily accessible manner and provides automatic summarization of the
annotations as well as intelligent retrieval of related information. By integrating both reading and annotating into the digital world we simplify the tasks of reading, researching, and annotating while at the same time provide extra features to the user to assist them in their work. The goals for the system are for it to be very flexible, easy to use, lightweight and widely applicable.

1.1 System Basics

ScreenCrayons incorporates a number of important ideas in order to reach these goals. The first key idea is that the system is based entirely on screen captures. The beauty of this is that absolutely anything the user can visualize on the screen can be captured and

![ScreenCrayons' annotation of a spreadsheet](image)

Figure 2: ScreenCrayons’ annotation of a spreadsheet
annotated. Screen captures allow for taking notes on any visible element, from simple text to complex graphics and photos. This gives the user the freedom to use the applications they choose, yet still gives them the power to be able to easily create and manage annotations. This allows the user to easily integrate digital note taking into their everyday tasks. They are not tied to a particular application or even a specific file format when they want to annotate information. They are free to take notes whenever and wherever it is convenient and necessary in the course of their work. We will demonstrate the many freedoms that this concept presents, as well as some of the tasks that screen captures make more difficult in the context of this system.

Because the notes captured by this system are the size of the entire screen, it can be difficult to view multiple notes at once. ScreenCrayons overcomes this difficulty by providing multiple levels of summarization based on the users’ annotations and the context of the notes within the windowing system. This allows for easy viewing of multiple notes at once, while still retaining all of the full screen contextual information that is no more than a few mouse clicks away.

Another ScreenCrayons feature is the ability to use the captured notes and annotations to find other sources of related information. Since we know what parts of the information the user finds relevant based on their annotations, we can use the annotated information with surrounding context to perform searches. ScreenCrayons incorporates the information it finds into the users’ normal flow of reading, finding, and annotating.
1.2 ScreenCrayons at Work

Let’s examine how ScreenCrayons can fit into someone’s daily work activities. Thomas is an office manager for a research company. When he gets to work he opens his email and notices that he has received a detailed specification and price sheet for the copier he is planning on buying for the office. He reads over the document and uses ScreenCrayons to take notes on what features he thinks really make this copier stand out. He has also received an email from one of his researchers with office party ideas. Thomas marks some of the ideas on the list that he thinks would be viable.

His inbox also contains a preliminary 3rd quarter budget for his department. He opens the file in his spreadsheet program and marks of few of the items that might be problematic for next quarter. Also, one of the lead researchers in his department has sent him an email about the last set of runs with the stress testing equipment that he needs to review. The researcher thinks the equipment may need to be upgraded, so Thomas takes note of this for later review.

After checking his email Thomas brings up an interoffice memo he was working on yesterday. The memo reminds the employees about the company's policy regarding sensitive information. The memo covers everything from giving out computer passwords to discussing upcoming product launches that haven't been officially announced. He makes a few notes on the memo for items he needs to finalize before he sends it out.

Thomas heads off to the weekly project status meeting to review all the current and upcoming projects. He grabs a copy of the presentation off the network and takes some notes about an upcoming project that will require a heavy-duty winch in the machine lab. Thomas
also remembers that his wife wanted him to look into getting a new microwave, so he also adds a note to ScreenCrayons to remind himself.

After returning to his desk Thomas looks through his notes in ScreenCrayons and decides to examine the results of the last stress test. He brings up the custom application that displays the results and agrees that the equipment will need to be upgraded. He captures some notes on the results so that later he can start pricing out the equipment upgrade.

Thomas also notices that ScreenCrayons has found more information related to the office party. One of the restaurants recommended for the office party has a group discount available that he wants to look into. There is also some information related to the copier he wants to buy. It is a specification sheet for an updated model that he should look into before making a final decision. ScreenCrayons also shows a few web links to companies that sell winches that might work in the machine lab.

There are several points that this example illustrates about note taking and how notes are used.

1. Notes occur spontaneously during work and are not always related to the current task, like Thomas’ note about the microwave.

2. Notes occur in the context of many applications. In our example Thomas used email, a word processor, a spreadsheet, a web browser 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. They also indicate what the user
thinks is important or relevant.

4. Notes are frequently a source for later writing and/or referencing as in the notes on the interoffice memo.

5. Notes can refer to tasks where additional information can be useful, such as the needed winch.

In thinking about annotations we find 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. Notes should be easy to store, as well as easy to reference.

5. The resulting notes must be condensed and summarized so that they can be easily browsed and manipulated later.

6. We must be able to use the notes to find related information automatically.
The goal of ScreenCrayons is to provide a lightweight, universal, note taking facility that satisfies all of these needs without interfering with other work. We will show how ScreenCrayons addresses these ideas to help users capture information, take notes, and find information related to the notes they have taken.
Chapter 2 - Review of Previous Work

ScreenCrayons relates to a number of different research areas. Digital ink and annotation are probably two of the most applicable topics since ScreenCrayons is a digital annotation system. Research in these two areas has focused on leveraging the advantages of digital media to annotate, organize and categorize information. We will review some of these digital annotation systems and discuss their strengths and weaknesses as they relate to the goals we have for a lightweight, universal annotation system. ScreenCrayons also uses information layout to present the user with summarized notes and easy viewing of annotations. Information layout and information visualization are both very broad topics, but we will focus on reviewing two areas that relate to ScreenCrayons: viewing subsets of visual information, and visualizing data that is organized in a tree structure.

We also need to examine document analysis research. ScreenCrayons has the capability to automatically associate ink strokes with regions of the underlying image. This is critical in order to create note summarizations. Without these regions we must rely solely on the users’ ink strokes, which are normally imprecise and do not indicate the entire region of interest. Because ScreenCrayons also searches for related information, we will discuss some of the research that has been done in information retrieval. From this body of work we will focus on document retrieval systems that use keywords as well as contextual words to perform searches. We will also examine a specific area of information retrieval that uses digital annotations as a source of information to perform queries.
2.1 Annotation

Many researchers have tackled different aspects of the digital ink and annotation problems. One system that has demonstrated many annotation techniques and ideas is XLibris [GOL02, GOL99, MAR01, SCH98]. XLibris is an annotation system built on a pen tablet display that mimics paper documents by viewing entire pages at a time and giving the user visual cues to indicate their location in a document. XLibris has also been used to demonstrate user annotations for making queries and finding related information. One of the major drawbacks to the XLibris system is that it requires documents to be imported into its proprietary data format, normally through printing or scanning. This limits where XLibris can be used because many sources of information cannot be easily printed or scanned. Another drawback is that all annotation occurs in the XLibris system. This limitation makes it difficult to use the system spontaneously in the course of a user’s normal work.

Others have also looked at ink reflowing [BAR03, BER01] in terms of user expectations and feelings about the reflowed ink. This particular system was based on Internet Explorer, and therefore used HTML text as the underlying data type. This allowed for an editable text document that could be reflowed to test repositioning of annotations. They showed that users only approve if the reflowed annotations are moved and realigned appropriately. Since ScreenCrayons doesn’t need to support documents that are in flux, we don’t need to store the actual text data. By not limiting ourselves to text data, we allow the user to be able to annotate anything from any source.

Another aspect of annotation is how users can use different colors to simplify and facilitate annotation. Digital ink has been used for simplifying the interactive machine learning process. Image Processing with Crayons [FAI03] used digital annotations to support
quick and robust classification of digital imagery. Users can use quick digital ink strokes to feed information into machine learning algorithms for classifying imagery. Each color of ink represents a different classification. Also, the use of different colors and styles of pens [GOT02] has also been examined and has shown that it makes it easier for users to differentiate between annotations. In ScreenCrayons we continue with a similar metaphor of colored crayons or “concept categories” to annotate and also to categorize annotations. This helps the user to organize and more easily reference notes at a later time based on category names and colors.

There have been several projects to support the annotation and organization of information. Some systems such as Notecards [HAL88], gIBIS [CON88] and Aquanet [MAR91] 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 available 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 effort at the
time a note is taken. ScreenCrayons supports this idea by capturing entire screen shots, with all contextual information that is available on the screen.

An important early annotation system was the Dynomite project [WIL97]. The system combines recorded audio with the users annotations, that can be linked together through time. The specific time that an annotation is drawn automatically links it to the audio that was recorded at the same time. The digital ink captures the user's intent and comments, and the audio captures 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 seek to mirror this property with image data rather than audio information.

Most annotation systems, including those we have discussed here, only support annotations of specific artifact/document file types. The Watson system [BUD00] 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 difficult to learn new, mutually incompatible note systems for each application. 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.
2.2 Information Layout

Visualization of information has been looked at from many different aspects. Some have covered different conceptual ideas on the topic [CAR96, CAR97]. These presented many different ideas from research as well as commercial applications. Others have covered topics related to viewing information like using focus plus context for viewing an entire set of information while still being able to view details of specific portions of the information with ease. The Document Lens [ROB93] is one method for viewing large amounts of highly detailed information. The user can get a low detail overview of all of the information, while still having access to detailed views of small portions of the information. ScreenCrayons leverages this same idea, but extends it to automatically determine which of the “small portions” of information to display to the user.

Also, ideas for viewing trees with varying degrees of zoom have been explored with degree of interest trees [CAR02, NAT02]. These two approaches of focus plus context and degree of interest trees relate directly to our task of being able to organize and summarize the annotations that users make. ScreenCrayons organizes the users annotations into a tree structure that provides the user an expandable view of their annotations.

2.3 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 [BAI94]. 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 [BRE02] as well as techniques for identifying large rectangular blocks of background.

We moved away from such techniques for several reasons. The first was 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 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.

2.4 Information Retrieval

In information retrieval one area of interest applicable to ScreenCrayons is the idea of using as much contextual information as possible to do retrieval [BUD00, WOO00]. Instead of just using search terms, these systems use large portions of text and even entire documents to find other related documents. These systems have shown that by using more
than just keywords, we can find information that is much more useful and applicable. ScreenCrayons uses a hybrid technique that leverages widely available keyword based document indexes, then performs contextual comparisons between documents and annotations to present the user with more applicable results.

The area of information foraging [PIR95, PIR98, PIR97] takes a biological and anthropological approach to information theory. They incorporate the predator-prey relationship into finding information and determining the quality of the information. In this way you can create a balance between continuing to search in the same source and moving to a new source. This would be useful to the ScreenCrayons system in the future if it were expanded to use multiple information sources such as multiple search engines and/or documents on the users local system.

2.5 Queries through Digital Annotations

XLibris [GOL99] is one system that has used digital ink annotations to perform queries and find information. This research focused on comparing queries derived from annotations, to queries that are refined based on the users relevance ranking of documents. They showed that queries derived from a users annotations produced much more relevant results. The system used the InQuery [CAL94] search engine for performing searches which supports weighted sum queries. ScreenCrayons also uses annotations to find related information that can be automatically presented to the user.

ScreenCrayons provides a lightweight, universal annotation system that extends and expands the work of prior annotation systems. It integrates many features that will help the user have a better and more useful annotation experience. Full screen images can be
summarized for easier viewing, while still having access to full contextual information. Also user’s annotations can be leveraged to find related information to present back to the user. These features provide a system that supports the user in the tasks they set out to accomplish, while not impeding them from accomplishing those tasks.
Chapter 3 - Screen Capture and Annotation

There are a number of goals we are trying to accomplish with ScreenCrayons. We want note taking and annotation to be very lightweight so that it interrupts the user as little as possible. Also the annotations should retain as much contextual information as possible from the user’s work environment. In order to support these two goals, we have built ScreenCrayons to use screen captures as the annotation medium. By using screen captures we allow the user to create annotations on an exact copy of their workspace. They do not have to transition to a separate environment to capture their notes. Screen captures also help to retain the user’s context by capturing everything in the user’s workspace at the time of capture. Any data or application visible on the screen when the user creates a note, will be preserved in the screen capture.

3.1 Annotations in Image Space

The annotation portion of ScreenCrayons allows for arbitrary digital ink strokes to be drawn on screen captures. The user can draw on the screen captures using the mouse to indicate items of relevance. This can be seen in figure 3 where three separate items have been highlighted. In addition to digital ink, the system will also use plain text notes so users can type comments if they desire. At the time of capture each annotation will be associated with an appropriate concept category for easy retrieval and referencing. The concepts categories allow related annotations to be placed together.
In order to allow the system to be used on any computer system and with any application, the system uses screen captures to collect information to be annotated. This approach makes it so that the only real link that ScreenCrayons needs into the operating system is a way of retrieving a screen capture of the current workspace. The system doesn't need to know anything else about the specifics of an operating system or architecture. Also, the system doesn't need to know anything about the applications it is capturing information from. It will work with the most common office applications as easily as it will work with custom-built applications.
ScreenCrayons must also be unobtrusive during normal computer use by either residing in a system tray icon or a small window that can always be on top of other windows. The user only needs a way to capture the current screen and select a crayon category. This provides an easy to use interface that is very lightweight and won't interfere with the users normal computer use.

3.1.1 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 implemented. 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 downside of this approach is getting all interesting applications to conform to the protocol.

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 of screen captures is that some of the non-visible context is lost. This would include portions of a document currently scrolled out of sight.

3.1.2 Using Screen Captures

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
3. adding optional commentary to indicate what is important about this note.

Consistent with the work done with Image Processing with Crayons [FAI03] 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 [GOT02].
These crayons are also similar to the categories in Dynomite [WIL97]. 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 4. This "scribble on the screen" metaphor is trivial to learn and completely independent of any application.

![Figure 4: Crayon box and annotations](image)

Note capture is implemented by performing a screen capture. The Freestyle [FRA96] system and Alias Sketchbook Pro [ALI] 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 (see figure 5). 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 as seen in figure 6. 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, an image from 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.

### 3.2 Highlighting and Commenting

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 figure 7. 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.

Figure 7: Highlights and associated regions

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.

### 3.2.1 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 8. The masked area provides blank space where comments can be written.
Comments are simply digital ink used 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.

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 [OLS98]. 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.
Chapter 4 - Highlight to Region Association

The next step is for ScreenCrayons to be able to associate each user provided annotation with a region of the screen capture. This is necessary for two reasons. The first is to be able to summarize the annotations in a meaningful way. The second reason is so that text can be extracted from the annotations to be used for finding related information. Because of this, ScreenCrayons must be able to infer structure from screen captures. This is necessary in order to determine what information in the image each annotation corresponds to. The structure must be determined with no additional information other than the users annotations and the image. The algorithm must also be robust enough to deal with rough annotations since users will frequently not be precise when creating annotations. ScreenCrayons deals with underlines/highlights, and margin bars in special ways, while considering all other ink strokes as bounding regions. Based on the annotations, ScreenCrayons must be able to associate a corresponding region to each annotation, whether the region is a word, paragraph, image, or something else. In figure 9 we see where each of the four annotations has an associated region represented by a box. These associated regions will be used for summarization as well as extracting text.
Figure 9: Annotation regions

We will group highlights into four different categories to facilitate region association.

The first classification of highlight is an underline stroke, which is generally characterized by a horizontal stroke, that is in an area of continuous color or whitespace. These strokes are placed under an area of interest, like under a word or phrase. The second classification is a horizontal highlight, which is also characterized by a horizontal stroke, but is placed in an area of changing color or non-whitespace. These strokes are normally placed directly on top of an area of interest. The third classification is margin bars which are vertical strokes, normally placed in areas of continuous color or whitespace. These strokes can be placed next to areas of interest, such as paragraphs or even images. The final classification is circles and
scribbles, which encompasses anything not grouped under the first three categories. These annotations are very freeform and can be placed almost anywhere.

Our basis for finding the appropriate regions is that they should be surrounded by natural boundaries. In the case of words, the boundaries are spaces to the sides, and line breaks to the top and bottom. Paragraphs have margins to the sides, and line breaks to the top and bottom. These natural boundaries are normally sections of a continuous color, patterns of low contrast, and/or gradients [BRE02].

In order to verify that our region association approach is valid, we will test our algorithm against a series of annotated screen captures that have been manually assigned regions. With this data we can compare the regions that our algorithm has found to the human specified regions to verify the accuracy and validity of the algorithm. It should be noted that there is no exact region match for each ink stroke, therefore in our tests we will allow the algorithms results to vary by a few pixels from the human specified region, while still counting it as correct.

4.1 Finding Regions

Each highlight mark that the user makes is associated with some region of the screen capture. 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 [GOL02] and later Bargaron [BAR03] identify five types of marks that people make on paper. They are: circles, underlines, highlights, margin bars and marginalia (see figure 10). 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 view 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 11. Our basis for such natural boundaries is that most applications use long runs of "uniform" color to visually segment their presentations of information. 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.

Starting from the bounds of the annotations we can search up, down, left, and right to find the appropriate boundaries. This boundary finding will be done using continuity
images. Continuity images provide a way to find long runs of continuous pixels that are all the same or similar colors. This allows for solid color boundaries to be found as well as boundaries that consist of low contrast patterns and gradients. Each screen capture has separate horizontal and vertical continuity images that are calculated by iterating over the image separately in the x and y directions. At each location in the continuity image is a value of how many continuous pixels have been seen in a row (or column for the vertical image.) This allows for easy searching over the image space to find the continuous runs that should be the boundaries for each annotation.

4.2 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 can be seen in figure 12. The algorithm loops over each pixel from top to bottom and left to right. At each pixel in the image, the difference between the pixel and the pixel immediately to the left is calculated. If this difference is less than a specified threshold, then the pixels are considered part of the same run of pixels. If the pixel is part of the same run, then the value in \texttt{HCONTINUITY} is incremented, otherwise the value is set to 1. The data stored for each pixel in \texttt{HCONTINUITY} represents how long of a run of pixels exists to the left of the pixel in question.
The nature of the continuity is defined by the difference function and its threshold. Our implementation uses a difference function that is the maximum of the absolute values of each of the individual red, green, and blue differences between pixels. Our empirical trials found a threshold of 55 in a 0-255 RGB space to perform well in detecting runs of pixels. 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.

```plaintext
forall (X and Y in IMAGE)
  if (X == 0) HCONTINUITY[X,Y] = 1
  else if (difference(IMAGE[X,Y], IMAGE[X-1,Y]) < threshold)
    HCONTINUITY[X,Y] = HCONTINUITY[X-1,Y] + 1
  else
    HCONT[X,Y]=1;
```

**Figure 12: Continuity image algorithm**

The nature of the continuity is defined by the difference function and its threshold.

Our implementation uses a difference function that is the maximum of the absolute values of each of the individual red, green, and blue differences between pixels. Our empirical trials found a threshold of 55 in a 0-255 RGB space to perform well in detecting runs of pixels. 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.

<table>
<thead>
<tr>
<th>Image:</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuity Image:</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

**Figure 13: Continuity image details**

As figure 13 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. You can see in figure 14 an example of an image with corresponding graphical representations of the vertical and horizontal continuity images. The numbers from the continuity images are represented by pixel intensity, the brighter colored pixels represent higher numbers in the continuity image.
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 occur 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.

4.3 Associating Strokes with Regions

Having classified the highlight strokes and computed our continuity maps we now can compute the rectangular region associated with each stroke. Our algorithm searches up, down, left, and right to find natural boundaries that correspond to each ink stroke. We use the continuity images to locate potential natural boundaries. The continuity images help us find continuous runs of low intensity contrast. Associating strokes with regions has four cases that we have discussed previously: 1) area marks such as circles and scribbles, 2) horizontal underline, 3) horizontal highlight and 4) margin bars.

4.3.1 Circles and scribbles

Finding the corresponding regions of circle and scribble strokes is the easiest. We start with the bounding box of the ink stroke. 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 15, we search in each direction from the ink stroke's bounding rectangle to find the region bounds. We start by finding the left and right boundaries. To find the left boundary we start from the left edge of the ink bounding box where we check the vertical continuity image to see if there is a run of pixels that runs at least 98% of the total distance between the top and bottom of the ink stroke bounds. We repeat this for the right side of the ink bounding box.
Finding the top and bottom boundaries is very similar. We start from the top of the ink and search up in the same fashion as we did for the sides. Instead of using the vertical continuity image, we use the horizontal continuity image since we are looking for horizontal runs of pixels. We do the same thing for locating the bottom boundary of the region.

4.3.2 Underlines

If a horizontal stroke is in a boundary region of the same or greater length, as in figure 16, then we assume the stroke is an underline. From the vertical center of the ink stroke we search up to find the bottom boundary and then search up again from that boundary to find the top boundary. For finding the bottom boundary we use a technique similar to that described for circles and scribbles, but instead of searching for the first run we find, we actually want to locate the first horizontal band that doesn’t have a run. This is because we are starting in a horizontal boundary region, and we are trying to locate the bottom of a region to be selected. We continue searching up to locate another continuous horizontal run which becomes the top boundary of the region.
To find the side boundaries we start by shrinking the length of the highlight to give starting points for the vertical boundaries. This helps deal with underlines that have extended beyond the range of the region that is being selected. We use the space between the top and the bottom bounds as our predictor for the height of the vertical boundaries. We then search left and right to find the side boundaries just like we did with circles and scribbles.

### 4.3.3 Horizontal highlights

Horizontal ink marks that are not in a boundary area are assumed to be highlights, as in figure 17. We start by searching 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 as we did with underlines. The process is very similar to the underline algorithm, except we are not below the region in question, so we must search down to find the lower bounds.
4.3.4 Margin bars

Margin bars are also 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 18, we see that the center of the ink stroke is closer to the region on the right. This right boundary of the margin area 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 with the other stroke categories. A similar left searching approach can be used if the ink is closer to the material on the left.

Another case we must deal with is if the margin bar is placed on top of the item of interest. If this situation is detected, then we just search left and right to find the region bounds. We can easily detect this situation if the margin bar is not in an area of continuous pixel runs.
4.4 Verification of Algorithm

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 the regions that our algorithm found. 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 are 4 boundaries (top, bottom, left, and right) 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.
Chapter 5 - Organizing & Viewing Annotations

Organizing and viewing annotations are both critical pieces of ScreenCrayons, without them there is no way to refer back to notes that have been captured. Our goal with note organization was that notes should be easy to store, and also easy to reference. This supports the idea that ScreenCrayons should be lightweight and unobtrusive. Since notes are entire screen shots we also wanted to make sure that we had a reasonable way to view a condensed and summarized view of the notes, so that more than one note can be viewed on screen at a time. We will describe the techniques that we use to support these goals.

5.1 Organizing Annotations

Being able to capture and annotate information from applications is only part of the problem. In addition to this the user should be able to easily find information that they have annotated. In order to facilitate this, ScreenCrayons allows the user to create a hierarchical structure of concept categories. Each concept category has an associated annotation color and is referred to as a “crayon.” When annotating a screen capture the user selects a crayon (concept category), which sets where the annotation will be stored as well as the color and size of the annotation ink strokes. By using a hierarchical structure the user can place annotations in categories and subcategories to organize them. This allows for easy retrieval and referencing of annotated information. It also makes it easy to store related bits of information in close proximity in the hierarchy.
Once notes are created they must be browsed, augmented, discarded and organized if they are to become useful. The crayon manager, shown in figure 19, provides a simple tree structure for organizing both crayons and their notes. The crayon manager provides the interface for creating new crayons, assigning them names, 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.
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 [SCH98].

5.2 Summarizing Full Screen Images

One down side to using screen captures to retrieve information from applications is that they inherently fill the entire screen. This can complicate viewing the information that has been annotated. In order to view a screen capture at full quality, it requires the use of the entire screen. The only way to view multiple screen captures is to scale them so that more than one can fit on the screen at a time, but this reduces their visible quality [CAR02, PIR97]. By summarizing the screen captures, the system can effectively display multiple annotations at the same time without scaling. This way all of the contextual information is still available so that the user can view all information related to the annotations, while still hiding it when they don't need it.

The key to our note summarization technique is the selected regions associated with highlight strokes. Since the annotations should be indicative of what the user thought was important, we should be able to focus on displaying the annotated information. We take these regions and form a hierarchical containment tree. This provides summarization capability along with the ability to still view the entire screen capture if desired. One region is contained by another if most of its space is within the larger region. We require that most
rather than all of the rectangle be contained to allow for user sloppiness. In addition to the regions associated with highlights, we can also use other information pulled from the windowing system to augment the levels of the containment tree. This includes the window boundaries of the active application.

5.2.1 Hierarchical Containment Trees

In figure 20 we see an example hierarchical containment tree. The top part of the figure is a set of sample annotations with their corresponding regions that have been labeled A through E. The lower portion of the figure shows how the annotations fit in the tree. Each region that is mostly contained by another region becomes a child of that region. By organizing the regions into the tree we can then summarize the annotations by only viewing portions of the tree. By default the leaf nodes of the tree are displayed, but by expanding a child node the user can view parent nodes. This allows for views consisting of all the lowest tree level regions being displayed, as well as views that consist of the entire screen capture, with all the other possible views in between. The user is able to selectively expand and contract parts of the containment tree to view the annotated screen images. The transitions from one view to the next are animated to make it easier for the user to follow annotation regions as they change positions from one view to the next. The regions are displayed using a flow layout to show each expanded tree node.
5.3 Viewing and Animating Annotation Summarizations

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

1. Note is represented by its label (which the user can change) as seen in figure 21.

   ![Figure 21: Label view of annotation]

2. All leaves of the containment tree. These leaf images are presented in
a flow sequence as shown in figure 22. 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.

![Figure 22: Leaf view of annotation](image)

3. Direct parents of rectangles presented in step 2 or previous stages of step 3, as shown in figure 23. 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 contract individual nodes of the containment tree.

![Figure 23: Expanded view of annotations](image)
4. The bounding box of all highlights and comments, as seen in figure 24.

![Figure 24: Combined view of all annotations](image)

5. The bounding box of the active window at the time of image capture as seen in figure 25.
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.

5.3.1 Animation of Note View Transitions

All of the node expansion and contraction 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 22, the screen animates through figure 26 to its final form in figure 23. Without animation, the expansion and contraction of nodes make it difficult for the user to follow highlighted segments. Since the layout of images in the views use a flow layout, 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 expansion or contraction of the view is problematic. This is because there is no inherent structure to the underlying image. 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 any new view elements as shown in figure 26 with the large text block in the background. We also linearly interpolate the previously visible items to their new positions, as we can see with two of the elements in figure 26. For contraction the process reverses, so view elements that are leaving the view fade out and other elements smoothly transition to their new locations.

ScreenCrayons’ method of organizing notes in a hierarchical tree structure that is tied to crayons, makes note organization simple yet flexible. Users can easily setup crayons to represent concept categories that automatically categorize notes in the correct location when they are used to annotate. Notes that consist of full screen captures can be easily viewed by utilizing the information focus provided by the users annotations. The users annotations inform us what the user thinks is important in a note so we can focus on that, while still allowing the user to view the context of the note by expanding it’s view.

Figure 26: Animating note transition
Chapter 6 - Finding Related Information

One of the goals of ScreenCrayons was the ability to utilize the users notes and annotations to search for additional information. Lets look back at Thomas the office manager from section 1.2 and see how ScreenCrayons supported him in his work by finding related information. Once Thomas had created annotations he could come back to ScreenCrayons to view those notes. ScreenCrayons also had web links to additional documents available for Thomas to view, including links to a new model of a copier he planed on purchasing. We will show how ScreenCrayons finds and presents this additional information.

6.1 Overview

Using text extracted from the screen captures, ScreenCrayons can intelligently retrieve related information. The most important part of finding related information is finding the words and phrases in the screen captures that will be the most likely to separate related information from unrelated information [BUD00, WOO00]. Our approach to word importance is based on word frequency information from the annotations, screen captures, and the world in general. For example, if a word or phrase is found frequently in the annotations or the screen capture, but not found very frequently in the world in general, then it is probably a good candidate for a search term. Words in the screen captures with frequencies similar to the world in general are not as likely to be good at discriminating between related and unrelated information. In addition to this, the words found in annotations will have a higher importance than the surrounding words from the screen captures [GOL99].
These search terms are used to integrate with Google to provide related information. We use the search engine results as a preliminary source, then using information retrieval techniques we further filter and match the documents provided by the search engine to give the user the best matches possible [VEL97]. The initial "keyword search" is not powerful or explicit enough so we provide this additional level of filtering before presenting the results to the user.

The interface for presenting related information to the user is also important. The interface must present the user with the information without causing a significant interruption to their flow of work [BRA96, RAO95, SEB99]. The search for related information will occur in the background, and then must be presented in an appropriate fashion to the user when it is available.

The process of finding related information consists of a number of steps. First of all we need to have notes available that the user has annotated. Using the screen shot image and annotations associated with each note, we will show how we perform the necessary steps to assist the user in finding related information.

The process begins by first associating each annotation with a corresponding region in the screen capture, which we have already described in detail. Any text in the associated region is extracted and a list of all words and word pairs is created. Each item in the list is ranked according to its importance in the annotations and its frequency of occurrence in normal text. The highest ranked words and word pairs from the annotations are then used as keywords to search for related documents using Google. These documents from the
queries are ranked according to how similar they are to the original annotations and the most similar documents are presented to the user.

6.2 Extracting Text From Images

The first step in the searching process is getting the text out of the screen capture. Once we have the appropriate region associated with an annotation the text must be extracted. Since the notes are based on screen captures we must be able to recognize the text in the image. This task is somewhat simplified when compared to traditional methods where text is extracted from scanned or photographed images since direct screen captures have no noise and no problems with rotation and skewing. One difficulty is the need to be able to extract text from screen captures with non-uniform backgrounds such as textures and gradients like those found on some web pages.

We use a third party OCR package called Textract to recognize the text in annotations as well as the entire image for use as contextual information. We decided on this approach after trying to write our own OCR engine. Even though we could limit the engine to only support machine draw text from a screen capture, are attempts where not very accurate. The first reason for this is that many computers use antialiased text, which means that two identical characters of the same size, font, and color, will not always be represented on the screen in exactly the same way. This makes it difficult to consistently and reliably recognize characters. The second reason for this was mentioned previously; many screen captures have non-uniform backgrounds and textured images, which can make it difficult to separate text from background. Since the focus of this project is not to develop a robust
text extraction system we decided it would be acceptable to use an external toolkit to supply the text.

Once we have text extracted from the image we take each word and each word pair from the annotation regions and assign each a value according to how it was annotated. Focus selections like underlines and highlights receive 4 points, and passage selections such as margin bars and regions receive 2 points. All remaining contextual information from the image receives 1 point [GOL99].

6.3 Finding Important Words

After extracting the text from the annotations, we need to determine which words and word pairs are the most important. This is necessary because we are using Google which requires keywords as input, as opposed to a system that allows weighted word scores as queries. Every word score from the previous step is multiplied by the negative log of its frequency in a standard corpus. This gives words that are generally more common lower importance scores, and words that are more unique higher importance scores. We multiply by the negative log in order to decrease the range of the importance values. In the case of word pairs, we multiply the frequencies of the two words together for input into the equation. This gives them higher importance levels than the words individually. This is necessary because word pairs are more descriptive that individual words when it comes to finding related documents.

This gives us a list of words and word pairs with corresponding “importance” scores based on occurrence in the annotations and the overall frequency of the word in the world in general. We choose the highest ranked words and word pairs to use for input into the
search engine. In our implementation we have used the top 3 keywords from the list, but this could be extended to any number to extend the possible amount of information found. By increasing this number we increase the number of potential documents that can be found, but we also increase the number of searches that must be done as well as increasing the number of documents that must be processed and filtered.

6.4 Search Mechanism Connectivity

Once we have search terms we can feed them into our search engine. We used the publicly available Google Web API [GOO]. We perform a number of searches using our keywords. First we search with each of the keywords individually, and then we search with each keyword paired with all other keywords. Since we are using 3 keywords this results in a total of 6 searches. Remember that our keywords can either be individual words or they can be word pairs. For example, if the top 3 search terms are “online,” “internet auction,” and “buying” then we would perform the following 6 searches where quotes denote word pairs that should be found together:

- online
- "internet auction"
- buying
- online "internet auction"
- "internet auction" buying
- buying online

From each of these searches we take the top 10 results for a total of 60 results from which duplicate URLs are removed.
6.5 Filtering Search Results

Once we have a set of possible related documents, we need to filter them based on similarity to the original annotations. We use the vector space model from information retrieval to compare each search result to the original annotations. We retrieve all of the text from each returned document and use the algorithm in figure 27 to calculate a score for each document.

```
foreach word W in document
    if W is in annotation
        count = count + 1
    score = count / number of words in document
```

Figure 27: Document score algorithm

This gives us a normalized similarity score for each document. With this we can then return the highest ranked documents to the user for viewing. We have found through experimental data that we will present later, that document scores of less than 0.4 are not considered very relevant by the user. Therefore we return the top 5 documents by score, excluding documents with scores less than 0.4.

6.6 Presenting Related Documents

We wanted to integrate related information into the normal flow of finding, reading, and annotating, so we decided to show the links to related documents in the same place users view notes. Since ScreenCrayons allows users to view notes in a zoomable fashion, we decided to follow a similar concept for viewing related information.

When a note is created and saved, it is put on a queue for processing. This way searching and filtering tasks can be done in the background with idle network and CPU time.
When the note reaches the head of the queue it goes through the process we have described for finding related information. Once a note has been processed, the user is able to view any available related documents that have been found. When the user views a note that has related information a Google icon below the note indicates additional documents have been found (see figure 28).

If the user wants to view the related information they can progressively zoom in by clicking on the plus button that appears on the icon when the mouse hovers over it. Users can zoom back out by clicking on the corresponding minus icon. This is the same mechanism used to zoom in and out of annotations in ScreenCrayons.
The first zoom level shows the keywords that the system extracted from the annotations (see figure 29). The second zoom level shows actual links to the related information where the text for the link is the title of the related document (see figure 30). The last zoom level takes advantage of Google's web page snippet feature that shows the query in context of the document to give the user a brief overview of the results (see figure 31).

By clicking on any of the hypertext links the default web browser opens the associated document. This allows the user to view the contents and easily annotate any
information that they find. The system also keeps track of which links the user has followed so that they can remember what related information they have already viewed.

6.7 Results of Document Filtering User Test

In order to verify the effectiveness of our word weighting algorithm and document filtering algorithm, we performed a user study to see how well our algorithms compare to searches performed by users. We selected twenty test subjects from varying age groups and backgrounds. All subjects had at least a minimal exposure to computers and Internet based search engines. The subjects were asked to perform the following tasks:

First, the subjects were asked to select a topic that they wanted to find information about. There were no restrictions on their choice of topic. After choosing a topic, they were asked to choose a set of search terms to be used with the Google search engine to find information related to the topic. Then they entered these search terms into our test framework which presented them with the top ten search results returned by their query. The user interface that the test subjects used can be seen in figure 32.
For each result the test subjects would perform two tasks. This first task which was required for each search result, was for the subject to rank the result:

1 - Very Irrelevant
2 - Irrelevant
3 - Somewhat Irrelevant
4 - Somewhat Relevant
5 - Relevant
6 - Very Relevant

The other task was for each subject to digitally annotate any portions of the search result that applied to the topic they had selected. This task was optional for each search result.
since not every result would have information applicable to the subject's topic. The subject’s annotations were immediately associated with regions of the results so that the user could change any incorrect region associations.

Once the subject had completed going through the initial search results the test framework would use the word weighting algorithm to create a list of ranked words and word pairs. The three highest ranked would be used together to perform a single search using Google. Once again the subject would be presented with the top ten search results, this time only selecting a ranking for each result.

The final step was for the test framework to use the word ranking algorithm in conjunction with the document filtering algorithm. Using the top 3 ranked words and word pairs, a series of 6 searches were performed as explained in section 6.4. The top 10 results from each search were returned and the similarity score for each page was computed. The top 10 ranked documents were presented to the subject from highest ranking to lowest. Once again the subject was asked to rank each of the results.

In figure 33 we can see the results of this experiment. The data points displayed are averages over the twenty tests. The “user” data points are the results from the subject’s initial search terms and the users ranking of those results. The “unfiltered” results are from the search using the word weighting algorithm and a single Google search. The “filtered” results are from using the word weighting algorithm in conjunction with the page filtering algorithm.
The “Display Order” is the order in which the results were shown to the subject. For the first two tests this is the order returned by Google, but for the last test the order is based on the similarity score for each result being presented from lowest to highest.

The data shows that the results from the combination of the word ranking and page filtering algorithms are almost always ranked higher than the users own search results. This is even more pronounced in the first few results indicating that the algorithms are able to present more relevant data to the user quicker than the users own search queries. We can also see that the word filtering algorithm alone is not much better, and sometimes worse, than the users own queries. This data indicates that our algorithms in conjunction with annotated notes, can present users with related information that is generally more relevant than the users own queries.
Chapter 7 - Contributions and Future Work

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 minimal 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 Thomas the office manager. Creating a “to do” list while reading email is as simple as creating a “to do” crayon that is placed in the crayon box and always available. Adding an item to the list consists of grabbing the “to do” 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 “to do” list. We can see an example of what this might look like in figure 34.
Handling notes from the email, memo, presentation 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 Thomas’s custom stress test application. 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 simultaneously is irrelevant to the image/note tools.

Creating notes is also spontaneous. When unrelated items like the microwave 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"
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 50¢ in 2006 it seems that space is not an issue.

ScreenCrayons provides a unified annotation and organization tool that allows users to easily bring together information from different sources into one place. The system assists users in the search for information by using text and annotations from the stored notes to find sources of information related to what the user has already found. By pulling techniques and ideas from image processing, information retrieval, user interfaces, information visualization, and annotation systems, ScreenCrayons is a powerful yet easy to use research and information organization tool.

### 7.1 Future Directions

Annotating multi-page documents can be somewhat cumbersome because they generally take up more than one screen at a time. This requires either capturing each screenful, or only capturing screens where you want to take notes. It would be interesting to experiment with ways to directly capture multi-page documents easily. One method that could still work across most applications is a printer driver. A user could select their...
ScreenCrayons printer driver and print the document which would directly import a series of notes, probably one per page, that could be easily read and annotated. The system might also be able to support the notion of multi-page notes. The ScreenCrayons note taking interface could be augmented to allow the user to “read” the document by easily switching between successive pages.

Another note organization method that could be used to replace and/or augment the current hierarchical structure would be concept tags. By allowing for “tagging” of notes ScreenCrayons could allow the user to easily place notes in multiple categories. If a user utilized two different crayons when annotating a note, the note could be tagged with both crayons. This way the user could find the note by searching for either tag and notes are not limited to a single category.

An extension that might be useful for annotating would be using pressure information from pressure sensitive input devices like those of tablet PC’s and pen tablet input devices. The pressure information might be useful to improve the annotation to region association algorithm by more accurately detecting leading and trailing edges of annotations based on the users pressure. Also the pressure could be used to influence the score that individual words receive during the searching and filtering process. Annotations that were created with more pressure could receive higher weights. The pressure information could also be used for setting the width of ink strokes.

The current implementation of ScreenCrayons uses Google which is a keyword search engine. It would be interesting to use a searchable document repository that supported weighted word queries, to determine if we could retrieve information that was even more
relevant. The weighted word queries would more closely mirror the information that we extract out of the annotations for use in searching. Also ScreenCrayons currently only supports text based document formats (HTML, TXT, etc) for filtering. This could be expanded to many other common document formats.

ScreenCrayons could also be extended to support enhanced searching of notes. Since the notes contain images as well as the extracted text information, we could allow for full text searches of all of the notes. This would make it even easier to find specific pieces of annotated information.
Chapter 8 - References


[BUD00] Budzik, J., Hammond, K., User Interactions with Everyday Applications as Context for Just-in-time Information Access Free Form Digital Ink Annotations,


Hold Your Knowledge in Place, Hypertext '91, ACM, pp 261-2751991.


[NAT02] Nation, D., Roberts, D., Card, S., Browse Hierarchical Data with the Degree of Interest Tree, Xerox PARC, Palo Alto, California, 2002.


