A Self-Organized File Cabinet

Dawn Lawrie
Department of Computer Science
University of Massachusetts
Amherst, MA 01003*

Daniela Rus
Department of Computer Science
Dartmouth College
Hanover, NH 03755

Abstract

The self-organizing file cabinet is an information retrieval system associated with a user's physical file cabinet. It enhances a physical file cabinet with electronic information about the papers in it. It can remember, organize, update, and help the user find documents contained in the physical file cabinet. The system consists of a module for extracting electronic information about the papers stored in the file cabinet, a module for representing and storing this information in multiple views, and a module that allows a user to interact with this information. The focus of this paper is on the design and evaluation of the self-organized file cabinet.

1 Introduction

Most ordinary offices contain a desk with a computer and lots of papers scattered on it. They also have file cabinets and shelves full of books. If one considers only the physical documents in this room, it is highly probable that the owner of the office could only locate a portion of what is in it. She may remember an interesting article she read a year ago but not its author, and therefore, cannot retrieve it to show it to an interested student. Or perhaps a deadline is approaching and she needs a document, but can't remember where she put it. Is it still in the file cabinet? Or did she already take it out and put it on her desk?

These problems are faced by many people today, and the only remedy they see is that they need to be better organized. There is another solution. Rather than trying to alter human nature, one can go to a computer for help. By sharing the task of organization, one could be a lot more efficient. All that is needed is one more tool in the office, a scanner. Then a link between her file cabinet and the physical world is created. Each new document that is placed in the file cabinet is first scanned. A drawer is chosen by the user or the system and then placed in the front of the drawer. Now "author" is no longer the only information she can use to search for the document she wants to give that student. She will be able to query the system through text strings or color or both to find the document

In this paper we describe a system we created for information and knowledge management in an office setting. The self-organizing filing cabinet is a system that enhances a physical filing cabinet with electronic information to support better filing and queries. The system works as follows: each document is scanned before being filed away. The scanned image can be filtered with OCR to recover the words of the document. The scanned image can also be processed by color and other layout filters such as those we developed in our previous work [RS95b, RS97]. This information is added to a database. The scanned document is then compared with the rest of the database to identify the best physical filing location. The contents of the database are kept organized as a hierarchy of clusters. When looking for information in this physical filing cabinet, a digital query can be used to compute locations of all relevant documents in the physical space. The database can also be used to browse the contents of the collection.

This paper also contains three evaluation studies: the first study measures the performance of the system; the second study measures the effectiveness of using the file cabinet to organize information for the user; and finally, a small user study explores the utility of such a system.

1.1 Related Work

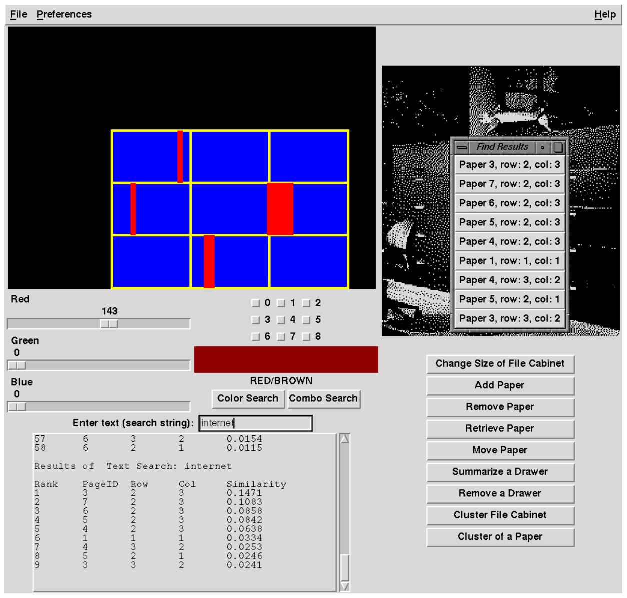
The quest for using automation to improve information access in an office setting has a long history [Mal83]. Recent efforts to enhance physical environments with electronic information include several projects. The Intelligent Room project at the AI Lab at MIT created a room surveyed by cameras that can recognize and understand physical gestures. Progress on this project has been reported in [Tor95]. Euro Xerox and Hitachi created interactive desks where the user can write with a stylus pen on the desktop. The desktop consists of a display that can capture the user's input. A camera mounted on the desktop is used to project on the desktop rather than extract information [AM+94]. Finally, [RS97] describes a self-organizing desk.

Our virtual file cabinet system draws from progress made in several areas: self-organizing systems [Koh90, Sam69, CKP93, APR98, APR99], information retrieval and organization [Sal91, SA93, Wil88, RA95], robotics and vision [MRR96, HKR93], and automated document structuring [TA92, MT*91, RS95b, JB92, NSV92].

2 Example

Figure 1 shows a typical setup for the self-organizing file cabinet. The user interacts with the file cabinet through a GUI. The right side of the GUI contains a photograph

*Most of this work was done while at the computer science department of Dartmouth College.



```
Paper 3, row: 2, col: 3
Paper 7, row: 2, col: 3
Paper 6, row: 2, col: 3
Paper 5, row: 2, col: 3
Paper 4, row: 2, col: 3
Paper 1, row: 1, col: 1
Paper 4, row: 3, col: 2
Paper 5, row: 2, col: 1
Paper 3, row: 3, col: 2
```

```
57      6       3       2       0.0154
58      6       2       1       0.0115

Results of  Text Search: internet

Rank    PageID  Row     Col     Similarity
1       3       2       3       0.1471
2       7       2       3       0.1083
3       6       2       3       0.0858
4       5       2       3       0.0842
5       4       2       3       0.0638
6       1       1       1       0.0334
7       4       3       2       0.0253
8       5       2       1       0.0246
9       3       3       2       0.0241
```

Figure 1: A snapshot of the virtual file cabinet system. This figure shows the basic GUI for the virtual file cabinet. It also shows the response of the file cabinet to a query. Note that each drawer is represented in a side view so that the approximate location of the documents found can be marked in the virtual drawers.

of the physical file cabinet setup in the office. The left-hand side gives an electronic visualization of the physical file cabinet. We call this image the virtual file cabinet. The bottom part of the file cabinet contains a series of buttons that can trigger the retrieval and organization capabilities of the system.

The first task that needs to be accomplished in setting up a file cabinet is adding papers. In order for a paper to be added, there needs to be an ASCII and TIFF version of the document. These are created by scanning the document and then running the scanned TIFF file through an OCR filter. Although currently these steps must be done separately, it is possible to imagine a system that could take the TIFF file and do the rest on its own. After the electronic information is captured from the scanned image (in the form of text, images, color, and layout information), the user has the choice of (1) selecting the drawer location for the new document in the file cabinet; or (2) asking the system for a drawer recommendation. A drawer is recommended on the basis of the greatest number of similar documents at a given threshold. Usually a range of thresholds is used so that the document is filed with the most similar documents. This range is specified by the user prior to the recommendation. Finally, the document is placed in the front of the drawer.

The user can currently query the file cabinet based on text, color, or a combination of these types. Queries based on layout information about the document also could be supported. The system can answer queries such as "where is the paper with the red table in the lower right hand corner?" Or "which papers are about intelligent agents and have red pictures on the left side?" The file cabinet system responds by highlighting the location of the relevant documents in the virtual file cabinet. These locations correspond to the physical location of the documents in the real file cabinet. A button in the "Find Results" window gives the user access to the picture of the document and finds its exact location in the drawer, i.e. the number of documents in front of it. The user can then proceed to the relevant physical drawer to retrieve the document.

The file cabinet will also provide a table of contents. The self-organizing file cabinet employs an information organization algorithm that presents the user with a visual summary of all the topics present in the file cabinet. By selecting one topic, presented as a cluster, the locations of all documents relevant to that topic are highlighted.

3 The System Description

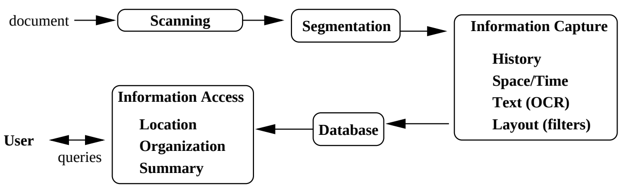


Figure 2: The system components of the self-organizing file cabinet. Documents to be filed are first passed through a scanner. The scanned images are filtered by OCR and color. Other image segmentation modules could easily be added that would recover figures, table, and other layout information. Each of the layout features is then entered into a database that contains multiple views of the documents according to these features. In addition, the database contains a field for the physical location of the document in the real file cabinet. The user has the option of specifying a physical location for the document in the file cabinet, or of asking the system for a recommendation based on the current contents of the file cabinet. The database is indexed, organized, and available for queries.

The vision of a self-organizing file cabinet is achieved by the system shown in Figure 2. Our main goal in creating this system was to develop tools to automatically create a reference database associated with the documents in an office space. We believe this to be a useful system because people in general have great difficulty maintaining organized systems of documents. This difficulty is often due to the fact that one document could belong logically in several different locations. Thus, there may be inconsistencies in placing and retrieving physical documents.

Our system addresses this problem by maintaining an electronic image of the physical cabinet. A picture of each document is computed and retained in the virtual file cabinet after it has been scanned in by the user. We believe that although the initial time spent scanning a document can be much greater than the time to place a document in a drawer, the ease with which one can find the document outweighs the initial time investment. There is also the possibility that the file cabinet could be enhanced to directly communicate with the scanner, and a more sophisticated scanner would enable a number of pages to be scanned automatically.

The user can query the virtual file cabinet by specifying a combination of word and color information. The system also offers the capability of finding other similar documents stored in the file cabinet. In addition, the system can present the user with a "table of contents" that captures the topics and subtopics of the documents contained in it. Finally, the system can help a user file documents by suggesting the best drawer for the current document based on the contents of the file cabinet.

There are four main components to the self-organized file cabinet: (1) the database that captures multiple representations for each document; (2) the search engine used to query the cabinet; (3) the organization component (used to compute a table of contents for the file cabinet and to recommend drawers for filing); (4) the GUI visualization scheme for the file cabinet. The rest of this section is devoted to describing each of these modules.

3.1 The Database

The file cabinet database is structured to mirror a file cabinet in the physical world from the design of the structure to the way in which file cabinet manipulations are performed. For example, in order to retrieve a paper, one first thinks of retrieving the paper from the cabinet. The first question that needs to be answered is which drawer the document is in. Once a drawer is known, the task can be reduced to retrieving a paper from the drawer, and the files are "thumbed through" to find the correct one. We designed classes in C++ to make normal operations on a filing cabinet as natural and intuitive as possible. The parent structure is the filing cabinet itself. A filing cabinet consists of drawers. The implementation structure of the drawers is a dynamically allocated ordered list.

To support queries that combine textual with color information, we created a database indexed by words, color, and physical information. The database comprises a collection of inverted indices, one for each attribute. An inverted index associates each attribute instance with a list of documents within which the attribute is found. The advantage of this representation scheme is a decrease in search time: given a specific attribute (a word, a color, *etc.*). The list of documents containing this attribute is available in constant time. Each index is computed incrementally by a filter that operates on the document's scanned image. Currently, our filter library consists of two filters: OCR and color. The architecture is expandable, and we will add new filters in the future.

3.1.0.1 OCR. The OCR filter we used was the software product OmniPage Lite by Caere, but any off-the-shelf OCR package could be used. From a random sampling of the 88 documents scanned for the experiments, it was determined that the OCRed text was 97.5% correct by word with a 2.7 standard deviation. This error rate is better than most published error rates [Har99] because the test pages came from a magazine where the text was black printed on glossy white paper and was of a size and boldness where the OCR did very well. In addition, problems with the layout were not counted as an error because we do not consider proximity of words important in the information retrieval part of the file cabinet.

3.1.0.2 Color. The color filter works by building a color histogram annotated with layout information for each object. The filter determines the 24 most prevalent colors occurring in the document and the location of each color. Location is a layout attribute determined by placing a 3×3 grid on the paper.

3.1.0.3 Space, time, and history. Each paper is assigned a location in the file cabinet using coordinates that can be provided by the user or computed by the system. Each paper is also given a time-stamp, which is the current time the paper is placed in the drawer. If the paper is moved, it is given a new time stamp. In addition, each paper is positioned within the document array as it is in the physical file cabinet so that the location within the drawer is readily available.

3.2 Filing and Retrieving Documents

The filters defined in Section 3.1 generate a web of representations for each document. We compile this multiplicity of representations in a database. This database supports the following file operations: adding a paper to the file cabinet, removing a paper from the file cabinet, computing a table of contents for the file cabinet, and computing the best location for filing a new document. The information in this database changes dynamically, as driven by these operations. In response to each event, the database is updated automatically.

The file cabinet can be queried with keywords, with an entire document (full text), with color and layout information, or any combination of these attributes. We use an augmented version of the *SMART System* [Sal91], which is a sophisticated text-retrieval system. We augmented SMART to also support color and layout indices. SMART copes well with text partially corrupted during the OCR process. Its basic premise is that two documents are similar if they use the same words. Documents and queries are modeled as points in a *vector space* defined by the important words occurring in the corpus. When all texts and text queries are represented as weighted vectors, a similarity measure can be computed between pairs of vectors that captures the text similarity. We use this similarity measure as the basis for computing hyper-links between documents that are similar to each other in this statistical framework.

The textual information contained in the documents in the file cabinet is organized by topic using *the star clustering algorithm* we developed [APR98, APR99] on the document space. The star algorithm is simple, easy to implement, efficient, and gives a hierarchical organization of a collection into clusters. Each level in the hierarchy is determined by a threshold for the minimum similarity between pairs of documents within a cluster at that particular level in the hierarchy. This method conveys the topic-subtopic structure of the corpus according to the similarity measure used.

We have developed an off-line version of the star algorithm to handle static collections and an on-line version to handle dynamic collections [APR98, APR99]. Both algorithms are used in the file cabinet system. These two algorithms compute clusters induced by the natural topic structure of the space. To compute accurate clusters, we formalize the clustering problem as one of covering a thresholded similarity graph by cliques. Covering by cliques is NP-complete and thus intractable for large document collections. Recent graph-theoretic results have shown that the problem cannot even be approximated in polynomial time [LY94, Zuc93]. We instead use a cover by dense subgraphs that are star-shaped[1], where the covering can be computed off-line for static data and on-line for dynamic data.

The star algorithm is based on a greedy cover of the thresholded similarity graph of a document collection by star-shaped subgraphs; the algorithm itself is summarized in Figure 3 below.

[1]In [SJJ70] stars were also identified to be potentially useful for clustering.

For any threshold σ:

1. Let $G_\sigma = (V, E_\sigma)$ where $E_\sigma = \{e : w(e) \geq \sigma\}$ be a similarity graph.
2. Let each vertex in G_σ initially be *unmarked*.
3. Calculate the degree of each vertex $v \in V$.
4. Let the highest degree unmarked vertex be a star center, and construct a cluster from the star center and its associated satellite vertices. Mark each node in the newly constructed cluster.
5. Repeat step 4 until all nodes are marked.
6. Represent each cluster by the document corresponding to its associated star center.

Figure 3: The off-line star algorithm. The on-line star algorithm is described in [APR98, APR99].

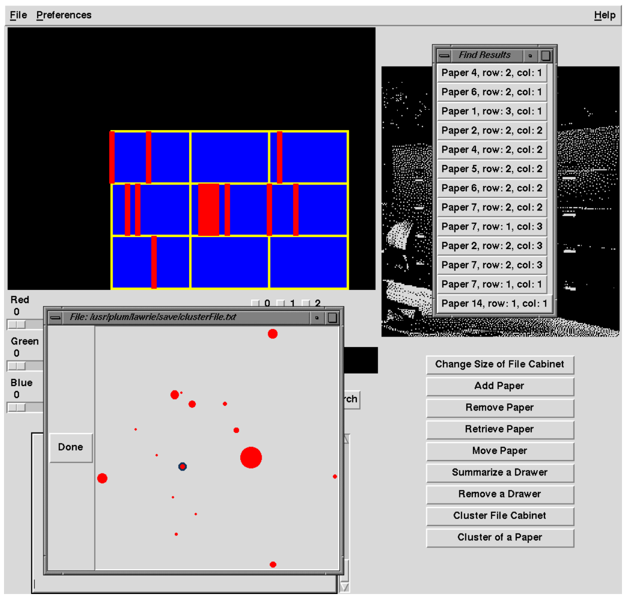
We show [APR98, APR99] that the off-line and on-line algorithms produce high quality clusters very efficiently. Asymptotically, the running time of both algorithms is roughly linear in the size of the similarity graph that defines the information space. We have derived lower bounds on the topic similarity within clusters guaranteed by a star covering, thus providing theoretical evidence that the clusters produced by a star cover are of high quality (in other words, have a high degree of precision). The file cabinet uses the off-line version of the star clustering algorithm to organize an existing file cabinet and extract a table of contents. The file cabinet uses the on-line version of the star algorithm to recommend a drawer and to insert a new document in an existing file cabinet organization.

3.3 The Graphical User Interface

The GUI was spatially designed with four quadrants as shown in Figures 1 and 4. In the upper left-hand corner, a graphical representation of the file cabinet appears. In the upper right hand corner, a picture of a file cabinet is shown. The lower left-hand corner displays the search operations. The output from a search is printed in the scrolling text area. In the lower right hand corner there are a series of buttons that allow for the manipulations of documents and the file cabinet as well as buttons pertaining to the clustering by topic features.

The graphical representation of the virtual file cabinet is displayed in blue with drawers outlined in yellow on a black background. Each drawer is actually a side view of the open drawer rather than a front view. This enables the interface to display the approximate position of a paper within the drawer by drawing a red line to highlight the document. When a drawer becomes full, many papers could be represented by one line. The results of most searches yield more than one paper. In order to figure out exactly which paper is the one the user is interested in, the user can highlight a specific paper in green.

The lower left part of the screen is devoted to the searching mechanisms. The top part of this section is used to select search colors by mixing red, green, and blue. This allows selection of twenty-four different colors. The constructed color is displayed to the right of the sliders. Above this color display are nine radio buttons used to select portions of the page to search. The search



```
Paper 4, row: 2, col: 1
Paper 6, row: 2, col: 1
Paper 1, row: 3, col: 1
Paper 2, row: 2, col: 2
Paper 4, row: 2, col: 2
Paper 5, row: 2, col: 2
Paper 6, row: 2, col: 2
Paper 7, row: 2, col: 2
Paper 7, row: 1, col: 3
Paper 2, row: 2, col: 3
Paper 7, row: 2, col: 3
Paper 7, row: 1, col: 1
Paper 14, row: 1, col: 1
```

Figure 4: The Graphical User Interface to the virtual file cabinet shows the output of the system in response to a request for the table of contents. Each blob represents a topic contained within the file cabinet that was computed by the star algorithm. The distance between the blobs is proportionate to the distance between their corresponding topics computed by the star algorithm in the vector space model. The user can select a topic by clicking on a blob. The location of all the documents contained in that cluster are then highlighted.

is initiated by clicking the "Color Search" button. Below the color search part of the display is a line to enter text queries. The text query commences by pressing enter in that box. Next to the "Color Search" button is the "Combo Search" button. A combination search takes the results of a text query and color search and displays only those papers that are found by both searches. The results of queries are displayed in the scrollable text area at the bottom of the screen. The search identifies pages by page identification number, drawer row, and drawer column. The text query also includes information about the similarity of the document to the query. Documents are listed in rank-order. After any search, a "Find Results" window will appear that provides access to information about the found pages including a way to view a particular scanned page.

The lower right part of the screen consists of all buttons used to manipulate the file cabinet, as well as those related to clustering documents by topic. These buttons include changing the size of the file cabinet; adding, removing, showing, and moving a paper; summarizing and removing a drawer; and clustering the whole file cabinet or finding the clusters that include one particular paper to discover similar papers.

In the Menu Bar there are two pull down menus. One is "File" which allows one to load or save a file cabinet, or to quit the application. The user is prompted to specify a path when saving or loading a file cabinet. There are two menu items in the second pull-down menu, labeled "Preferences". The first, "Set Image Directory", asks the user to specify the directory where all scanned documents in the file cabinet are located. The second, "Recommendation Threshold", allows the user to set up the range of the thresholds used by the file recommendation algorithm. This algorithm also requires a frequency to be used with the range, which is also specified in this window.

4 Experiments and Discussion

The file cabinet was implemented on an SGI Indy, and its database was written in C++. The file cabinet database was integrated with existing pieces of software that had been used in the self-organizing desk [RS97].

We have identified three different ways of evaluating the virtual file cabinet system. First, we measured the performance of each module in the system. Second, we devised an experiment for evaluating the most complex function of the system, which is the recommendation of the filing location. Finally, we did a small user study to evaluate peoples' reaction to such a system.

4.1 Performance

The system was tested on an SGI Indy with an R5000 processor to make sure that all operations occurred at a reasonable speed so that users would not become impatient with the system. The results of this test are shown below. The time is recorded in *clockticks*. The times shown are an average of at least ten runs.

Function	*Avg*	*Min*	*Max*
Delete Drawer	0.09	0	1
Find - Single	0.54	0	1
Move Page	0.72	0	3
Print Drawer	5.16	0	7
Search by Color	8.18	1	31
Change File Cabinet Size	8.27	0	22
Find - Group	16.80	2	32
Combo Search	54.80	33	179
SMART Search	61.80	35	161
Initialize	85.59	20	207
Save	103.77	71	176
Load	241.13	151	945
Add Page	334.71	274	674
Remove Page	572.55	512	763
Cluster Cabinet	710.78	498	1004
Recommend Drawer	814.90	330	1556
Cluster on a paper	871.44	782	1389

These timings show that most operations like search and move that would be utilized a number of times over the lifetime of a document are very fast – less than one hundred clockticks – which seems instantaneous to a user. The setup operations take longer. We believe that people do not mind waiting a bit when an application is starting up or loading, so those timings do not present a problem. The slowest functions (Cluster Cabinet, Recommend Drawers, and Cluster on a paper), however, could cause a problem. We reason that since a paper will only be added and removed once, the extra time will not cause too much impatience. The other functions all involve looking at all the papers in the cabinet and as a result are slower. Note, also, that these measurements were done on a very slow machine (an SGI Indy R5000). All of these functions take either constant or linear time, so they will scale.

4.2 Evaluating the Filing Location Recommendation

The most computationally complex part of the system is the drawer recommendation operation. This feature of our system enhances the virtual file cabinet so that it is more than just a mirror image of a real file cabinet. The ability to identify the most logical filing location for a new document leads to a system with limited self-organizing capabilities. Our goal is to have a file cabinet that is organized by topic rather than more traditional filing systems such as by author or journal.

There are two competing attributes that we would like our file cabinet to have. First we would like documents to be evenly distributed in the file cabinet, and second we would like papers on the same topic to be grouped together. This will create a cabinet where documents are evenly distributed and grouped by similarity. To this end we chose two measures for evaluating the filing location recommendation features. The first measure captures how well the cabinet was able to evenly distribute the papers throughout the drawers. The second measure captures the quality of the topic and subtopic clusters computed by the system.

Our experimental document collection consists of the pages from Communications of the ACM July 1994– Volume 37, Number 7, a special issue on Intelligent Agents [CAC93]. Some 88 of the total pages of this issue contain articles about different aspect of intelligent agents. We regarded each page in the magazine as a separate document. We hypothesized that all pages within an article would be more closely related to each other than pages of other articles. For the most part this proved to be correct, except for a few notable exceptions where pages were not related to any other page in the magazine. One was in the article "A Conversation with Marvin Minsky about Agents" where American society and education are discussed. Occasionally, a large portion of the final page of an article consists of references, which greatly affects its similarity to previous pages.

We set up three different types of experiments to test the ability of the file cabinet to recommend a drawer. Every file cabinet used had four drawers. Since the recommendation will vary depending on the order that papers are inserted, we created eleven different file cabinets for each experiment. In one of these runs the documents were inserted in the order they appeared in the magazine. In the ten other runs, the order of insertion was determined by a random order generator. Since page recommendation is a deterministic algorithm given a state of the file cabinet, multiple runs of the in-order insertion of pages were unnecessary.

In the first experiment we entered the entire collection at a range of thresholds 0.45 to 0.5 (in increments of 0.05) for the threshold parameter used by the star clustering algorithm. First similar documents were sought at a threshold of 0.5. If none were located, similar documents were sought at the threshold of 0.45.

In the second experiment we entered all pages but the first page of each article using the same threshold range as in the first experiment. We chose this variation because we observed that a very large number of documents were clumping together and hypothesized that the introduction pages of the articles tended to be similar, since all articles were on related subjects.

In the third experiment we used a higher range of thresholds, 0.55 to 0.75. This meant that if a document was not similar to any others at the 0.75 threshold, then a 0.70 threshold was tried. The threshold was decremented to 0.55, at which point the document was considered to have no similar matches and was therefore filed in the drawer with the fewest number of documents so far.

As shown in figures 5 and 6, the high threshold vari-

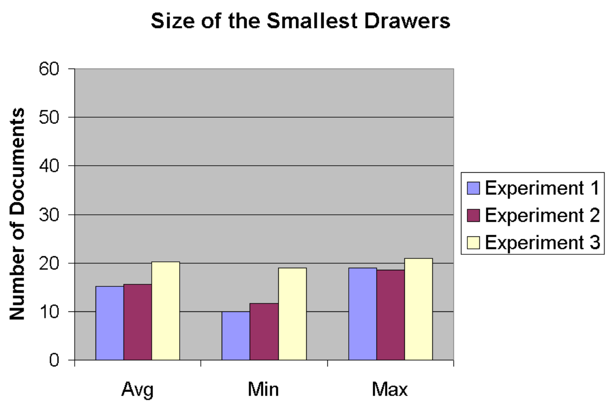


Figure 5: Compares the size of the smallest drawers from the three experiments. The average is the average size of the smallest drawer. The minimum is the smallest of the small drawers for that experiment. The maximum is the largest of the small drawers, so the mean and range of the smallest drawers are presented over the eleven runs performed for each experiment.

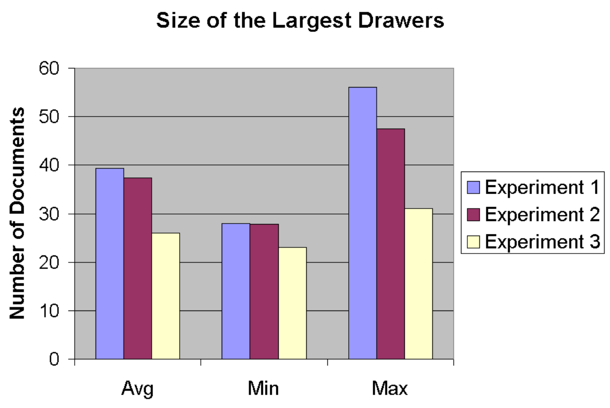


Figure 6: The mean and range of the largest drawers are presented over the eleven runs performed for each experiment.

ation gave the best results as far as distribution of papers among the file cabinet drawers. The largest drawer had an average size of 26.0 papers. The smallest drawer had an average size of 20.2. The other two variations performed almost equally, but the distribution of papers was more uneven.

In contrast the variations that used a lower threshold performed significantly better at grouping pages from a particular article. When all pages were filed at thresholds of 0.5 and 0.45, 82% of pages of an article grouped together when filed in random order. This value increased to 91% when the documents were filed in order. Without the first page of each article, no preference was given towards the pages being filed in order versus random order. Eighty-two percent of the pages of an article were found grouped in the same drawer when the pages were filed in order, and 83% when the pages were filed in random order. The high threshold variation only had 70% of the pages of an article group in the same drawer when pages were filed in random order. This number decreased to 62% when the pages were filed in order.

Characteristic	*People*
Dislike the Color Set-up	5
More Mouse Use	5
List Actions	4
Logical Search Operators	3
Titles Papers	3
Add Keywords to Papers	2

Figure 7: This table lists some of the reactions by the users and how many of those 6 users had the same reaction.

From these results it is evident that the most important choice the user will make is defining a range of thresholds to find related documents. This requires the user to have an idea about what the purpose of the file cabinet is before she begins adding documents to it. Of course, the user can change the range of thresholds at any time, but documents already filed will stay in current places. The importance of drawers containing similar topics is to facilitate the gathering of documents on a given topic. If all documents on a similar topic are found in only one or two different drawers it will be simpler to retrieve them.

4.3 User studies

User studies were performed to evaluate how well the interface was able to facilitate use of the file cabinet. In this sense we were more interested in constructive criticism than an evaluation of the statistical significance of effectiveness. The user studies revealed attributes of the file cabinet that were difficult or uncomfortable to use. To instruct people in using the file cabinet, we gave a brief demonstration that pointed out buttons and showed how to cluster the file cabinet. Finally, we demonstrated a color, text, and combination search before quitting the application and allowing them to use it. They were given some sixteen tasks which are listed in Appendix A.1.

There were six participants in the user study; all were members of the Department of Computer Science at Dartmouth College either as faculty or students. They were asked to evaluate this from the perspective of a software user and as a user of file cabinets.

The main complaint of the users as shown in Figure 7 was that they did not like trying to construct a color, and they wanted to do more things with the mouse and less with the keyboard. In general they wanted the capability to review actions performed by the system. Some users also expressed a desire to assign titles to pages rather than be presented with the identification number assigned by the file cabinet. Finally, many expressed an interest in adding information to what SMART was already indexing to emphasize their interest.

The user studies showed that constructing a color was very problematic. One issue was that people did not know which mix of colors composes the one for which they are looking. Another problem was the difficulty in defining the names for colors. For example, a person might call a particular color "green" that the computer identifies as a "yellow-green." A selection method that does not include names would alleviate this ambiguity. Another option would be to use a method of naming colors through a heirarchy of colors that uses full names, partial names, and base hues like the one described in [DMR98].

Users found the "Find Results" window intimidating. Each document found is presented as a button so that actions such as highlighting and displaying the page are easily accessible. The problem is that since each page is labeled with its coordinates (drawer by row and column, and page number), the user does not remember exactly which page she has selected. By changing the color of the last button pressed, this problem would be resolved.

The users wanted the scrolled text on the lower left side to be used as a record for all actions instead of just those involved with searches. This would enable one to keep track of all the file manipulations during a session.

Adding keywords to the document was a feature that many desired. If only the first page of a paper is scanned and added to the file cabinet, it is possible that the reason the person is filing the paper is not discussed on the first page. It would be very easy to modify a temporary version of the OCRed file with the added keywords and have SMART index the new file. The keywords could be kept with the filter data, and if in the future the user changed the keywords, the paper could be re-indexed by SMART. One user also desired that his keywords be given more weight than what appeared on the page. This could also be implemented using SMART.

5 Conclusion

We believe the experiments show that the virtual file cabinet is a usable and useful system for computing and maintaining references to a collection of documents contained in a physical file cabinet. Most of the file operations are very fast, seeming nearly instantaneous to the user. The experiments performed on the ability of the file cabinet to recommend drawers shows that it is a functionality that can be used. Although the users suggested many improvements that we could make to the file cabinet, they thought that such a tool would be beneficial.

6 Acknowledgements

We would like to thank Katya Pelekhov, Hans Kieserman, and Peter deSantis for their contribution of software to the project. We would like to thank Javad Aslam, Thomas Corman, John Danskin, Ahsan Kabir, Katya Pelekhov, and Cliff Stein for their participation in the user study. In addition, we would like to thank James Allan and Craig Allen for their insightful comments on the paper.

Support for this work was provided in part by ONR contract N00014-95-1-1204, DARPA contract F30602-98-2-0107, and Air Force MURI contract F49620-97-1-0382 at Dartmouth. Support for this work was also provided in part by the National Science Foundation, Library of Congress, and Department of Commerce under cooperative agreement number EEC-9209623 and the United States Patent and Trademark Office and Defense Advanced Research Projects Agency/ITO under ARPA order number D468, issued by ESC/AXS contract number F19628-95-C-0235 at the CIIR, University of Massachusetts. Any opinions, findings and conclusions or recommendations expressed in this material are the authors and do not necessarily reflect those of the sponsors.

References

[APR98] J. Aslam, K. Pelekhov, and D. Rus, Static and Dynamic Information Organization with Star Clusters in *Proceedings of the 1998 Conference on Intelligent Knowledge Management*, Washington, DC (November 1998).

[APR99] J. Aslam, K. Pelekhov, and D. Rus, A practical clustering algorithm for static and dynamic information organization, in *Proceedings of the 1999 Symposium on Discrete Algorithms* (SODA99), Baltimore, MD (January 1999).

[AM+94] T. Arai, K. Machii, S. Kuzunuki, and H. Shojima, InteractiveDESK: A computer augmented desk which responds to operations on real objects, in *Proceedings of Computer Human Interactions*, 1994.

[CKP93] D. Cutting, D. Karger, and J. Pedersen. Constant interaction-time scatter/gather browsing of very large document collections. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1993.

[DMR98] M. Das, R. Manmatha, and E.M. Riseman. Indexing Flowers by Color Names using Domain Knowledge-driven Segmentation, in the *Proceedings of IEEE Workshop on Applications of Computer Vision* (WACV98), pages 94-99, 1998.

[Har99] S. M. Harding, *Personal Communication*, 1999.

[HKR93] D. Huttenlocher, G. Klanderman, and W. Rucklidge, Comparing images using the Hausdorff distance, in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, 1993.

[JB92] A. Jain and S. Bhattacharjee. Address block location on envelopes using Gabor filters. *Pattern Recognition*, vol. 25, no. 12, 1992.

[Koh90] Teuvo Kohonen, The self organizing map, in *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[LY94] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM* 41, 960–981, 1994.

[Mae95] P. Maes, Artificial Life meets Entertainment: Interacting with Lifelike Autonomous Agents, Special Issue on New Horizons of Commercial and Industrial AI, Vol. 38, No. 11, pp. 108-114, *Communications of the ACM*, ACM Press, November 1995.

[Mal83] T. Malone. How do people organize their desks? Implications for the design of office information systems. *ACM Transactions on Office Information Systems*, vol. 1 no. 1, pp 99-112, 1983.

[MRR96] R. Manmatha, S. Ravela, and E. M. Riseman, Retrieval from Image Databases Using Scale Space Matching, in *Proceedings of the ECCV 1996.*

[MT*91] M. Mizuno, Y. Tsuji, T. Tanaka, H. Tanaka, M. Iwashita, and T. Temma. Document recognition system with layout structure generator. *NEC Research and Development*, vol. 32, no. 3, 1991.

[NSV92] G. Nagy, S. Seth, and M. Vishwanathan. A prototype document image analysis system for technical journals. *Computer*, vol. 25, no. 7, 1992.

[RA95] D. Rus and J. Allan. Structural queries in electronic corpora. In *Proceedings of DAGS95: Electronic Publishing and the Information Superhighway*, May 1995.

[RS97] D. Rus and P. deSantis. The self-organizing desk. In *Proceedings of the 1997 International Joint Conference on Artificial Intelligence*, August 1997.

[RS95b] D. Rus and K. Summers. Using whitespace for automated document structuring. To appear in *Advances in digital libraries*, N. Adam, B. Bhargava, and Y. Yesha, editors. Springer-Verlag, LNCS 916, 1995.

[Sal91] G. Salton. The SMART document retrieval project. In *Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 356-358.

[SA93] G. Salton and J. Allan. Selective text utilization and text traversal. In *Hypertext 1993 Proceedings*, pages 131-144, Seattle, Washington, 1993.

[Sam69] John W. Sammon Jr., A nonlinear mapping for data structure analysis, in *IEEE Transactions on Computers*, 18(5):401–409, May 1969.

[SJJ70] K. Sparck Jones and D. Jackson. The use of automatically-obtained keyword classifications for information retrieval. *Information Storage and Retrieval*, 5:174-201, 1970.

[Tor95] Mark C. Torrance, Advances in Human-Computer Interaction: The Intelligent Room, in *Working Notes of the CHI 95 Research Symposium*, 1995.

[TA92] S. Tsujimoto and H. Asada. Major components of a complete text reading system. In *Proceedings of the IEEE*, vol. 80, no. 7, 1992.

[Tur90] H. Turtle. Inference networks for document retrieval. PhD thesis. University of Massachusetts, Amherst, 1990.

[Wil88] P. Willett. Recent trends in hierarchical document clustering: A critical review. *Information Processing and Management*, 24:(5):577-597, 1988.

[Zuc93] D. Zuckerman. NP-complete problems have a version that's hard to approximate. In *Proceedings of the Eight Annual Structure in Complexity Theory Conference*, IEEE Computer Society, 305–312, 1993.

[CAC93] Special issue on visualization. *Communications of the ACM*, 36(4), 1993.

A.1 User Tasks

The following is a list of tasks the users were asked to perform during the user study.

- load: /usr/plum/lawrie/save
- set pref: /usr/plum/lawrie/images
- Find out how the documents in the file are related.
- Examine a paper in the file cabinet in more depth.
- Make the file cabinet bigger.
- Add page 102 to a new drawer
- Print the drawer.
- Move the paper to another drawer.
- Print that drawer.
- Find the paper in its new location.
- Remove a drawer.
- Remove the paper that you added.
- Make the file cabinet smaller.
- Find the articles written by Doug Riecken
- Look at page 63 in the magazine: what might you remember about this page a month later and then try to find it based on those details.
- Search for the article on pages 72-76.