Fuzzy Spatial Reasoning in Psycholinguistic Experiments on Braille Reading

Surangi Rodrigo, Department of Psychology, University of Auckland, New Zealand
Hans W. Guesgen, Computer Science Department, University of Auckland, New Zealand

Abstract— **This paper introduces a system for data collection and analysis of braille reading experiments undertaken by psycholinguists. The system is intended as a replacement for the conventional collection of data by analogue video and the analysis of it by watching the video tapes. It takes a braille reading movie and converts it into an abstract representation by taking into consideration the position of the reading finger and user-defined regions of the text. The spatial analysis module used by the system applies a set of efficient algorithms for reasoning about space and time, which allow for fuzziness in the spatial information (but do not exploit fuzzy logic in the traditional sense).**

Keywords— **Fuzzy spatial reasoning, braille reading.**

I. Introduction

For visually impaired people, braille is the standard text-encoding method to access written material. Since braille reading is based on touch rather than vision, it is about 3-4 times slower than reading with the eye. It is therefore even more important to format the text in such a way that reading the text is easy and quick. Psycholinguistic experiments with braille readers can give valuable insights in this respect. However, such experiments are usually very labour intensive, partly because of the nature of the braille system.

The braille system employs groups of dots to represent printed letters and numbers. The system's basic braille cell consists of six dots grouped in two vertical columns of three dots each, which are embossed on paper. A braille cell is approximately 5mm high and 2.5mm wide. The spaces between words are of the same size as a braille cell. From the basic cell, 64 different dot patterns (including the empty pattern) can be formed, which are easily identifiable to the touch.

The braille code has two levels of complexity, termed Grade-I and Grade-II. The Grade-I braille is a direct translation of print letters to braille cells, i.e., 26 of the possible dot patterns represent letters of the alphabet, while others are used for punctuation marks, markers of capitalization et cetera. Many of the 64 possible dot patterns therefore remain unutilized. Grade-II braille exploits this fact by using the unutilized cell patterns to denote contractions of letters that are frequently used together. An example of a contraction is the cell standing for the letter sequence *ing*. Instead of using the three individual cells for *i*, *n* and *g*, a single cell denotes that combination of letters in words such as *sing* and *doing*. As well as contractions, Grade-II braille employs abbreviations or short-form words; some entire words are represented by a single cell. In Grade-II braille the word *people*, for example, is symbolized by the use of the *p* cell in isolation (with a space on either side). These modifications allow Grade-II braille to be represented much more economically than Grade-I braille. A typical braille page contains 40 braille characters per line and 25 lines. One print page is equivalent to about one and a half pages of Grade-II Braille, which makes full use of contractions.

Experiments with braille readers have shown that perception of the characters of braille relies on active exploration by the distal pad of at least one finger [1]. The size of the braille cell is such that only one character at a time can be read by a single finger, which means that the perception of the characters in a word is sequential. A sighted reader can perceive several characters in a single fixation. Other experiments [2] gave insights into the effect of local ambiguity during braille reading, or they explored the way that braille readers deal with structural ambiguity [1].

Many of the experiments were conducted by recording the hand motion of the blind readers. This was achieved by monitoring the finger movements with a recording device such as the one described in Noblet et al. [3], which combines a solid state video camera (Hitachi KP120U), a control screen and a central measure unit interfaced to an Apple-II GS microcomputer. The central unit generated a flash every 40ms on a small light-emitting diode pasted on the nail of the reading finger and computed for each flash the lateral and vertical coordinates of the finger location through analysis of the image captured by the camera. Setups like this have limitations.

Unlike for experiments involving sighted readers, there are only a few subjects available to conduct experiments in braille reading, so it is very important to capture results of various aspects of the experiment during the data collection phase of the experiment. Therefore data collection by video recording has been adopted recently [1]. However, this recording method produces vast amounts of video data, which cannot be handled efficiently any more by watching the tapes to find the times spent by each reader in various zones of the sentences. It seems that this is definitely an area where the aid of computers and computer programs would encourage further research in the comprehension of braille for visually-impaired readers.

In this paper, we introduce a system called SABRED for the spatial analysis of braille reading recordings. The idea is to attach a mark in the form of a bright spot to the subject's reading finger and to use this spot to automatically determine the position of the reading finger. In addition, the experimenter can specify regions in the text that are to

0-7803-4863-X/98 $10.00©1998 IEEE

be used to analyze the finger movement. The system steps through the frames of the recorded video and determines the reading finger position with respect to the regions. The rest of this paper is organized as follows. In the next section, we will illustrate various aspects of the setup for the SABRED system, like making a movie, converting it into QuickTime format, analyzing the bright spot and specifying the regions. Then we introduce the algorithms used to analyze the movie. Finally, we will talk about a user interface for interpreting the results, and we will summarize the main results.

II. Setup

A braille reading QT movie shows a subject trying to decode braille in a tactile sense. All that is visible in the movie are the sheet of the braille text and the hand(s) of the reader. The experimenter may also put the printed words, above the braille words, as done previously when video editing equipment was used for the analysis. This may come in use if the experimenter later wants to view these braille reading movies on video, independently of the computer analysis.

Prior to the start of the experiment, subjects would be asked to indicate which finger of their preferred hand they would use for reading. A brightly colored sticker would then be placed on the nail of this reading finger. This bright spot is used to identify the reading finger during the analysis. The bright spot is basically a round or oval (as this closely represents the finger nail) sticker over the finger nail of the reading finger, and that is unique to all the frames.

There are two ways to get a movie of a braille reading session. The one is to use a video camera and then digitize the analogue signal that is produced. This procedure requires specialized computer hardware. The second method is by the use of a digital movie camera. At present it is far cheaper, in terms of money, to obtain a good analogue representation of the braille reading session and then digitize that into a form of a QT movie. We decided to use the HiPAL visual presenter, since it is an ideal piece of hardware for the purpose of recording the experiment. The visual presenter is mounted directly above the flat surface that is used to hold the braille text. This way we get a perfect aerial view of the whole braille decoding process.

The SABRED system has been implemented on a Power Macintosh 8500, which comes equipped with a completely revamped audio-visual (AV) subsystem that supports full-motion video capture, display, editing, and output at near-broadcast quality. The Apple Video Player software that is built in with the Macintosh operating system can be used to capture and create a QT movie.

The experimenter can use the QT movie and define the regions that they would like analyzed. The regions are specified by providing two mouse clicks per region; one to indicate the start of the region and the other to indicate the end of the region. The information corresponding to the regions would then be stored in a data structure and later used during the analysis of the reading process.

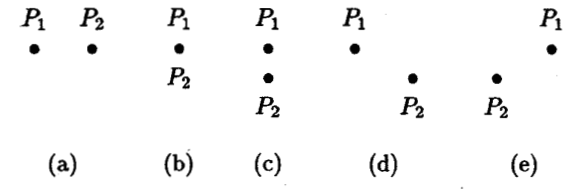

Fig. 1. Various positions of the two points.

III. Analyzing Braille Reading

In this section we will look at various aspects of analyzing braille reading movies, including the underlying concepts and algorithms.

A. Ordering in Space

Essential factors in analyzing braille reading are the concepts of space and ordering in space. Due to the nature of braille reading movies, we can restrict ourselves to 2- dimensional space, i.e, the points that are considered lie in a plane. Given two points in space, we need to compare the two points to see if they are to be considered identical or not. If the points are considered to be different, then we need to establish an ordering on the points.

Equal in Space— Given two points, $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, in space, we define P_1 to be equal to P_2 in space ($P_1 \approx P_2$), if the following holds:

$$(x_1 = x_2) \wedge (y_1 = y_2)$$

The equal-in-space predicate holds if the horizontal as well as the vertical coordinates of the two points are equal. Part (b) of Figure 1 applies to this.

Less than in Space— Given two points, $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, in space, we define P_1 to be less than P_2 in space ($P_1 \prec P_2$), if the following holds:

$$[(y_1 = y_2) \wedge (x_1 < x_2)] \vee (y_1 < y_2)$$

The less-than-in-space predicate holds if one of the following is true:

1. The vertical coordinates of the two points are equal and the horizontal coordinate of P_1 is less than that of P_2.
2. The vertical coordinate of P_1 is less than that of P_2.

Parts (a), (c), (d), and (e) of Figure 1 are graphical displays of rules 1 and 2 above. Part (a) describes rule 1; parts (c), (d), and (e) describe rule 2.

Greater than in space— Given two points, $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, in space, we define P_1 to be greater than P_2 in space ($P_1 \succ P_2$), if the following holds:

$$[(y_1 = y_2) \wedge (x_1 > x_2)] \vee (y_1 > y_2)$$

The greater-than-in-space predicate holds if one of the following is true:

1. The vertical coordinates of the two points are equal and the horizontal coordinate of P_1 is greater than that of P_2.
2. The vertical coordinate of P_1 is greater than that of P_2.

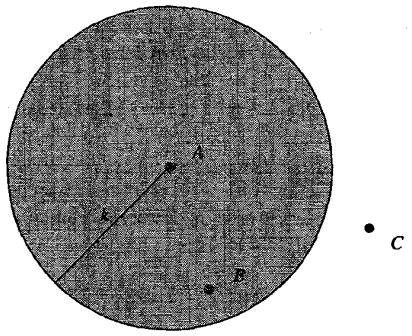

Fig. 2. Fuzzy region of A.

Figure 1 does not explicitly contain displays for this, but if the two points are interchanged in the diagram, then parts (a), (c), (d), and (e) would apply to this. Part (a) would then describe rule 1 and parts (c), (d), and (e) would describe rule 2.

Using the two comparisons defined earlier on equal in space and less than in space, we can say that P_1 is greater than P_2 in space if P_1 is not equal to P_2 in space and P_1 is not less than P_2 in space:

$$P_1 \succ P_2 \Longleftrightarrow \neg(P_1 \approx P_2) \wedge \neg(P_1 \prec P_2)$$

The comparison of points as described above in terms of P_1 less than P_2 in space, and so on is fine in theory. However, when it comes to implementing them there is one practical difficulty. This has to do with the fact that in the SABRED program it is necessary to take the reading finger locations, which are extracted from the braille reading movie and compare those points to the region boundaries, which are supplied by the experimenter by defining the regions. Even though it may be the intention of the experimenter to be very precise when defining regions, this may not always be possible due to various reasons. Due to this it is necessary to allow for some tolerance around the first point when comparing it with a second point. This tolerance defines a fuzzy region by drawing an imaginary circle around the first point with the radius equal to the defined tolerance value. This tolerance value is adjustable by the experimenter. Generally, smaller fuzzy regions give more accurate results and larger fuzzy regions less accurate results, as timing would not be accurate.

When using fuzzy regions, the two points, P_1 and P_2, would be equal in space if P_2 would lie in the fuzzy region of P_1, given some tolerance value. Figure 2 is an illustration of fuzzy regions with a tolerance value of k. Point B is equal to point A in space since B is in the fuzzy region of A. However, point C is not equal to A in space since C is not in the fuzzy region of A.

With the use of fuzzy regions, when checking for equality of points (in space), we now check to see if the second point is in the fuzzy region of the first point, as described above. Similarly, the other comparisons are also affected by fuzzy regions.

A 3-dimensional space-time point $\langle v, t \rangle$ is a combination of a 2-dimensional spatial region v, given by a start point (x_1, y_1) and an end point (x_2, y_2), together with temporal information, given by a time interval. The 3-dimensional space-time points are identified by a region number.

B. The CompareColors Algorithm

The algorithm CompareColors compares the pixel color obtained from the frames of the braille reading QT movies with that of the actual color, which is the color specified by the experimenter, to be the color of the reading finger. If the pixel color is found to be same as the actual color, to the extent of the tolerance value which is also supplied by the experimenter, then the algorithm returns true, otherwise it returns false. This is a supporting algorithm for the FishBone search algorithm.

Both pixel color and actual color are defined in terms of the HSV color model. It is recommended that the reading finger has a bright spot obtained from a pure hue. Further, it is encouraged to have a pure hue of a primary color for simplicity of the experiment. Blue proved to be a far better bright spot than red or green.

In the CompareColors algorithm, there is first a comparison to see if the saturation of the HSV color is at least the defined (low) value, which is supplied by the experimenter. Higher values for saturation indicate that the purity of the color is high. If this test is not satisfied, then the comparison of the two colors fails, because this implies that the color is a different shade, tint, or tone of the reading finger's bright spot and may correspond to something completely different on that frame of the QT movie.

When comparing the hue values to see if a pixel represents the bright spot, we must take a tolerance value into consideration to allow for a fuzzy region. The reason for having these fuzzy regions is because the bright spot that was initially chosen can be significantly different, in its HSV representation, to that found on the frame of the QT movie. One factor which could contribute to this change is the various lightings that are introduced during the video capture phase of the experiment. In order to cater for these variations in the color, the experimenter is able to define tolerance levels to accommodate fuzzy regions. Large tolerance values imply large fuzzy regions and small tolerance values imply small fuzzy regions.

C. The FishBone Search Algorithm

FishBone search is an efficient search algorithm in a 2-dimensional space. The motivation behind the development of this search algorithm is that, given a QT movie containing a large number of frames, we need to go through each of these frames and then find the center pixel of the bright spot of the braille reading finger. This algorithm uses the previously discussed CompareColors algorithm. FishBone search starts in a fashion similar to linear search, but once the first pixel of a matching color is found, the efficiency of the algorithm increases. Instead of simply continuing in a linear fashion from left to right and then top to bottom along the pixels until the center pixel of the read-

ing finger is found, FishBone search applies a heuristic to cut the search space down to very small area. The heuristic proceeds in the form of a fish-bone, and therefore the name FishBone search.

Once the first pixel of the bright spot is found, the algorithm searches for the center pixel of the bright spot, both along the x-axis and along the y-axis. There is also a redundancy check to make sure that the first pixel found is not associated with pixel noise. The algorithm also takes care of the fact that the bright spot may be a square shape rather than a round or an oval.

Once the center pixel of the reading finger is found, that information, together with the current frame number and the time elapsed since the start of the movie, are entered into a record of the data structure called FrameInfo. The information stored in FrameInfo will later be used for the analysis, as it is much faster to work with a data structure containing the necessary information than to manipulate the frames of a QT movie.

For each frame of the QT braille reading movie, the routine FishBone search is called to find the coordinates of the reading finger. Instead of beginning each search from the top left corner of each frame, we begin the search for the ith frame in a local area to the $(i-1)$th frame. The fact that the finger may move backwards, instead of forwards from the previous position is also taken into consideration.

The preconditions for the FishBone search algorithm are, firstly, a QT movie containing the braille reading movie with a bright spot distinct to the whole movie and, secondly, a sensible predefined saturation figure, together with a tolerance value for the hue component of the HSV color. The postcondition of the FishBone search algorithm is the FrameInfo data structure.

D. The FineRegions Algorithm

The FineRegions algorithm has been developed for the analysis of the braille reading movie. The idea behind the FineRegions algorithm is to go through each frame of the QT movie, locate the position of the reading finger (in terms of x and y coordinates), and then identify the region the reading finger falls into from the list of regions that have been defined by the experimenter. Actually, this corresponds to going through each frame of the FrameInfo data structure since this contains the data extracted from the braille reading QT movie and then identifying the region the reading finger falls into from the list of regions that has been defined by the experimenter, which is contained in the data structure called RegionLinkList.

Our aim here is to find the sequence of regions (with backward movement) that the reading finger moved when decoding the braille characters. In order to keep track of this sequence of regions, another data structure called RegionInfo is maintained. Each record of RegionInfo consists of an integer representing the region number and a space-time point. The space-time point consists of two long integers representing the time when the region was entered by the reading finger and the time when it exited the region, and two points to represent the start point of the region and the end point of the region.

The preconditions of the FineRegions algorithms is that the regions, defined by the experimenter, must be ordered so that the coordinates of the ith region must be less than in space than the coordinates of the $(i+1)$th region. One exception to this rule is that the end-point of the ith region may be equal in space to the start-point of the $(i+1)$th region.

The postcondition of the FineRegions algorithm is the RegionInfo data structure which contains the space-time points together with an integer representing the region number. A lot of information can be extracted by the experimenter from the information of the RegionInfo data structure. The order of the regions can tell about the nature of the braille decoding process, together with when and where regressions were present. Timing information can be obtained between any two regions by simply subtracting the start time of the first region from the end time of the second region.

IV. Conclusion

In this paper, we introduced the SABRED system for the computerized analysis of braille reading movies. A first evaluation of the system has shown that braille reading movies can be evaluated more efficiently by using the SABRED system for performing a preanalysis of the movie instead of manually locating the position of the reading finger and associating it with significant regions of the text. In the addition to the basic system, we developed a user interface in the form of a prototype.

The aim of computer-based prototyping is to provide a version at an early stage of development of the system with limited functionality so that users can interact with it [4]. We developed a prototype of the user interface for interpreting the results using HyperCard, which is a Macintosh-based multimedia package for combining text, graphics, sound and video. HyperCard allows the capability of developing a quick graphical user interface (GUI). It is a good tool for prototyping interfaces and for producing impressive visualizations and presentations. An attractive feature of HyperCard is that no programming skills are needed to create an interface.

Future work on the SABRED system includes the further development of the user interface. Beyond that, we are aiming at the development of advanced modules for the analysis of braille reading movies, like a module for the automatic detection of regressions for instance.

References

[1] N. Dickinson and J. McAllister. Sentence processing in braille: Effect of text format. Internal report, Department of Psychology, University of Auckland, Auckland, New Zealand, 1995.

[2] P. Mousty and P. Bertelson. Finger movements in braille reading: The effect of local ambiguity. *Cognition*, 43:67–84, 1992.

[3] A. Noblet, H. Ridelaire, and G. Sylin. Equipment for the study of operating processes in braille reading. *Behavior Research Methods, Instruments and Computers*, 17:107–113, 1985.

[4] J. Preece. *Human-Computer Interaction*. Addison Wesley, Readings, Massachusetts, 1994.