

# A Dynamic Grouping Technique for Ink and Audio Notes

*Patrick Chiu*

FX Palo Alto Laboratory  
3400 Hillview Avenue, Bldg. 4  
Palo Alto, CA 94304, USA  
Tel: 1-650-813-7498  
E-mail: [chiu@pal.xerox.com](mailto:chiu@pal.xerox.com)

*Lynn Wilcox*

FX Palo Alto Laboratory  
3400 Hillview Avenue, Bldg. 4  
Palo Alto, CA 94304, USA  
Tel: 1-650-813-7574  
E-mail: [wilcox@pal.xerox.com](mailto:wilcox@pal.xerox.com)

## ABSTRACT

In this paper, we describe a technique for dynamically grouping digital ink and audio to support user interaction in freeform note-taking systems. For ink, groups of strokes might correspond to words, lines, or paragraphs of handwritten text. For audio, groups might be a complete spoken phrase or a speaker turn in a conversation. Ink and audio grouping is important for editing operations such as deleting or moving chunks of ink and audio notes. The grouping technique is based on *hierarchical agglomerative clustering*. This clustering algorithm yields groups of ink or audio in a range of sizes, depending on the level in the hierarchy, and thus provides structure for simple interactive selection and rapid non-linear expansion of a selection. Ink and audio grouping is also important for marking portions of notes for subsequent browsing and retrieval. Integration of the ink and audio clusters provides a flexible way to browse the notes by selecting the ink cluster and playing the corresponding audio cluster.

**KEYWORDS:** Ink, audio, note-taking, grouping, clustering multimedia, marking, informal systems, freeform interaction, implicit structure, emergent structure.

## INTRODUCTION

In multimedia note-taking systems, interacting with data such as digital ink and audio is challenging. Due to the nature of note-taking activity, which requires informal and rapid interaction, users must be allowed to write and to record audio with minimal attention devoted to structuring data. Freeform systems have been designed to support this style of interaction for note-taking, in which the user may write ink strokes of any shape anywhere on the page, and record audio as unstructured streams. Typically, the ink and audio are correlated by time so that “playing” a selected ink stroke will play from the corresponding point in audio. Examples of such ink and audio note-taking systems are NoTime [7], Filochat [19], Dynamite [20] and

Audio Notebook [17]. When the user wishes to edit or browse the ink or audio data, the lack of structure becomes limiting. The challenge is to provide computational support to dynamically group the ink and audio data into units that are useful for user interaction.

For digital ink and audio, these computations are far more complicated than the ones for online text, where the underlying string data structure readily parses into meaningful groups of words, lines and paragraphs. In text, selection of a particular word or line is easy. For example, in many systems the user mouse clicks once to select a word, twice to select a line, and three times to select a paragraph. On the other hand, selection of handwritten ink corresponding to a word or line is difficult. This is because digital ink is composed of strokes which are graphical objects in a freeform space and not automatically mapped into meaningful units. Audio poses analogous problems. Even when it is mapped onto a timeline, it is difficult to select a segment of audio corresponding to a phrase or a turn in a conversation. These characteristics of ink and audio make it tedious for the user to manipulate multimedia data; for example, to delete a word in handwritten text or to mark a spoken sentence in audio.

In this paper, we describe a novel technique for dynamic grouping of ink and audio data for user interaction. It has been implemented in the current generation of the Dynamite [20] ink and audio note-taking system. Dynamite runs on a pen-based notebook computer with Windows 95. In addition to providing time synchronized ink and audio notes, the Dynamite system allows properties, or data types, to be added to specific ink for the purpose of indexing. Dynamite also allows segments of audio to be marked, or “highlighted”, for subsequent retrieval.

The grouping technique described here provides structure that facilitates selection of specific portions ink and audio. It is non-linear and gives rapid expansion of selections, thus providing users with control over the amount of grouping of the media. For ink, grouping levels correspond roughly to words, lines, or paragraphs of text. For audio, the levels correspond to phrases, turns in the conversations, or topic changes. The technique is based on *hierarchical*

*agglomerative clustering* of ink strokes and audio segments. Grouping is performed on ink and audio separately, and integration of these media can be accomplished by matching the resulting groups from each medium. Leveraging this tight integration of ink and audio data gives a new way to browse through multimedia notes by playing the audio groups corresponding to ink groups selected by the user.

In the following, we first describe related work, the clustering algorithm, and provide examples to demonstrate its operation. We discuss applications of this technique to selection and marking. We then show how integrating the ink and audio groups can be applied to browsing notes. We illustrate how these techniques work in practice with a real example taken from a Japanese language class. Finally, we discuss further directions for research. In the appendix, we describe an object-oriented implementation of the algorithm.

## RELATED WORK

Structuring and interacting with data in freeform systems is an active area of research (*e.g.* see [4], [16]), and understanding some of the general principles is helpful in designing a system for ink and audio note-taking. In freeform systems, flexibility in the organization of data is important. The data should not be structured prematurely; perceived structures as seen or heard by each individual user may emerge in interesting and unexpected ways, and may be subjected to multiple interpretations. When the domain of application is known, a more robust system can be achieved by designing it to dynamically interpret an anticipated set of frequently used structures appropriate for that domain. Some examples of such digital ink systems for writing, drawing and sketching are Electronic Cocktail Napkin [5], SILK [8], Tivoli [9], [10], and PerSketch [15].

For manipulating digital ink in a note-taking system, the style of interaction determines which grouping technique can be used. A system may support several interaction styles. A key characteristic is how many modes the pen interaction employs. When one mode is used for both inking and gesturing, also known as “modeless,” a pen stroke that has a certain shape (*e.g.* scratch out) or feature (*e.g.* tap or hold) is interpreted as a gesture command. All other pen strokes are rendered as ink strokes. Examples of systems with modeless interaction are Sharp Zaurus [21], Apple Newton [13], and Aha! InkWriter [1]. While being the simplest approach, one drawback is that sometimes an ink stroke is wrongly recognized as a gesture. A typical grouping technique for systems like the Zaurus with single mode interaction is based on a time grouping algorithm. Here, a set of strokes that are created within a certain threshold (say 1 second) are grouped together as a unit. A troublesome occurrence with this technique is that several lines of writing created close in time may be grouped together. One way to deal with this is to compromise the freeform nature and use background lines of the note page to aid grouping, as in InkWriter.

Other systems provide one mode plus the ability to tap, hold, or drag a selection rectangle while pressing a button on the barrel of the pen. We call these “one-and-a-half mode” systems. In this case, the problem of confusing ink with gesture is avoided at the cost of a slightly more complicated interface. An example is Dynamite [20]. Grouping by time or by selection bounding box inclusion is employed in one-and-a-half mode systems.

With multiple modes, a gesture mode or wipe mode may be provided in addition to the basic draw mode. While these systems afford a rich set of interactions, using the system is not easy for the novice or casual user. Examples are Tivoli [14], [9], [10] and PerSketch [15]. In systems with a gesture mode, allowing the selection gesture to be a closed curve of any shape provides a more flexible inclusion grouping than the standard inclusion rectangle provided in the one-and-a-half-mode systems. Tivoli has a wipe mode, in which a user can choose ink strokes and apply properties to them by touching the strokes; moreover, strokes that are not touched but are near in time or distance are also chosen [11], [12]. More sophisticated grouping techniques such as Projected-Ink Profile in Tivoli or Object Lattice in PerSketch allow multiple interpretations of the ink data on the page by computing a variety of perceptually meaningful groups and choosing one based on gestures made by the user. Because a gestural cue is needed to disambiguate the multiple interpretations, multiple modes are required.

Employing the hierarchical agglomerative clustering technique, simpler systems without multiple modes can support interaction with a range of meaningful groups. For example, the user can select the word, line, or paragraph containing a stroke by moving up to the corresponding level of the hierarchy via the interaction of repeated taps or holding down the pen on the stroke. In comparison, time-based grouping has the limitation that each stroke belongs to exactly one group. Moreover, our clustering method can parse skewed handwriting, which the Projected-Ink Profile cannot.

Interacting with audio data is even more difficult than interacting with ink data. This can be partially explained by the fact that audio does not have a natural visual representation. Some systems map audio to a timeline and display audio as bands on the timeline separated by white space corresponding to silence [22]. Other systems display the amplitude waveform of the audio signal [3]. If speaker identification is performed, more information about the audio can be displayed by using different colored bands on the timeline to represent the different speakers in the audio, as in [6]. However, even with these visual displays, selecting the desired portion of audio is difficult, since it involves choosing an interval of the timeline to play and listening to determine the actual content.

Some systems provide methods to mark audio for later access. In the system described by Degen *et al.* [3], a mark can be placed at a single point in time. The Dynamite system [20] marks segments of audio by providing a special

button which begins the mark a few seconds before the button is pressed, and which either ends after 10 seconds, or which can be ended by pressing a button. Neither of these techniques provides meaningful groups of audio, such as a spoken phrase or a speaker turn in a conversation.

### DYNAMIC GROUPING

Dynamic grouping of ink and audio is performed on primitive elements of the media. For ink, these elements are the strokes defined between pen down and pen up events. For audio, the primitive elements are segments of audio delineated by silences with duration of more than 200 ms. The choice of 200ms is based on data collected by Stifelman [17], who found that phrases in speech could be identified by the duration of the preceding pause. Although the optimal threshold for her data was 155ms, we chose 200ms to assure that phrases would not be broken. Dynamic grouping for ink and audio is performed separately. The resulting groups from each medium can be matched to provide tighter integration in a multimedia system.

The dynamic grouping technique for either ink or audio is based on hierarchical agglomerative clustering. A tree structure describing possible groupings is created (see Figures 3 and 5). Initially, each primitive element of the media defines a group. The tree structure is created by iteratively merging the two groups which are closest, where closest is defined by a distance function appropriate to the medium. For ink, the distance depends on both the spatial and temporal distance between strokes. For audio, the distance depends on the length of the silence interval between the groups. Each merge defines a level in the cluster tree. At a given level of the tree, the groups defined may correspond to words, lines, or paragraphs for ink, and spoken words, phrases or speaker turns for audio. For example, in Figure 3, level 8 of the cluster tree defines a group containing strokes forming the second line of the handwritten text in Figure 2.

The user interacts with the system to dynamically select the grouping level by tapping on an ink stroke or a point on the audio timeline while pressing the selection button on the pen barrel. The system then highlights the ink or audio group containing the selection, where the grouping is defined by the current level in the cluster tree. The user can change the cluster level to create a larger group by repeated tapping or by holding down the pen.

### Agglomerative Clustering

Clustering is performed using a hierarchical agglomerative clustering algorithm. Clusters are initialized as the primitive elements of the medium. The pairwise distance between all clusters is computed, and the two closest clusters are merged to form a single cluster. This creates level 1 of the cluster tree. The process is then repeated, so that all pairwise distances between clusters are examined and the two closest clusters are merged to form level 2. This continues until only a single cluster remains at the final level. The distance between clusters is defined as the

minimum distance between primitive elements in each of the clusters. The distance between primitive elements depends on the medium, and will be discussed in detail below.

### Ink Distance

The pairwise distance between clusters is defined as the minimum of the pairwise distances between the primitive elements, namely the pen strokes, contained in each group. The distance between two pen strokes is a joint function of the distance between the times they were created and the spatial distance between them. More precisely, the distance  $D$  between strokes  $X$  and  $Y$  is

$$D = D^T(X,Y) + \alpha D^S(X,Y)$$

where  $\alpha$  is a constant weighting of the spatial distance  $D^S$  relative to the time distance  $D^T$ . The time distance between  $X$  and  $Y$ , assuming that  $X$  was drawn earlier in time than  $Y$ , is

$$D^T(X,Y) = M, \quad \text{if } Y_s - X_e > M$$

$$D^T(X,Y) = Y_s - X_e \quad \text{otherwise,}$$

where  $Y_s$  is the start time for stroke  $Y$  (pen down) and  $X_e$  is the end time for stroke  $X$  (pen up). The time distance  $D^T$  is limited to  $M$  so that it does not become unbounded.

The spatial distance between  $X$  and  $Y$  is the weighted sum of the distance  $d_x$  in the  $X$  direction and the distance  $d_y$  in the  $Y$  direction, or

$$D^S(X,Y) = d_x + \beta d_y,$$

where  $\beta$  is a constant specifying the relative importance of distances in the  $X$  and  $Y$  directions. Since most users write along horizontal lines, we set  $\beta > 1$ , which makes horizontal grouping more likely than vertical grouping. The distance  $d_x$  is the distance from the end of one stroke to the beginning of the next stroke along the  $X$  axis. If strokes overlap, the distance is zero:

$$d_x = Y_{ulx} - (X_{ulx} + X_L) \quad \text{if } Y_{ulx} - (X_{ulx} + X_L) > 0$$

$$d_x = 0 \quad \text{otherwise}$$

where  $X_{ulx}$  is the x-coordinate of the upper left corner of the bounding box for  $X$ , and  $X_L$  is the length of the bounding box. The above formula assumes that  $Y_{ulx} > X_{ulx}$ . The distance  $d_y$  is computed similarly. Depending on variations in writing size, it may be desirable to scale the spatial distance based on stroke size.

### Audio Distance

The primitive elements for audio are segments of recorded audio separated by silences of duration greater than 200 ms. The distance between adjacent (in time) primitive elements of audio is simply the duration of the silence that separates them. In effect, only adjacent clusters are grouped.

### Examples of grouping ink and audio

We illustrate real working examples of hierarchical agglomerative clustering of ink and audio. These are created by the current generation of Dynamite.

#### *Ink Grouping Example*

Figure 3 shows an agglomerative cluster tree for the handwritten ink strokes in Figure 1. On the note page, the “4” is composed of two primitive elements 4a and 4b since a pen up event occurred in creating the vertical stroke. The check mark was written after all the digits. Since the distance between the two strokes of the “4” is smallest, this is the first level of the cluster tree. At level 2, the “8” and “9” are merged. At level 6, the bottom line of digits is grouped. At level 9, the bottom two lines are grouped and at level 10, all the strokes are merged. Note that the check mark was written closer in time to the bottom line than the top line, but the spatial component of the distance measure causes it to merge with the top line rather than with the bottom line, which would be the case for a purely time-based distance.

#### *Audio Grouping Example*

The timeline in Figures 1 and 2 displays the audio activity. During recording, segments of audio activity are detected and these show up as gray bands. The white space between bands indicates silence. By examining the lengths of the silences on the timeline, one can see the distance between audio segments. There are 6 primitive elements of audio, which are ordered by time and labeled as {A1, A2, A3, A4, A5, A6}, Their transcriptions are:

- A1 “In the second line”
- A2 “we write a 4”
- A3 “which consists of two strokes.”
- A4 “We also write a 5.”
- A5 “Let’s put in a third line”
- A6 “and write in some numbers.”

Figure 5 shows an agglomerative cluster tree for these audio segments. Since segments A1 and A2 are separated by the least amount of silence, they are grouped first at level 1, followed by segments A5 and A6 at level 2. At level 3, segment A3 is grouped with segments A1 and A2, and at the final level 5, all segments are grouped together.

### **A Naive Grouping Algorithm**

The fact that grouping for a selected stroke changes only at a small number of discrete levels above that stroke is a critical part of the agglomerative clustering algorithm. When the user wants to increase the size of the cluster, the fact that there are only a few levels makes it easy for the user to select. In order to illustrate, we consider a naive grouping algorithm. This algorithm increases the size of the group linearly by consecutively adding the next closest primitive element. The user selects an initial stroke, and at each level of the tree the next nearest stroke is merged into the cluster. Figure 4 shows the cluster tree for the stroke 4a. At level 1, 4a is merged with the next closest stroke, namely 4b. At level 2, the group consists of the digits “4”

and “5”. Subsequent digits are added to the group linearly, one at a time.

This algorithm provides a more gradual increase in the cluster size, so the user has to move through many more levels to find the proper cluster. It also requires a separate cluster tree to be built for each stroke. Using the agglomerative clustering algorithm, Figure 3 shows that only 4 levels are needed to achieve the range of clustering for stroke 4a, namely level 1, 8, 9, and 10. For the naive algorithm, Figure 4 shows that all 10 levels are required. Another problem here with the naive algorithm is the unnatural twisting of the “2” and “3” at levels 7 and 8. This occurs because the “2” is closer than the “3” in distance to the digits in the middle line, overpowering the relatively small difference in time when the “2” and “3” were written.

### **APPLICATION TO SELECTION AND MARKING**

Dynamic grouping provides an efficient method to select meaningful units of ink and audio. In pen-based systems, selection of groups of ink strokes is required for editing operations. These operations might be moving the ink to another location on the page, scaling the ink, or deleting it. Some systems, such as Dynamite, provide the capability to mark a portion of ink or audio by adding properties or data types. To perform marking, the user makes a selection, then chooses the desired property from a toolbar palette. For example, the user can select a group of ink, and apply a property such as “To Do” or “Name” to the ink. In many systems, the user can change the color of an ink group, or increase/decrease its thickness. When performing these operations, the user generally has in mind a particular group of ink, typically a word, a line, or a phrase. Since taking notes requires a high cognitive load and leaves little time for other tasks, selecting the desired group needs to be simple.

Dynamic clustering provides structure to digital ink that allows the user to interactively make a selection without too much effort. The user taps while holding down the pen button on a portion of the desired ink, and the ink stroke touched by the tip of the pen is selected. Tapping again on the selection will expand the selection to include the strokes in the next cluster on the tree above. This is repeated until the desired cluster is reached. To avoid excessive tapping, the system can employ a “hold” command, whereby with the pen held down on a portion of ink, the selection progressively expands until the user lifts the pen up from the writing surface.

There are two nice properties of the structure given by our clustering method. First, expansion of a selection is rapid and non-linear, because the cluster sizes increase in this manner as one moves up the tree. Furthermore, since at any given state in the tree, there is only one way to expand up, there is no ambiguity that must be resolved by the user. Hence, the interaction can be done in a modeless or one-and-half mode style, and complex multiple-mode style (*e.g.* Tivoli[9], PerSketch [15]) is not necessary.

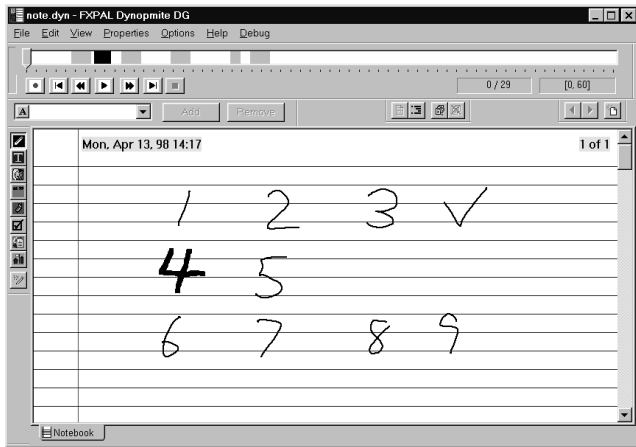


Figure 1. A note page with ink and audio. The timeline at the top displays audio activity in gray. There are eleven strokes, with the “4” consisting of two strokes. The “4” in bold was selected and marked by the user, and the corresponding cluster on the timeline was automatically marked.

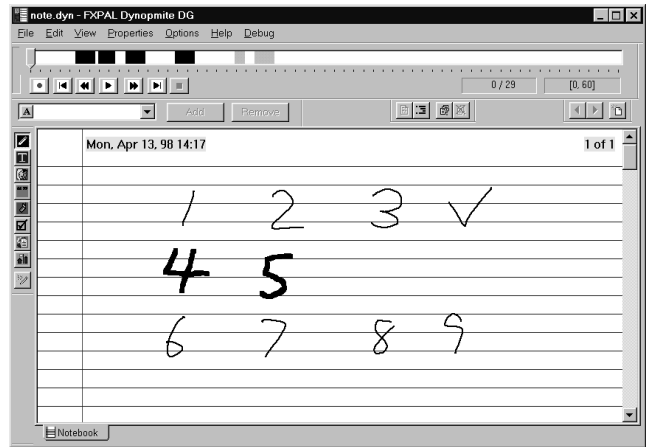


Figure 2. The second line “4 5” in bold was selected and marked, and the corresponding cluster on the timeline was automatically marked.

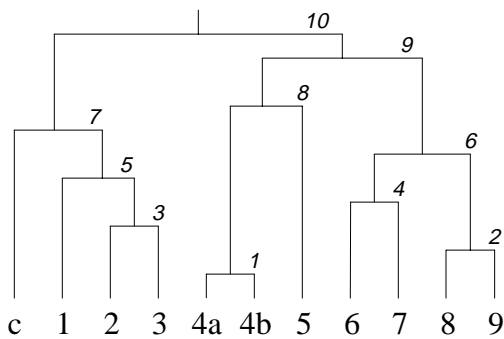


Figure 3. The tree for hierarchical agglomerative clustering of the ink strokes in figures 1 and 2. The “c” is the check mark, and 4a and 4b are the two strokes that make up the “4”.

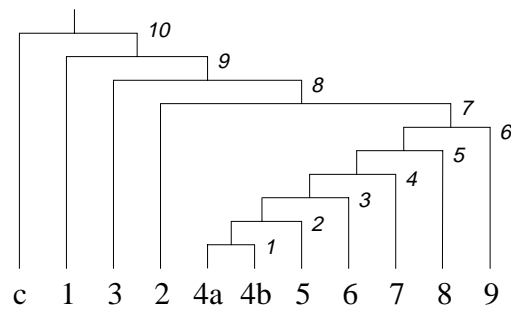


Figure 4. The tree for a naive hierarchical clustering induced by the ink stroke 4a in Figures 1 and 2. Note the “twisting” of the “2” and “3”.

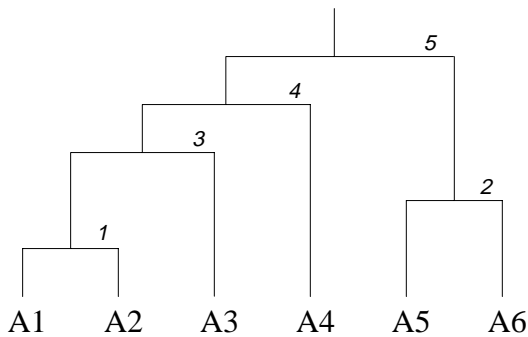


Figure 5. The tree for hierarchical agglomerative clustering of the audio segments in figures 1 and 2. The sequence of segments as ordered by time are labeled {A1, A2, ..., A6}.

For the example in Figures 1 and 2, interactive selection of the handwritten digits proceeds as follows. When the user taps on one of the strokes in the “4”, say 4a, this stroke is selected. Tapping again on the selection expands the selection to include 4b, and the result is shown in Figure 1. Another tap expands the selection to the whole line consisting of the digits “4” and “5”, shown in figure 2. The next tap will expand the selection to include the bottom line “6 7 8 9”, and one more tap will select the all the strokes on the page.

### INTEGRATION OF INK AND AUDIO GROUPING

In systems such as Dynamite, where the ink and audio streams are recorded synchronously, it is often desirable to have a joint selection of ink and its corresponding audio, and vice versa. For example, in Figure 1 when the user marks a group of ink notes, the corresponding group of audio notes is marked. A straightforward way to determine the audio corresponding to a given ink group would be to determine the earliest and latest times for ink strokes in the group, and use these times as the audio boundaries. Since these times may not necessarily fall during silences in the audio, the times could be adjusted forward and backward to obtain the minimum segment of audio bounded by silence that covers the time interval for the ink.

For a more robust grouping of the audio, the audio segment could be computed using the audio grouping. The audio corresponding to the ink cluster would be matched to the smallest audio cluster that contains the ink. Similar methods could be used for finding a portion of ink corresponding to a selected audio group.

Going back to the example of Figures 1-5, we describe the effects of this joint clustering. Selecting the “4” plays {A2}, which plays from the beginning of the second clause. Selecting the whole line “4 5” plays the cluster {A1, A2, A3, A4}, which plays from the beginning of the sentence. In contrast, with the straightforward method described above, *both* groups “4” and “4 5” are bounded on the left by {A2}, and plays from the beginning of the second clause.

### APPLICATION TO BROWSING NOTES

Browsing through the notes is facilitated by integration of ink and audio. The ink can be used as indexes to the audio. By correlating time-stamped ink strokes to the recorded audio, the user can select ink strokes and play the corresponding point. There have been many systems with this feature, beginning with NoTime [7]. One significant issue is that human note-taking often lags behind the audio recording, and there is a question of how far to back up the play point. Some systems leave this adjustment to the user by providing only tape-deck style controls. Other systems offset the play point by fixed amount of a few seconds. In the Audio Notebook [17], the nearest phrase boundary preceding the selected ink is automatically detected, and play begins there rather than abruptly in the middle of a phrase. In SpeechSkimmer [2], audio browsing is supported by providing methods to fast forward and jump

to likely boundaries determined by pitch and silence. These methods of automatic adjustments are not entirely satisfactory because the user will wish to vary the amount of adjustment.

With the integration of ink and audio groups obtained from hierarchical clustering, the play point adjustment is adaptive depending on the amount of ink selected. While an audio group provides a span with both a start and end point for play, it is better to use only the start point and let the user control the play duration, according to a recent study on browsing speech [18]. (Of course, it is necessary to have the end points of the audio segments for clustering.) Selecting a single stroke would play from the beginning of the audio group corresponding to the stroke, and selecting the word containing this stroke would move the play point relatively farther back to match a higher level audio group. Selecting the line containing this stroke would take the play point back even farther. This adjustment of play point via the ink selection operation is helpful because it allows the user to maintain focus on the ink notes and reduces the need to go to another part of the interface and fiddle with the tape-deck controls.

### EXAMPLE OF USE

We are in the process of evaluating these techniques by letting people use them to take and make notes. The Dynamite system is available to the lab for use in staff meetings, presentations, and Japanese language classes.

We give a real example showing how these techniques work in practice. This is from a weekly Japanese class, see Figure 6. There are one and a quarter pages (when printed on standard 8.5 by 11 paper) of written notes, and about half a page is visible in the scrolling window. There is an hour of audio recorded from the class, and the timeline displays ten minutes at a time. From our experience, we found ten minutes to be about the right granularity. Scaling the audio to fit the timeline confused users.

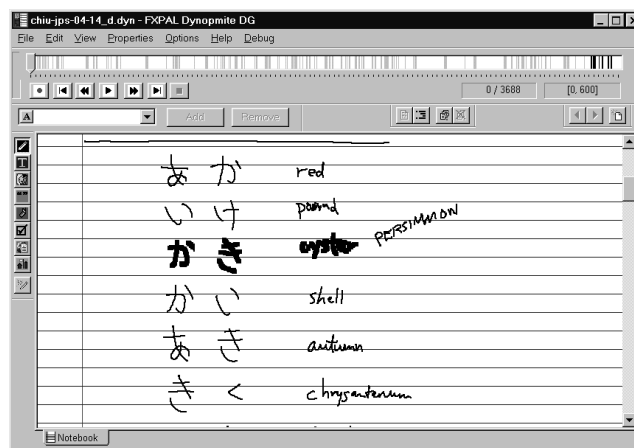


Figure 6. A note page from a Japanese class. The highlighted Japanese characters and the word “oyster” correspond to the dark bands on the far right of the timeline.

A few days later, the student reviewed the notes. The two Japanese characters on the third line were selected by tapping four times on the first character, and pressing the PLAY button on the timeline starts play at the beginning of the corresponding audio group. After listening for ten seconds, the student realized that it was playing in the middle of a discussion and decided to go back further. By tapping again one more time on the selection, the selection expanded to include all the ink strokes of the word "oyster". The PLAY button was pressed and the corresponding cluster of audio was played. This cluster for the line has a starting play point 42 seconds earlier than the lower level cluster for the two Japanese characters. After listening for a minute, the student discovered something he had missed in class: This pair of Japanese characters can have two different meanings ("oyster" or "persimmon") depending on how the characters are stressed when pronounced. The annotation "persimmon" was added to the notes. The result is shown in Figure 6.

Next, the student wanted to change the color of the annotation. He selected it without difficulty. The skewed handwriting of "persimmon" was correctly grouped by the hierarchical agglomerative clustering algorithm. The reason is that the annotation was written later, and the time component is the dominant term in the grouping computation. Finally, he selects a color for the selection from the toolbar palette.

### CONCLUSION

The dynamic grouping technique described in this paper contributes to developing more robust user interactions with digital ink and audio data, particularly in freeform note-taking systems. By grouping and structuring the ink and audio into units that are meaningful for user interaction, we provide a flexible way to select and mark parts of ink and audio. Selections can be expanded rapidly in a non-linear fashion. By integrating the ink and audio groups, we introduce a more flexible browsing technique that adaptively adjusts the starting play point of an ink selection according to the amount selected. We gave an example to illustrate these techniques in practice.

The grouping technique is based on hierarchical agglomerative clustering with a different distance measure for each medium. With an ink distance measure based on time and distance, the algorithm performs better than common time-based algorithms. We have implemented it in an object-oriented way that is extensible to other media such as video. The simple hierarchical framework is suitable for simple modeless and one-and-a-half mode styles of interaction, unlike more complex schemes that do multiple interpretation but also require multiple modes.

### FUTURE WORK

There are two interesting directions we are planning to explore. One is to incorporate other features into the distance measure. For ink, strokes with the same color or line width are defined to be "closer". For example, a word written in one color will be more accurately grouped from

nearby strokes of a different color. For audio, speaker turn information can be utilized so that primitive segments that fall within the same speaker turn are defined to be closer, and better grouping is achieved. The other direction we are planning future research on is to apply dynamic grouping to other media such as video. In this case, one possible way to define the distance measure is by looking at the difference of color histograms between video frames, which is a standard technique for segmenting video.

### ACKNOWLEDGMENTS

We would like to thank our colleagues Bill Schilit, Gene Golovchinsky, and Joe Sullivan for their contributions to early interfaces for clustering; and several of our colleagues at FXPAL for comments on the paper.

### REFERENCES

1. *aha! InkWriter Handbook*. Microsoft Corp, aha! Software, Mountain View, CA, 1993.
2. Arons, B. SpeechSkimmer: a system for interactively skimming recorded speech. *ACM TOCHI*, 4 (1), 1997. ACM, New York, pp. 3-38.
3. Degen, L., Mander, R., and Saloman, G. Working with audio: integrating personal tape recorders and desktop computers. *Proceedings of CHI'92*. ACM, New York, pp. 413-418.
4. Edmonds, E., Moran, T.P., and Do, E. Interactive systems for supporting the emergence of concepts and ideas. *SIGCHI Bulletin*, 30 (1), 1998. ACM Press, pp. 24-25.
5. Gross, M.D., and Do, E. Demonstrating the electronic cocktail napkin: a paper-like interface for early design. *Companion to CHI'96*. ACM, New York, pp. 5-6.
6. Kimber, D., Wilcox, L., Chen, F., and Moran, T.P. Speaker segmentation for browsing recorded audio. *Companion to CHI'95*. ACM, New York, pp. 212-213.
7. Lamming, M., and Newman, W. Activity-based information technology in support of personal memory. Technical Report EPC-1991-103, Rank Xerox, EuroPARC, 1991.
8. Landay, J.A., and Myers, B.A. Interactive sketching for the early stages of interface design. *Proceedings of CHI'95*. ACM, New York, pp. 43-50.
9. Moran T.P, Chiu, P., van Melle, W., and Kurtenbach, G. Implicit structures for pen-based systems within a freeform interaction paradigm. *Proceedings of CHI'95*. ACM, New York, pp. 487-494.
10. Moran T.P, Chiu, P., and van Melle, W. Pen-based interaction techniques for organizing material on an electronic whiteboard. *Proceedings of UIST'97*. ACM, New York, pp. 45-54.

11. Moran, T.P., Pedersen, E.R., McCall, M.K., and Halasz, F.G. *Wiping Metaphor as a User Interface for Operating on Graphical Objects on an Interactive Graphical Display*. U.S. Patent 5,548,705, August 20, 1996.
12. Moran, T.P., Pedersen, E.R., McCall, M.K., and Halasz, F.G. *Time-Space Object Containment for Graphical User Interface*. U.S. Patent 5,404,439, April 4, 1995.
13. *Newton MessagePad Handbook*. Apple Computer, 1993.
14. Pedersen, E.R., McCall, K., Moran, T.P., and Halasz, F.G. Tivoli: An Electronic Whiteboard for Informal Workgroup Meetings. *Proceedings of InterCHI'93*. ACM, New York, pp. 391-398.
15. Saund, E., and Moran T.P. A perceptually-supported sketch editor. *Proceedings of UIST'94*. ACM, New York, pp. 175-184.
16. Shipman, F.M., and Marshall, C.C. *Formality considered harmful: experiences, emergent themes, and directions*. Technical Report ISTL-CSA-94-08-02, Xerox Palo Alto Research Center, 1994.
17. Stifelman, L. *The Audio Notebook: Paper and Pen Interaction with Structured Speech*. PhD Thesis. MIT, 1997.
18. Whittaker, S., Hirschberg, J., and Nakatani, C. H. Play it again: a study of the factors underlying speech browsing behavior. *Summary of CHI'98*. ACM, New York, pp. 247-248.
19. Whittaker, S., Hyland, P., and Wiley, M. "Filochat: handwritten notes provide access to recorded conversations". *Proceedings of CHI'94*. ACM, New York, pp. 271-276.
20. Wilcox, L. D., Schilit, B. N., and Sawhney, N. Dynamite: A Dynamically Organized Ink and Audio Notebook. *Proceedings of CHI'97*. ACM, New York, pp. 186-193.

21. *Zaurus Operation Manual*. Sharp Corporation, 1996.

22. Zellweger, P., Terry, D., and Swinehart, D. An overview of the Etherphone system and its applications. *Proceedings of the 2nd IEEE Conference on Computer Workstations*, 1988, IEEE, New York, pp. 160-168.

## APPENDIX

We briefly describe an object-oriented implementation of the algorithm as follows. The classes `ClusterInk` and `ClusterAudio` are derived from a base class `ClusterMaker`. For each medium, the system keeps track of a list of primitive elements where each element has an index. From this, the associated distance matrix, in which the entry  $a[i][j]$  is the distance from the  $i$ -th element to the  $j$ -th element, is computed by `ComputeDistanceMatrix()`. Since the distance depends on the type of medium, this is a virtual function. Taking the distance matrix, `MakeCluster()` performs the hierarchical agglomerative clustering as described above and caches the result in a tree. When a primitive element is targeted for grouping, the groups in the levels directly above it in the cached tree are accessed by `GetTargetCluster()`.

```
class ClusterInk : ClusterMaker;
class ClusterAudio : ClusterMaker;

class ClusterMaker {
    virtual ComputeDistanceMatrix(
        Matrix *distMatrix);
    MakeCluster(Matrix distMatrix);
    GetTargetCluster(
        int targetIndex,
        int targetHeight,
        LeafIndexList *clusterList);
};
```