Boston University Computer Science Technical Report No. 2005-007

SymbolDesign: A User-centered Method to Design Pen-based Interfaces and Extend the Functionality of Pointer Input Devices

Margrit Betke*, Oleg Gusyatin, and Mikhail Urinson

Department of Computer Science, Boston University,
Boston, MA 02215, USA

February 26, 2005

Abstract

A method called "SymbolDesign" is proposed that can be used to design user-centered interfaces for pen-based input devices. It can also extend the functionality of pointer input devices such as the traditional computer mouse or the Camera Mouse, a camera-based computer interface. Users can create their own interfaces by choosing single-stroke movement patterns that are convenient to draw with the selected input device and by mapping them to a desired set of commands. A pattern could be the trace of a moving finger detected with the Camera Mouse or a symbol drawn with an optical pen. The core of the SymbolDesign system is a dynamically created classifier, in the current implementation an artificial neural network. The architecture of the neural network automatically adjusts according to the complexity of the classification task. In experiments, subjects used the SymbolDesign method to design and test the interfaces they created, for example, to browse the web. The experiments demonstrated good recognition accuracy and responsiveness of the user interfaces. The method provided an easily-designed and easily-used computer input mechanism for people without physical limitations, and, with some modifications, has the potential to become a computer access tool for people with severe paralysis.

*Contact information: betke@cs.bu.edu, www.cs.bu.edu/faculty/betke.

1 Introduction

Portable miniature computers with pen- or pointer-based interfaces, such as personal digital assistants (PDAs), tablet computers, and wearable computers, have become popular in recent years. Research communities, such as the European Research Consortium for Informatics and Mathematics "User Interfaces for All" (ERCIM UI4All) [17], are working on paradigms for universal access to these devices. A paradigm that provides a theoretical framework for the principles, assumptions, and practices of accessible and universal design of pen- or pointer-based interfaces is proposed here. It focuses on the needs of both people whose physical conditions affect their computer use and limit their performance, as well as people without these characteristics. Adopting universal design principles for all users is beneficial because it can reduce fatigue, increase the rate of communication, decrease errors, and decrease learning time [10]. Accessible design of pen- or pointer-based interfaces calls for user-centered design that

- is simple and intuitive,
- minimizes user input errors,
- minimizes recognition errors of the system,
- allows for flexibility and includes redundancy. This means it provides choices in features and modalities and enables the user to customize settings whenever possible to accommodate different user preferences or abilities [10].

This paper presents a method called "SymbolDesign" to create user-centered pen- or pointer-based interfaces that adhere to the above paradigm. Using this method, users can create their own interface based on sets of commands that correspond to symbols or gestures that they designed themselves. The interfaces are simple and intuitive, minimize errors, and allow for flexibility. Instead of requiring the user to memorize and practice predefined symbols, SymbolDesign creates an interface that "understands" the user-designed spatio-temporal patterns and "learns" how the user typically draws them.

The SymbolDesign method can be used with various pointer- or pen-based input devices. Since it does not need information about the type of pointer device used to control the computer, it provides a general extension to the capabilities of the device. A survey by Meyer [40] gives an overview of the technology of pen input devices and describes interface products available in 1995. Currently available products are, for example, Microsoft's Tablet PC [45] and ZyonSystems' i-Pen [62], which both include handwriting recognition software, and the Camera Mouse [4], an interface used by adults and children with severe motion impairments.

People who are severely paralyzed and nonverbal from cerebral palsy, stroke, multiple sclerosis, amyotrophic lateral sclerosis, or brain injury often depend on the computer to facilitate or enrich their communication with friends, family, and care givers. Alternative pointing devices allow quadriplegic users, who have a limited range of voluntary motions, to control the computer with, for example, head or tongue movements [3, 11, 24]. The Camera Mouse tracks the computer user's movements with a video camera and translates them into the movements of the mouse pointer on the screen [3]. Body features such as the tip of the user's nose or finger can be tracked. Another mouse-substitution system for people with severe disabilities is Eagle Eyes [11], a gaze estimator based on measuring the electro-oculographic potential [59]. Five electrodes are placed on the face and used to estimate gaze direction. The gaze direction is then converted into mouse pointer coordinates. Other systems provide mouth-actuated joysticks, infrared head-pointing devices, and head-mounted gaze estimators as alternative mouse-input devices (e.g., [1, 12, 24, 27, 30, 31, 33, 38, 39, 41, 53]). There are some users, whose physical conditions are even more limiting – they may only be able to use eye blinks for communication [2, 21].

People with or without physical conditions that affect their computer use may benefit from a system, such as SymbolDesign, that allows them to design a small set of symbols representing commands, for example, to select customizable "shortcut-like" functions, use an internet browser, or play a computer game. These symbols may be gestures painted with the user's finger or hand and detected with a camera-based computer interface, such as the Camera Mouse [3], the Finger Counter [7], an augmented desk interface [43], or a "gesture spotting" system [58]. The symbols may also be entered into the computer with a traditional mouse, a wireless gyroscopic mouse, an optical pen, or a stylus. With the SymbolDesign method, the computer user can choose single-stroke movement patterns that are convenient to create with the selected input device and map them to the desired set of commands. For example, the user may decide to choose the symbol O, drawn by a circling movement, to represent the selection command or a zigzag pattern to indicate deletion of a word.

It is also noteworthy that the interface does not require additional hardware, works with a small amount of processing resources, and does not need to use screen space, which is so valuable for miniature computers (an inconvenient use of screen space, for example, would be a virtual keyboard). Drawing the spatio-temporal patterns created with the SymbolDesign method does not demand a high precision in pointer or pen control.

The SymbolDesign method creates interfaces with a relatively small number of commands (1–20). Fast and reliable access to even a small number of commands can be useful – they can replace the functionality of the traditional computer mouse and provide some of the functionality of the traditional keyboard, e.g., input of letters, digits, and keyboard shortcuts. PDAs can provide computer access with a few customizable buttons. Camera-based interfaces for quadriplegic users provide computer access by recognizing a small number of patio-temporal patterns, for example, the three gaze directions (left, right, and center) of the EyeKeys system [39] or the eye blinks or winks of the BlinkLink [21].

Pen-based handwriting systems [44, 52, 55, 61] also provide solutions for recognition of spatio-temporal patterns. They generally work with much larger alphabets than the interfaces created with the SymbolDesign method, but they do not have the same flexibility as the SymbolDesign method in allowing the users to specify his or her own symbols. This user-centered design principle is particularly important if the intent is to provide an interface for people with motion impairments who choose the symbols according to their gesturing abilities.

Existing symbol or shorthand recognition methods [20, 25, 32, 37, 60] are restrictive in the sense that the user has to use predefined symbols, whose meaning often is not intuitive and that take time to learn. Several studies have been reported where users were asked to learn and test a specific set of characters: Goldberg and Richardson [20], for example, invented and evaluated the Unistroke alphabet. In this alphabet, many common letters, such as E, A, T, I, and R, are assigned to simple straight-line strokes. Goldberg and Richardson's experiments showed that pauses between single-stroke patterns were significant, a property that we also observed and that our interfaces use for distinguishing consecutive input symbols.

MacKenzie and Zhang [37] evaluated the usability of the "Graffiti" alphabet, which also contains single-stroke characters that only slightly resemble real letters and digits. Sears and Arora [47] compared the relative effectiveness of the Graffiti alphabet and the Jot alphabet, which also contained only single-stroke characters. Isokoski and Raisamo [25] developed and evaluated an alphabet with single-stroke characters that were based on sequences of two to four horizontal and vertical lines. Zhai and Kristensson invented and tested a shorthand writing system for pen-based computers called SHARK (shorthand aided rapid keyboarding) [60] and extended it with the SHARK^2 system [32, 61]. In these languages, words are represented by the spatio-temporal patterns that a pen would make if the user selected the letters of the words with an on-screen keyboard.

Forsberg et al. [18] invented and evaluate an alphabet of single-stroke gestures to enter music notation into a computer. Long et al. [36] presented several gesture alphabets, each containing up to 13 single-stroke gestures, to the subjects in several user studies. The goal was to aid interface developers in finding gestures that are likely to be perceived as similar by users or that may be difficult for users to learn and remember. Frankish et al. [19] conducted user studies with pen-based interfaces that were based on handwriting recognition. There are similarities between our problem and the problem of online handwriting recognition, which has been summarized in surveys by Tappert et al. [52], Wakahara et al. [55], and Plamondon and Srihari [44]. For large-alphabet languages like Chinese, Korean, or Japanese, pen and tablet-based interfaces have been developed that recognize each character as it is being written. Both spatial and temporal data are analyzed, for example, the positions of the pen, the number of strokes, and the direction and speed of the writing. The recognition algorithms can be categorized into three groups: (1) preprocessing methods that segment the writing into units such as characters or words, (2) noise reduction techniques that smooth the writing, de-skew slanted characters, and normalize stroke size or length, and (3) feature analysis algorithms that recognize characters or words by matching techniques [52, 55] that compare the input pattern with stored reference patterns or evaluate it with a classifier, such as a neural net [35], hidden Markov model [48, 49], or stochastic network [5, 29].

The core of the SymbolDesign system, first introduced in our earlier work [22], is a trainable classifier. In the current implementation, the classifier is a feed-forward artificial neural network, which is used to recognize user-defined spatio-temporal patterns produced by the pointer or pen. The classifier has an adaptive architecture that enables it to automatically and efficiently adjust itself in order to accommodate the necessary number of pattern classes. A static classifier, for example, similar to the one-hidden-layer neural network proposed by Leung and Cheng [35] to recognize nine basic strokes of Chinese characters, was not an option for three reasons. First, the potential of such a classifier, i.e., the number of classes it can distinguish reliably, has a fixed upper bound that limits the size of the alphabet. Second, among classifiers with enough potential to distinguish between members of the user-defined alphabet, the smallest classifier should be chosen for performance reasons. Third, since the user is allowed to change the alphabet, i.e., add and remove symbols, at any time, the choice of classifier was limited to models that can dynamically adjust their architectures without loosing the data that were already computed. Inspired by Sjogaard's modified cascade correlation algorithm [50], we developed a new method that creates one-hidden-layer neural networks dynamically and efficiently, i.e., the network accommodates the three requirements listed above.

Our paper is organized as follows. Section 2 gives an overview of the SymbolDesign method. Sections 3 and 4 describe pre- and postprocessing of the classifier input and output, respectively. The method to create a dynamic network as a classifier is described in Section 5. The experiments and results are presented in Section 6. The paper concludes with an in-depth discussion in Section 7.

2 System Overview

An interface designed by the SymbolDesign method consists of three major components: a preprocessor, a classifier, and an interpreter (Figure 1). The interface processes sequences of gestures or symbols online. It takes as input spatio-temporal patterns obtained from the pointer or pen device, preprocesses them, passes them to the classifier, and then produces as output commands that can be interpreted by the computer's operating system or application program.

The spatio-temporal pattern is defined by the path that the user produces when moving the pointer or pen or gesturing in the air. Both the spatial as well as the temporal components of this "digital ink" are important for recognizing the meaning of the pattern. The temporal information, analyzed as a sequence of event inter-arrival times, plays a role in the pre- and postprocessing of the classifier data. The spatial information, consisting of path points, is analyzed by the classifier.

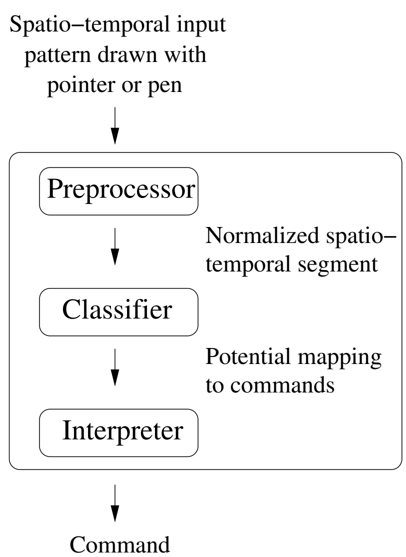

Figure 1: Overview of the three system components, and their inputs and outputs, of an interface created by the SymbolDesign method.

3 Preprocessing of Spatio-Temporal Patterns

This section describes how input data are preprocessed before they are passed into the classifier. This procedure requires both intensive processing and close interaction with the input hardware. An effective scheme is employed to ensure efficient use of computational resources as well as responsiveness of the system (Figure 2).

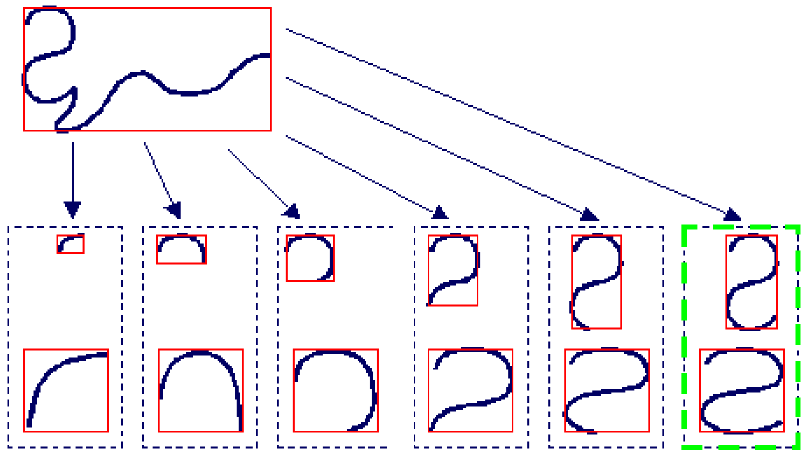

Figure 2: Preprocessing of Spatio-Temporal Patterns: The input pattern (top), which was drawn from left to right, is shown processed in segments (bottom row). Each segment includes the previous segment plus additional positions on the input path. The bounding boxes of the segments are resized to fit a fixed-size rectangle. The pixels of this rectangle constitute the classifier input. Here six resized images were passed into the classifier. None of the patterns produced a high classifier score, except the last (right-most) pattern. This pattern was recognized as a mirrored S symbol after the classification and post-processing steps were performed.

3.1 Input Acquisition and Buffering

The system employs two buffers, a preliminary storage buffer and a main buffer. The main buffer stores the current input segment processed by the interface system. The preliminary buffer is necessary to store pointer positions that do not introduce significant changes to the pointer path that is currently in the main buffer. Once the preliminary buffer has accumulated enough data, its contents are appended to the main buffer and recognition is restarted. The purpose of this preliminary storage buffer is to prevent interruption of the recognition process due to minor (perhaps involuntary) pointer motion. After the main buffer is updated with the new path segment, some of the old pointer positions are removed from it to ensure it does not exceed a length limit. Moreover, positions older than a certain age limit (e.g., 1 second) are also removed. The length limit of the main buffer and the maximum age of the pointer position are determined based on the complexity of the trained spatio-temporal pattern and the speed of the pointer. Their main purpose is to aid the classifier's operation under the real-time constraints.

3.2 Cursor Path Segmentation and Processing

Because recognition is performed continuously and any single-stroke spatio-temporal pattern can be used, the interface system expects that a pattern may start with any path position among the positions stored in the main buffer and must end with the most recent position in the buffer. If exhaustive processing of the accumulated input were performed after each buffer update (which includes path segmentation, multiple scaling operations and multiple classifier evaluations), a heavy load on the computational resources would result. Resulting delays would be unacceptable for a user interface. To save on computation time, the recognition process is therefore suspended until the pointing motion reduces, which results in a smaller rate of buffer reads. Consequently, the time-consuming recognition process usually runs after all input, which may contain a valid spatio-temporal pattern, has been acquired, while the buffer is kept up to date at all times. Furthermore, this allows controlling an acceptable processor load by adjusting the preliminary buffer length and the main buffer update-rate threshold.

When the system decides that the current buffer update rate gives it enough time to process the accumulated input, recognition is initiated. First, the interface system finds all pointer path segments that can potentially be recognized as valid spatio-temporal patterns. Each segment starts with the most recent position in the buffer. The bounding box of the minimal segment has to have an area that is at least as large as the classifier input size. Each following segment contains the previous one plus enough pointer positions to allow resizing of the bounding box (see Figure 2). This resizing process is a normalization step that is necessary to equate the area of the segment's bounding box to the classifier's input size. The fact that the original segments are usually much larger than their resized versions allows the system to grow each segment by several pointer positions at a time.

The classifier evaluates each resized segment, and its output is stored for further analysis. Once the classifier's output is produced, a single processing cycle (segment extraction, resizing, and evaluation) of the system ends. At this point, the recognition process might be forced to halt until the interpretation of a command is completed. In this case, the system either interprets information collected so far or discards it. Otherwise, a new segment is extracted and processed. The next section describes how the processing results are analyzed and interpreted these cases.

4 Classifier Output Analysis and Interpretation

This section describes how the classifier's output is analyzed to facilitate recognition of the drawn symbol and how this symbol is then interpreted as a command.

Input path segments are extracted, resized, and then processed by the classifier in order of their arrival. Both the classifier score, which is an interpretation of the spatial information, and the temporal information stored with the segments are used to determine whether the segment corresponds to a spatio-temporal pattern in the user's alphabet. As temporal information, the inter-arrival time of input events is used to evaluate candidate segments. The system operates under the assumption that the speed of the movement is approximately stable when a valid pattern is being drawn, but slows down when one pattern is finished and a new one is started. If the endpoint of an input path segment with a high classifier score corresponds to a peak in the inter-arrival time of input events (Figure 3), the system can be confident that it did, in fact, recognize a valid spatio-temporal pattern. If two or more overlapping input path segments are recognized by the classifier (Figure 3, top), the one that best correlates with the peak in input event inter-arrival time is interpreted to be the spatio-temporal pattern that the user intended. If no good segment candidate was found by the time recognition is interrupted, the collected data are discarded.

With the proposed method, simple spatio-temporal patterns, which can represent frequently used commands, such as "selection," or "go backward," are recognized almost instantly. However, a user may select an alphabet that contains simple patterns that are also found within more complex symbols. A circle, for example, is an intuitive pattern to use, but it is a part of some digits and letters that the user

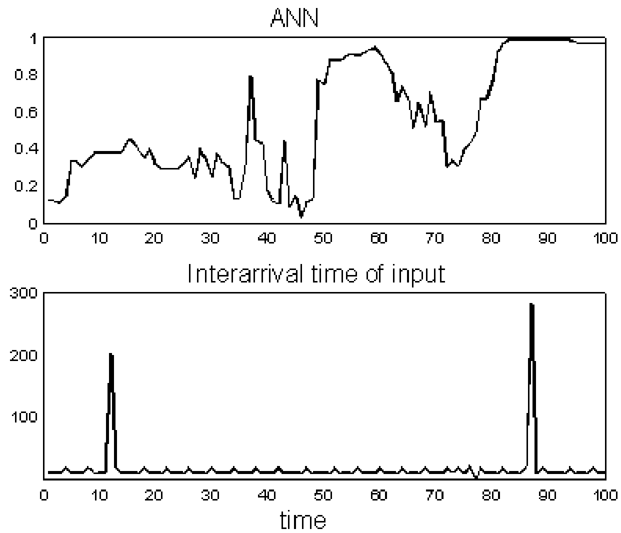

Figure 3: Using the rate of input events to choose a spatio-temporal pattern (STP). Top: Artificial neural network (ANN) output over a period of 100 time units. Bottom: Inter-arrival time of input events over a period of 100 time units. At time $t = 12$, a significant pause in pointer movement was registered, but no candidate STP was recognized by the ANN at that time. At time $t = 60$, the ANN recognized a candidate STP, but no pointer pause was registered. At time $t = 88$, the ANN recognized a candidate STP and a pointer pause was registered. Only in the last case does the system output that this candidate STP is a valid STP.

may also include in his or her symbol alphabet. For this reason, it is necessary to complete processing the spatial and temporal information of all segments even if the classifier produced a high classifier score for some path segment.

The SymbolDesign system uses the delay time between consecutive patterns as a parameter that can be set when an interface is created. A parameter value can be chosen that is appropriate for a particular user or application. Experienced computer users who carefully trained the classifier to recognize their handwriting or gesture input usually do not need feedback on one command before they can start entering the next one. During text processing, for example, they may want to enter letters with only very small delays between each. Inexperienced computer users may want to have more pronounced delays between entering two consecutive spatio-temporal patterns. For both user groups, certain tasks, like web browsing, usually require a delay. Feedback on one action is given before the next action can be performed.

The user can associate each recognized spatio-temporal pattern with a command or event. Any event that the operating system can process can be chosen, for example, keyboard keys, combination of keys pressed, mouse clicks, special buttons like back, forward, refresh etc. When the system recognizes a spatio-temporal pattern, it generates the corresponding event. In case of mouse clicks, the exact monitor location of each selection event is determined using the position of a predefined “hot-spot” relative to the pattern’s bounding box. The default location of the hot-spot is the center of the bounding box, but any point within the box could also be chosen. Note that it is also possible to associate series of events, “macros,” with spatio-temporal patterns, which may be convenient in some applications.

5 The Classifier: A Dynamic Artificial Neural Network

In the current implementation, the SymbolDesign system creates an artificial feed-forward neural network [13] as a classifier for spatio-temporal patterns. Other choices for supervised classifiers are regularization networks or support vector machines [14, 54]. As described in the introduction, there were two primary requirements for the choice of the classifier: real-time performance and an adaptive architecture. Although training a network can take a significant amount of time, the evaluation complexity of even large networks is rather low. In fact, some real-time applications utilize ensembles of neural networks and still meet all the deadlines. On the other hand, fulfilling the requirement of an adaptive architecture was challenging because most neural networks are used with static architectures. The SymbolDesign system provides a solution by creating a dynamic neural network that can efficiently change the symbols, i.e., "classes" it can recognize. This section discusses the network architecture, training set acquisition, and training methods.

5.1 Adaptive Architecture

The neural network receives the resized pointer path image as input. Its input layer must therefore be equal to the number of pixels in the resized image, which is a parameter that the system user can set. We worked with a size of 16×16. A higher resolution can represent more complex patterns but takes longer to evaluate. The resolution should be chosen according to alphabet size and input device precision.

The number of nodes in the output layer equals the number of classes defined by the user, i.e., the network potential. It changes when the user adds or deletes symbols to the alphabet. We chose an adaptive network design that ensures that adding or deleting a node does not result in the loss of weights that were already computed for other nodes.

We chose a network architecture that contains one hidden layer. The number of nodes in the hidden layer is a parameter that can be adjusted and is critical for the network performance and potential. From the performance point of view, the number of hidden nodes should minimized. We propose a combination of a method to automatically choose the size of the hidden layer and a probabilistic technique to assess the likelihood of training convergence. The mean squared error (MSE) is used as a measure of performance and a convergence criterion. It is the average of the squared differences between desired and measured output values over all output nodes.

Our method was inspired by Sjogaard's work [50] that modified the Cascade Correlation algorithm [16]. The original Cascade Correlation algorithm implemented a dynamic network architecture by sequentially increasing (cascading) the number of hidden layers of the network [16]. The number of training examples used by this method is unfeasible for our application. The "modified Cascade-Correlation architecture," [50] has one hidden layer to which nodes are sequentially added. It performed at least as well as the original Cascade-Correlation architecture on a set of benchmark problems [57]. Our SymbolDesign method follows Sjogaard's approach [50] to add one node at a time to the hidden layer until the network can solve the classification task. The principal difference between our and Sjogaard's approach is that the modified Cascade Correlation architecture does not take into account a priori information about the problem at hand, such as input dimensionality. In our approach, the size of the hidden layer is selected according to the complexity of the classification task. Our SymbolDesign method thus avoids testing architectures that are unlikely to deliver the potential required for the given task. We next describe how the SymbolDesign method determines the size of the hidden layer for the initially tested network.

Given the dimensionality of a classification task, i.e., the number N_x of input nodes, the number N_y of output nodes and the number N_p of training samples, the number N_z of hidden nodes in a one-hidden-layer fully-connected feed-forward neural network can be estimated by bounding the number N_w of weights [56]:

$$\frac{N_y N_p}{1+\log_2 N_p} \leq N_w \leq N_y(\frac{N_p}{N_x}+1)(N_x+N_y+1)+N_y. \quad (1)$$

The number N_z of hidden nodes [23] is then given by:

$$N_z = \frac{N_w}{N_x+N_y}. \quad (2)$$

Minimum and maximum values for the number of hidden nodes can be computed by applying the respective bounds in Eq. 1 to N_w in Eq. 2. The minimum value is needed for "easy" classification tasks, the maximum value for "hard" tasks. Since a-priori information about the complexity and similarity of the patterns that the user may choose is not available, it is reasonable to assume that the number of hidden nodes actually needed lies somewhere in the middle between the minimum and maximum values. We therefore initialized the number of hidden nodes to the average of maximum and minimum values (Figure 4).

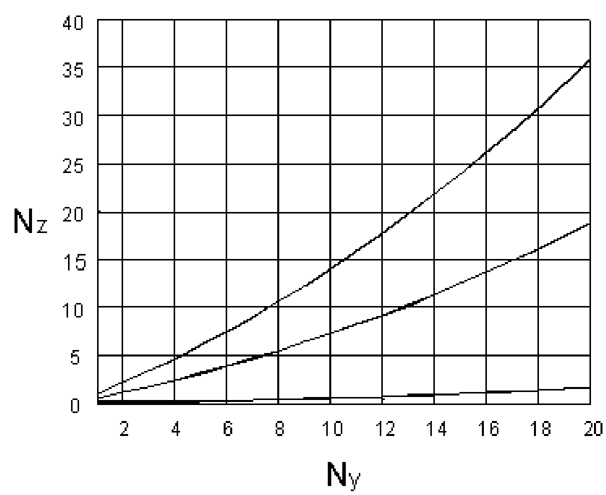

Figure 4: Minimum, average, and maximum number of hidden nodes N_z as a function of $N_y = 1, \ldots, 20$ classes, 10 examples per class ($N_p = 10N_y$), and $N_x = 256$ input nodes.

Slightly overestimating the network potential associated with the size of the hidden layer is not harmful, but it is detrimental if the potential is not sufficient to accommodate all classes. This can happen if either the initial value of number of nodes in the hidden layer is too small or if the user adds more patterns to be recognized to the existing network. In such cases, the network cannot be trained to produce the desired classification, and the system does not know what compromised the training procedure: Training may have just reached a local minimum, or there may not exist a suitable minimum. In the first case, training should be restarted with random weights. In the second case, the training process may not converge, and the number of hidden nodes must therefore be increased. Our SymbolDesign method takes a probabilistic approach to resolve these issues. It estimates the number N of training attempts that it must make before it establishes the inability of the training procedure to converge for a given network state by computing [26]:

$$N = \frac{\ln(1 - F_W(a))}{\ln(1 - F_X(a))}, \tag{3}$$

where X is the sum of squared errors on any individual attempt, W is the lowest value for X, $F_X(a)$ is the fraction of attempts that would result in a value of X less than or equal to a, a confidence threshold, and $F_W(a)$ is the fraction of X values that result in a value of W less than or equal to a. Once N attempts to converge have been made, the number of hidden nodes should be increased.

5.2 Training

Training of the dynamic neural network proceeds in two stages (Figure 5). The purpose of dividing training into two stages is to ensure that the network will produce correct classifications while trained on as few examples as possible, hence minimizing training time.

During the first stage, called "basic training," the network is presented with only a few examples of each pattern (e.g., five). In rare cases, these examples are sufficient to train the network to stably recognize a pattern. However, most spatio-temporal patterns require larger training sets to ensure correct classification. Basic training provides the network with a rough estimate of the classification task, so the error rate might be high due to partially learned decision boundaries of certain classes. This means that the neural network was not presented with enough examples of one or more classes to be able to distinguish between all of them accurately. The system identifies problematic classes by assigning a confidence value to each symbol in the alphabet. Initially, all classes have the same value.

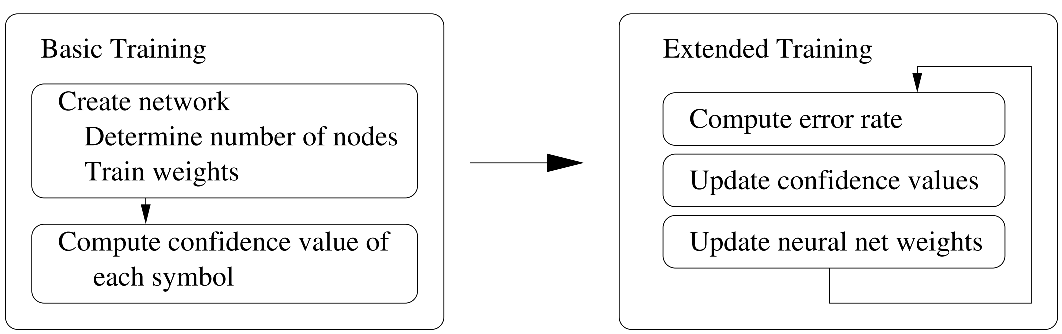

Figure 5: Overview of the two training stages. In the basic training stage, a few instances of each symbol are used to create and train the neural network. For each symbol, a value is computed which represents the system's confidence that the network can reliably recognize the symbol. In the extended training stage, additional symbol instances are passed into the neural network, its weights are adjusted, and the confidence values updated. This process is repeated until network performance converges.

In the second stage, called "extended training," the user can start experimenting with the system. During this experimentation, the recognition error rate is usually still high. The user is asked to notify the system about its mistakes so that the confidence values can be updated to reflect the recognition error rates of the symbols. The confidence values are increased for every correct classification and decreased for every incorrect one.

To speed up the training, the user is asked to provide additional examples for the symbols. The chance of a specific symbol to be selected is proportional to its associated confidence value. As a result, symbols that were recognized poorly after basic training are emphasized during extended training. Each new sample symbol is added to the training set if the current network does not recognize it correctly, or is discarded otherwise. In addition, for each new sample pattern, the system continues to adjust confidence values. The network is updated after each time an additional pattern is entered by the user. This procedure, if continued long enough, typically results in a sufficient number of examples for each class.

A standard back-propagation algorithm with gradient descent is employed for training the neural network [13]. To increase the speed of training and reduce the likelihood of converging into local minima, a number of convergence acceleration techniques [9] are used: The first few examples of a symbol are passed into the network sequentially in batch mode, which is quite robust for small training sets and computes an average performance gradient. A conjugate gradient descent method is used to force the search for the best set of weights into a direction orthogonal to that of the previous step, instead of the direction of the steepest gradient. The system also uses a "momentum variable" that controls the extent to which the previous change in network weights affects the current change in network weights.

After achieving convergence in batch mode, the training switches to a stochastic mode, which is more robust for large training sets. The stochastic approach updates the network for every randomly chosen example thus producing a "noisy" performance gradient which often leads to better solutions than solutions computed by the standard gradient descent method. In the stochastic training mode, it is simple to track the correspondence between the inputs and changes in the network.

Both training modes employ a variable learning rate [9] that is recomputed according to the observed changes in the network's mean squared error. That is, the learning rate is increased if there is no improvement in performance and is annealed if the error drops.

6 Experiments and Results

This section describes the methodology and quantitative results of experiments with twenty human subjects who did not have physical conditions that limited their computer use. It also describes our initial experiences working with two subjects with severe disabilities and gives some qualitative results.

6.1 Testing Methodology of Quantitative Experiments

The twenty human subjects who participated in our quantitative experiments belonged to two groups: The ten subjects in group A were sophomore college students with strong computer skills. The ten subjects in group B were individuals with some computer literacy. None of the subjects was previously exposed to the system.

Two experiments were performed with Pentium IV 1.4 GHz machines running the Windows XP operating system. In the first experiment, subjects were asked to use the SymbolDesign method with a traditional computer mouse to perform:

Task 1:

1. Design an alphabet of five commands.
2. Train the interface to recognize these commands.
3. Test the interface by using the commands with a web browser.

During the first step, the users were instructed to use the symbol O as a selection ("click") command. They were then asked to invent four symbols to correspond to the browser functions "back," "forward," "stop," and "favorites."

Upon completion of the first step, subjects were asked to perform the basic training of the classifier. They provided the system with five samples of each of the five symbols, drawn with the mouse. Given these training inputs, the system assigned a confidence value to each symbol. Subjects were then asked to conduct extended training of the classifier. During this phase, the subjects were asked to open an internet browser and, using only the five newly created symbols, browse the internet in a natural way with the mouse as the input device. We recorded the time it took each subject to configure the system – the "average adjustment time." The final phase of the experiment was the test phase, designed to evaluate the "recognition accuracy" of the interface. The user was asked to input 50 commands while browsing the internet. In this test phase, we recorded the number of correctly performed actions out of the 50 attempted actions. We also measured the average processing time to recognize each symbol, the "response time."

In the second experiment, the subjects were asked to expand their existing alphabets to include ten new symbols – the digits 0 through 9. The mapping of these symbols to commands was straightforward – the symbols were mapped to the text input of the digits themselves. The training procedure was the same as in the first task. For basic training of the classifier, each digit was drawn five times with the computer mouse. During the extended training phase, the users entered the digits into a word processor. We recorded the time it took each subject to configure the system – the average adjustment time. For the test phase, the subjects were asked to launch the word processor and produce all ten digits in consecutive order, using the mouse as a pointing device. In this phase, we also recorded the recognition accuracy of the interface, i.e., the number of correctly performed actions out of the 10 attempted actions, and the response time, i.e., the average time it took the interface to recognize a digit. In summary, the subjects used the SymbolDesign method to create an interface to perform:

Task 2:

1. Add the 10 digit symbols to the previously designed alphabet.
2. Train the interface to recognize these symbols.
3. Test the interface by inputting the digits into a text processing program.

Note that user access to the keyboard and full mouse functionality was allowed only during the configuration and training stages in both tasks, but not during the testing phase.

6.2 Results of Quantitative Experiments

The subjects in both groups were able to successfully complete both tasks assigned. Subjects with strong computer skills (Group A) needed 3 minutes of adjustment time to decrease the error rate of the classifier to a point where the system could be used reliably with a web browser. For Task 1, the average recognition rate was 95%. Other recognition and timing results are listed in Table 1. Individuals with less computer experience (Group B) took longer to adjust to the interface than users with strong computer experience (Group A). Subjects in Group A created interfaces that exhibited smaller error rates for both tasks than users in Group B. Samples of symbols created by users are shown in Figure 6. In the extended training phase, typically 20–30 instances of each symbol were needed. The average response time of the interfaces in recognizing a symbol was under 600 milliseconds, which resulted in an average recognition rate of 1.7 symbols per second.

Table 1: Average recognition accuracy of the interface and average adjustment time for two groups of 10 subjects, each subject performing Task 1 fifty times and Task 2 ten times.

Subject Group	Av. Recognition Accuracy Task 1	Av. Recognition Accuracy Task 2	Av. Adjustment Time Task 1	Av. Adjustment Time Task 2
A (strong computer skills)	95 %	87 %	3 min	12 min
B (some computer skills)	85 %	75 %	6 min	18 min

Our subjects enjoyed using the interfaces they had created and some of them continued playing with them after both tasks were finished. These users were able to use the interface, for example, to work with a calculator user interface.

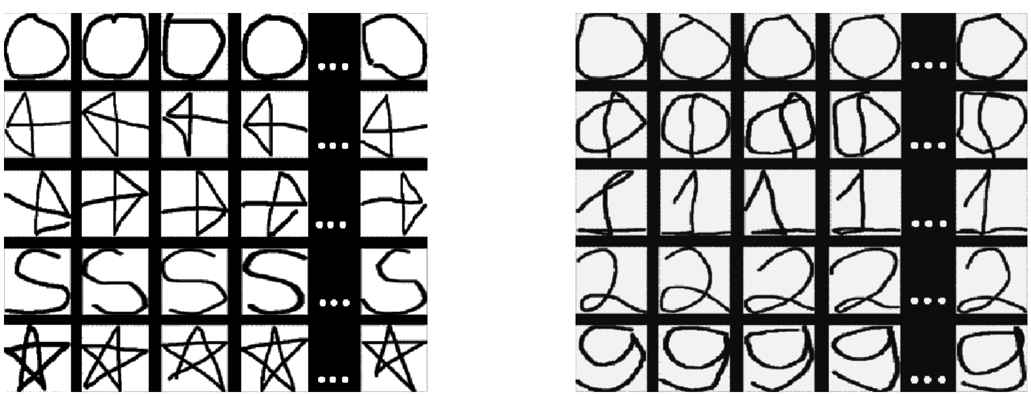

Figure 6: Left: Instances of five symbols as produced by a user in Group B who was asked to perform Task 1. The user was asked to assign the symbol O to the "select" command ("left mouse click" command). The remaining four symbols were chosen by the user. Symbol $\rightarrow$ was chosen for the "forward" command, symbol $\leftarrow$ for the "backward" command, symbol S for the "stop" command, and symbol $*$ for selecting "favorites." Right: Instances of five out of eleven symbols as produced by a user in Group B who was asked to perform Task 2. The subject's pattern instances were used to train the system to recognize the ten digits and the selection symbol 0.

6.3 Initial Experience with Users with Quadriplegia

The SymbolDesign system was tested with two subjects with severe disabilities, shown in Figure 7. Subject 1, a 28-year old man, and subject 2, 40-year-old woman, were both born with severe cerebral palsy that resulted in quadriplegia. The subjects were non-verbal, but could communicate with their caregivers using small head movements. Their controlled head movements were often disturbed by significant involuntary movements. The involuntary movements were tremor, i.e., unwanted oscillatory movements [46], and reflexive movements, which were very fast and difficult to predict. Subject 1 had difficulty holding his head straight up and had better control of horizontal than vertical movements. Subject 2 had better control of vertical than horizontal movements.

Figure 7: Subject 1 (top left image) and subject 2 (top right image) are using the Camera Mouse as a mouse-pointer substitution interface. Subject 2's attempt to draw a vertical line is shown in blue on the left monitor in the bottom image. The subject was instructed to draw a straight line between the two red regions. She failed to reach both regions. During the drawing, the subject's face and the tracking results of the Camera Mouse were observed on the monitor to the right. It was verified that the Camera Mouse was functioning correctly and that the drawn pattern corresponded to the user's head motion.

The subjects' primary method of communication were small head nods to select letters spelled out by an assistant or to directly answer the assistant's questions. Both subjects had experience using the assistive technologies to access the computer such as manual switches, Eagle Eyes [11], and the Camera Mouse [3]. They had used these interface technologies to access text entry software based on on-screen keyboards and to play reaction games with graphical displays [3, 6, 15].

The Camera Mouse was the interface preferred by both subjects and was therefore used in the experiments with our symbol recognition system. We asked the subjects to help us design symbols that they could draw comfortably. We started with the symbol O, which is used in our other experiments as a selection command. Our quadriplegic subjects were not able to draw the symbol O successfully – the drawn symbols were significantly distorted because the users were not able to move their faces in circular patterns. Sometimes involuntary reflexive movements distorted the patterns. Tremor [46] was not a significant factor in the pattern distortion, because the Camera Mouse was used in a tremor-suppression mode that filtered oscillatory movements.

We then tried simpler patterns, containing straight-lines, such as triangles, and finally asked the subjects to draw symbols that just contained one line – either a horizontal, vertical, and diagonal line. The lines that the subjects were able to draw with the Camera Mouse were not straight, as can be seen in Figure 7, which shows subject 2's attempt to draw a horizontal line. It was a considerable effort for the subjects to draw these lines. For example, in nine attempts, it took subject 2, on average 19.0 s to draw a line, with a standard deviation of 15.6 s. The subjects were generally able to move the mouse pointer towards the intended direction, but then did not have enough control to maintain the direction of movement. We asked them to start from some region on the screen and try to reach a goal region. The subjects typically were able to reach or come close to the goal region by using some indirect route. This route was different in each drawing attempt. Since the subjects could not reproduce a spatio-temporal pattern, they were not able to work with our symbol recognition system.

7 Discussion

The main contribution of our work is the SymbolDesign method for creating user-centered, pen- or pointer-based interfaces. The method combines two original techniques: (1) a new method to dynamically create classifiers that evaluate the spatial information of input patterns, and (2) a new method to exploit temporal information by pre- and post-processing candidate segments of the input patterns. Since these spatio-temporal patterns are completely determined by the user, the SymbolDesign method is general and can be conveniently applied to create interfaces for various tasks. The fact that the system does not have a priori information about the alphabet that a person will create makes the symbol classification problem challenging. Our solution, the SymbolDesign method, dynamically creates classifiers that are able to efficiently distinguish between the a-priori-unknown number of symbols of unknown appearance and structure.

Users without physical limitations quickly became comfortable with designing interfaces and using them. The interfaces performed well during testing – symbols were recognized reliably and efficiently. We tested interfaces with alphabets of up to 15 symbols (10 digits and 5 browser commands). The number of symbols were appropriate for the purpose of replacing the functionality of the mouse and some functionality of the keyboard. Given that the average response time of the interfaces in recognizing a symbol was under 600 milliseconds in our experiments, we believe that somewhat larger alphabets could be used without raising the response time per symbol significantly.

The SymbolDesign system may also be able to handle a much larger number of commands, for example, to fully replace the functionality of the keyboard. Such expansion, however, might raise the computational demands of the created interface to a level where symbol recognition becomes so slow that people would not want to use the interface. This would then require a redesign of the system's major components to achieve the necessary speed-up. Such a redesign would be simplified by our modular approach in developing the SymbolDesign method in the first place: each of the main components – input processor, classifier, and interpreter – could be changed or completely altered.

To the best of our knowledge, there is no prior work that describes a method to allow users to create their own interfaces, i.e., designing symbols or gestures and mapping them to commands. There are also no prior user studies where subjects were asked to invent symbols for computer input. The user studies we conducted were aimed at evaluating whether the SymbolDesign method indeed followed the paradigm of user-centered, accessible design. In our analysis, we focused on evaluating the errors of the interfaces in recognizing symbols, the amount of training time needed, and the average recognition time during use. Our subjects without physical limitations used the input device that they were most experienced with – the traditional computer mouse – to design, train, and use the interfaces. This avoided the problem that measured recognition errors could be due to the user's inexperience with the input device or, for the example of the Camera Mouse, due to inaccuracies of the image analysis method. For these tests, the reported errors are therefore probably mostly due to failures of the interface. We speculate that the error rates were larger for less experienced computer users than for experienced computer users because they may have been hesitant and paused while they were drawing a symbol or they may have drawn samples of symbols that varied too much in shape.

An important concept of accessible design is enabling users to customize their settings to accommodate their preferences or abilities. In our work, the customization is inherent, since users choose the symbols and commands of the interface themselves. Users are likely to select patterns they can remember and easily reproduce, thus producing interfaces that are simple and intuitive to control, which is another important component of the user-centered paradigm. Furthermore, users are allowed to create redundant symbols, that is, in the interface they design, two or more symbols could map to the same command. The SymbolDesign method also includes redundancy and is flexible in the sense that it allows users to work with both the symbol interface and the traditional input tool. For example, a caregiver may use the computer mouse to help a user with severe disabilities launch a web browser, and the user then would work with the Camera Mouse and an interface he or she created with the SymbolDesign method to surf the web.

Unfortunately, the two subjects with severe cerebral palsy who participated in our experiments were physically not able to train an interface and use it to surf the web. Nonetheless, we believe that our concept to develop interfaces that learn to recognize gestures that the users choose themselves is particularly important for people with severe disabilities. Our flexible interface design may enable users with less severe motion impairments to select gestures that they have the abilities to perform.

Our initial experiments showed that our subjects with severe cerebral palsy were physically too limited to be able to gesture spatio-temporal patterns such as lines or circles, but were able to move the pointer to certain regions on the screen and keeping it in these regions for a while. We thus propose the idea to discretize the input patterns further in order to extract "pivot points," i.e., screen regions that the pointer travels to and dwells in. The neural network would then have to classify patterns constituted by pivot points. Another idea is to use "gesture spotting methods" [28, 34] to locate the start and endpoints of a gesture. If at least one motion pattern can be reliably recognized, the interface can serve as a "binary switch" and control scan-based applications, for example, software created by so-called "wifsids," widgets for single-switch input devices, developed by Steriadis and Costantinou [51], or the applications described by Grauman et al. [21].

Our future work will be divided into two areas of investigation. We will (1) further improve the SymbolDesign technology, and (2) conduct additional user studies to evaluate the interfaces created by the SymbolDesign technology. To improve the SymbolDesign technology, we will work on reducing the errors in classification rates. This may be possible by taking advantage of additional input characteristics of the "digital ink," for example, the direction and speed of the drawing movement. Another idea is to change the preprocessing phase to extract symbol features that are then passed into the classifier, instead of passing in a full image of the symbol. This would likely reduce the input dimensionality, and thus, the size of the neural network. Feature extraction would be beneficial if input characteristics that emphasize differences between classes can be identified and the computational costs associated with their extraction are not significant. We will also test whether a different type of classifier should be used to create interfaces that are more reliable and efficient than the neural-network-based interfaces.

A design feature that we will add to the system, is an optional window that will provide users with visual feedback. It will show the spatio-temporal pattern on the screen, while it is being drawn. Initially we decided against this use of screen space, since screen space is so valuable for miniature computers. In future work, we will revisit this idea to see whether recognition results could be improved when users see the patterns they are drawing. This might be particularly useful for people with little computer experience.

We will also investigate the idea to use a "mask" during the training process. This mask would show the "average pattern" that the user draws for a particular symbol. Such a mask might teach the user to follow this particular pattern, instead of drawing different instances of the same symbol with too much variability. Using the mask in the training phase may help the user to minimize his or her input errors during regular use, which is an important consideration for our accessible design paradigm. The use of a mask has been shown to be helpful for other gesture recognition systems, for example, for the Finger Counter interface [7], where the mask was a fixed hand gesture for the user to imitate. Note that in our application, the masks must be dynamically determined by an averaging procedure during training, since the symbols are created by the users and thus are not known in advance.

Some commercial online handwriting systems require the user to enter each character separately into a fixed-size block [8, 40]. This requirement simplifies handwriting recognition because it yields character segmentation. Handwriting interface systems without this requirement have to address the problem of "run-on writing," where neighboring characters touch or overlap one another. Solutions perform character segmentation and recognition simultaneously, for example, by segmenting and matching all spatio-temporal patterns that can be characters, and then ranking the patterns by their recognition scores [42]. This "candidate lattice" approach is valuable if multi-stroke patterns can consist of subpatterns that can be individual characters by themselves, as occurs in Japanese. In our application, a similar issue with subpatterns can arise: A single-stroke pattern may contain a prefix that by itself could be a pattern. If the user enters such a pattern and happens to slightly pause directly after entering the prefix, our size-invariant, pause-dependent recognition method would identify the prefix instead of the full pattern and issue an unwanted command. In our experiments, the users only chose prefix-free single-stroke patterns. To allow the user to work with patterns that contain prefix subpatterns, we could extend our system with the candidate lattice approach [42].

Our original goal was to minimize the restrictions on alphabet elements and to let the users decide which patterns they can remember and easily reproduce. However, it is not a serious restriction to require the user to design only prefix-free single-stroke patterns, as there exists an enormously rich set of such patterns. The alphabet invented by Isokoski and Raisamo [25], for example, contains 73 prefix-free single-stroke patterns.

We plan to conduct a user study that will further investigate the choices that subjects make in selecting symbols. For example, the symbols designed by the twenty users without disabilities were often iconic, i.e., the symbols suited or suggested their meanings, as can be seen for the symbols $\rightarrow$ and $\leftarrow$ mapped to the "forward" and "backward" commands in Figure 6.

The users in our experiments liked the option that they could design their own patterns. The users in the experiments conducted by Long et al. [36] also wanted the ability to define their own gestures. We will further investigate whether user acceptance of pen- or pointer-based interfaces indeed increases when users are allowed to choose their own symbols. We will also study whether some users might be overwhelmed by the task of having to design their own distinct symbols and would prefer to work with a pre-defined alphabet. The issue of gesture similarity arises for anybody who designs a gesture alphabet [36]. An advantage of our SymbolDesign method is that the users find out during the extended training phase if they inadvertently proposed symbols that are too similar for the classifier to distinguish.

Frankish et al. [19] analyzed the relationship between recognition accuracy and user acceptance of pen-based interfaces that used online handwriting recognition. They reported that the impact of recognition performance on user satisfaction depends on the nature of the task being performed. Users seemed to perform a cost/benefit analysis. For example, for the task of entering a name and phone number into a virtual fax form, users accepted the costs associated with recognition errors, but for the task of entering a text into a virtual appointment book, they did not. We will investigate whether users of interfaces created with our SymbolDesign system also perform a cost/benefit analysis. They may be most inclined to work with these interfaces when traditional input devices cannot be used, for example, when they need to control a miniature or remote computer. A cost/benefit analysis may also be very important for people with severe disabilities. For quadriplegic, non-verbal users, whose physical conditions prevent them from using traditional input devices, interfaces that they can design to match their physical abilities, would be extremely beneficial. The most urgent task of our future work will therefore be to develop the pivot-point or gesture spotting methods outlined above.

Acknowledgments

We wish to thank John J. Magee, Rick Hoydt, Robyn Pancholi, James Gips, and the students of the Boston University Video Game Creators Consortium for assistance. The work was supported by the National Science Foundation with grants IIS-0093367, IIS-0308213, IIS-0329009, and EIA-0202067.

References

[1] Applied Science Laboratories, Bedford, MA. http://www.a-s-l.com.

[2] J.-D. Bauby. *The Diving Bell and the Butterfly.* Vintage Books, 1997.

[3] M. Betke, J. Gips, and P. Fleming. The Camera Mouse: Visual tracking of body features to provide computer access for people with severe disabilities. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 10(1):1–10, March 2002.

[4] Inc CameraMouse. Hands-free computer control. http://www.cameramouse.com, 2005. Abilene, TX, USA.

[5] S. J. Cho and J. H. Kim. Bayesian network modeling of strokes and their relationships for on-line handwriting recognition. In *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, pages 86–90, September 2001.

[6] R. L. Cloud, M. Betke, and J. Gips. Experiments with a camera-based human-computer interface system. In *7th ERCIM Workshop on User Interfaces for All*, pages 103–110, Paris, France, October 2002.

[7] S. C. Crampton and M. Betke. Counting fingers in real time: A webcam-based human-computer interface with game applications. In *Proceedings of Universal Access in Human-Computer Interaction Conference (UA-HCI)*, pages 1357–1361, Crete, Greece, June 2003.

[8] H. D. Crane, J. S. Ostrem, and P. K. Edberg. Method for distinguishing between complex character sets. U. S. Patent 4 531 231, July 1985.

[9] Y. L. Cun, L. Bottou, G. Orr, and K. Muller. Efficient backprop, neural networks: Tricks of the trade. In *Lecture Notes in Computer Sciences 1524*, pages 5–50. Springer-Verlag, 1998.

[10] Diamond Bullet Design. Usability first. http://www.usabilityfirst.com, 2005. Ann Arbor, MI, USA.

[11] P. DiMattia, F. X. Curran, and J. Gips. *An Eye Control Teaching Device for Students without Language Expressive Capacity – EagleEyes.* The Edwin Mellen Press, 2001. See also http://www.bc.edu/eagleeyes.

[12] Don Johnston, Inc., Penny & Giles HeadWay, infrared head-mounted mouse alternative. http://www.donjohnston.com.

[13] R. O. Duda, R. E. Hart, and D. G. Stork. *Pattern Classification, Second Edition.* John Wiley & Sons, New York, 2001.

[14] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50, 2000.

[15] C. Fagiani, M. Betke, and J. Gips. Evaluation of tracking methods for human-computer interaction. In *IEEE Workshop on Applications in Computer Vision*, pages 121–126, Orlando, Florida, December 2002.

[16] S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 524–532, Denver, CO, USA, 1990. Morgan Kaufmann, San Mateo.

[17] European Research Consortium for Informatics and Mathematics (ERCIM). User interfaces for all. http://ui4all.ics.forth.gr/workshop2004/technical-description.html, 2005.

[18] A. Forsberg, M. Dieterich, and R. Zeleznik. The music notepad. In *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology (UIST '98)*, pages 203–210. ACM Press, 1998.

[19] C. Frankish, R. Hull, and P. Morgan. Recognition accuracy and user acceptance of pen interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '95)*, pages 503–510. ACM Press/Addison-Wesley Publishing Co., 1995.

[20] D. Goldberg and C. Richardson. Touch-typing with a stylus. In *Proceedings of the SIGCHI Conference on Human factors in computing systems (CHI '93)*, pages 80–87. ACM Press, 1993.

[21] K. Grauman, M. Betke, J. Lombardi, J. Gips, and G. R. Bradski. Communication via eye blinks and eyebrow raises: Video-based human-computer interfaces. *Universal Access in the Information Society*, 2(4):359–373, November 2003.

[22] O. Gusyatin, M. Urinson, and M. Betke. A method to extend functionality of pointer input devices. In C. Stary and C. Stephanidis, editors, *Proceedings of the 8th ERCIM Workshop on User Interfaces for All, Lecture Notes in Computer Science 3196*. Springer-Verlag, Vienna, Austria, June 2004.

[23] R. Hecht-Nielsen. Kolmogorov's mapping neural network existence theorem. In *Proceedings of IEEE First Annual International Conference on Neural Networks, Vol. 3*, pages 11–13, San Diego, CA, 1987.

[24] T. Hutchinson, K. P. White JR., W. N. Martin, K. C. Reichert, and L. A. Frey. Human-computer interaction using eye-gaze input. *IEEE Transactions on Systems, Man and Cybernetics*, 19(6):1527–1533, 1989.

[25] P. Isokoski and R. Raisamo. Device independent text input: a rationale and an example. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI '00)*, pages 76–83, Palermo, Italy, 2000. ACM Press.

[26] M. S. Iyer and R. R. Rhinehart. Method to determine the required number of neural network training repetitions. *IEEE Transactions on Neural Networks*, 10(2):427–432, March 1999.

[27] Q. Ji and Z. Zhu. Eye and gaze tracking for interactive graphic display. *Machine Vision and Applications*, 15(3):139–148, July 2004.

[28] H. Kang, C. W. Lee, and K. Jung. Recognition-based gesture spotting in video games. *Pattern Recognition Letters*, 25(15):1701–1714, November 2004.

[29] K.-W. Kang and J. H. Kim. Utilization of hierarchical, stochastic relationship modeling for Hangul character recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1185–1196, September 2004.

[30] A. Kapoor and R. W. Picard. Real-time, fully automatic upper facial feature tracking. In *Proceedings of the Fifth IEEE International Conference on Automatic Face Gesture Recognition*, pages 10–15, Washington, D.C., May 2002.

[31] K.-N. Kim and R. S. Ramakrishna. Vision-based eye-gaze tracking for human computer interface. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Vol. 2*, pages 324–329, Tokyo, Japan, October 1999.

[32] P.-O. Kristensson and S. Zhai. SHARK^2: A large vocabulary shorthand writing system for pen-based computers. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology (UIST '04)*, pages 43–52. ACM Press, 2004.

[33] LC Technologies, Eyegaze System. http://www.lctinc.com.

[34] H.K. Lee and J.H. Kim. An HMM-based threshold model approach for gesture recognition. *Transactions on Pattern Analysis and Machine Intelligence*, 21(10):961–973, October 1999.

[35] W. Leung and K. Cheng. A stroke-order free Chinese handwriting input system based on relative stroke positions and back-propagation networks. In *Proceedings of the 1996 ACM Symposium on Applied Computing (SAC'96)*, pages 22–27, Philadelphia, PE, US, 1996. ACM Press.

[36] A. C. Long Jr., J. A. Landay, L. A. Rowe, and J. Michiels. Visual similarity of pen gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing systems (CHI '00)*, pages 360–367, The Hague, The Netherlands, 2000. ACM Press.

[37] I. S. MacKenzie and S. Zhang. The immediate usability of graffiti. In *Proceedings of Graphics Interface '97*, pages 129–137, Toronto, Canada, 1997. Canadian Information Processing Society.

[38] Madentec. http://www.madentec.com.

[39] J. J. Magee, M. R. Scott, B. N. Waber, and M. Betke. EyeKeys: A real-time vision interface based on gaze detection from a low-grade video camera. In *IEEE Workshop on Real-Time Vision for Human-Computer Interaction (RTV4HCI)*, Washington, D.C., July 2004. IEEE Computer Society. 8 pp.

[40] A. Meyer. Pen computing: a technology overview and a vision. *ACM SIGCHI Bulletin*, 27(3):46–90, 1995.

[41] C. H. Morimoto, D. Koons, A. Amir, and M. Flickner. Pupil detection and tracking using multiple light sources. *Image and Vision Computing*, 18(4):331–335, March 2000.

[42] H. Murase. Online recognition of free-format Japanese handwritings. In *Proceedings of the 9th International Conference on Pattern Recognition, Vol. 2*, pages 1143–1147, Rome, Italy, November 1988. IEEE Computer Society Press.

[43] K. Oka, Y. Sato, and H. Koike. Real-time fingertip tracking and gesture recognition. *IEEE Computer Graphics and Applications*, 22(6):64–71, November/December 2002.

[44] R. Plamondon and S. N. Srihari. Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, January 2000.

[45] Microsoft Press Release. With launch of tablet PCs, pen-based computing is a reality. http://www.microsoft.com/presspass/features/2002/nov02/11-07TabletLaunch.asp, November 2002.

[46] C. Riviere and N. Thakor. Assistive computer interface for pen input by persons with tremor. *IEEE Eng. Med. Biol.*, 15(3):29–36, 1996.

[47] A. Sears and R. Arora. Data entry for mobile devices: an empirical comparison of novice performance with jot and graffiti. *Interacting with Computers*, 14:413–433, October 2002.

[48] B. K. Sin and J. H. Kim. A statistical approach with HMMs for on-line cursive Hangul (Korean script) recognition. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 147–150, October 1993.

[49] B.-K. Sin and J. H. Kim. Ligature modeling for online cursive script recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:623–633, June 1997.

[50] S. Sjogaard. A conceptual approach to generalization in dynamic neural networks. *Neural Computation*, 2:198–209, 1991.

[51] C. E. Steriadis and P. Constantinou. Designing human-computer interfaces for quadriplegic people. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 10(2):87–118, 2003.

[52] C. C. Tappert, C. Y. Suen, and T. Wakahara. The state of the art in on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(8):787–808, August 1990.

[53] Tash solutions. http://www.tashinc.com.

[54] V. N. Vapnik. *Statistical learning theory.* John Wiley & Sons, New York, 1998.

[55] T. Wakahara, H. Murase, and K. Odaka. On-line handwriting recognition. *Proceedings of the IEEE*, 80(7):1181–1194, July 1992.

[56] B. Widrow and M. A. Lehr. 30 years of adaptive neural networks: Perceptron, madaline, and backpropagation. *Proceedings of the IEEE*, 78(9):1415–1442, 1990.

[57] D.-Y. Yeung. Node splitting: A constructive algorithm for feed-forward neural networks. In J.E. Moody, S. J. Hanson, and R. P. Lippman, editors, *Advances in Neural Information Processing Systems, Vol. 4*, pages 1072–1079. Morgan Kaufman Publishers, San Mateo, CA, 1991.

[58] H. S. Yoon, J. Soh, Y. J. Bae, and H. S. Yang. Hand gesture recognition using combined features of location, angle and velocity. *Pattern Recognition*, 34(7):1491–1501, July 2001.

[59] L. Young and D. Sheena. Survey of eye movement recording methods. *Behavior Research Methods and Instrumentation*, 7(5):397–429, 1975.

[60] S. Zhai and P.-O. Kristensson. Shorthand writing on stylus keyboard. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI '03)*, pages 97–104. ACM Press, 2003.

[61] S. Zhai, P.-O. Kristensson, and B. A. Smith. In search of effective text input interfaces for off the desktop computing. *Interacting with Computers*, 2005. In press, corrected proof available online since March 2004.

[62] ZyonSystems. i-Pen – Presentation Digital Pen / Optical Pen Mouse. http://www.zyonshop.com/product/ipen.htm, 2005. Seattle, WA, USA.