Univ Access Inf Soc (2006) 4: 223–236
DOI 10.1007/s10209-005-0013-9

LONG PAPER

Margrit Betke · Oleg Gusyatin · Mikhail Urinson

Symbol design: a user-centered method to design pen-based interfaces and extend the functionality of pointer input devices

Published online: 29 September 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.

Keywords Universal access · Assistive technology · Universal interfaces · User interfaces · Camera interfaces · Pen-based interfaces · Video-based human-computer interfaces · Dynamic neural networks

M. Betke (✉) · O. Gusyatin · M. Urinson
Department of Computer Science, Boston University,
Boston, MA 02215, USA
E-mail: betke@cs.bu.edu
URL: 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 pen- or pointer-based interfaces require a 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 "SymbolDesign," a method 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 device used to control the computer, it provides a general extension to the capabilities of the device. An overview of the technology of pen input devices, including interface products available in 1995, was provided by Meyer [40]. Currently available products are, for example, Microsoft's Tablet PC [45] and ZyonSystems' i-Pen [63], 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 [60]. 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, 54]). There are some users, whose physical conditions are even more limiting, and 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 [59]. 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 noteworthy that interfaces created by the SymbolDesign method work with a small amount of processing resources, do not require additional hardware, and do 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 interfaces 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, 53, 56, 62] 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 their 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, 61] 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 was also observed in the context of the work presented in this paper, and that the interfaces created by SymbolDesign use for distinguishing consecutive input symbols.

MacKenzie and Zhang [37] evaluated the usability of the "Graffiti" alphabet, which 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 contains only single-stroke characters. Isokoski and Raisamo [25] developed and evaluated an alphabet with single-stroke characters 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) [61], and extended

it with the SHARK^2 system [32, 62]. In these languages, words are represented by the spatiotemporal 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 evaluated an alphabet of single-stroke gestures to enter music notation into a computer. Long et al. [36] reported several user studies in which a number of gesture alphabets, each containing up to 13 single-stroke gestures, were presented to subjects. 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 the problem discussed in this paper and the problem of online handwriting recognition, which has been summarized in surveys by Tappert et al. [53], Wakahara et al. [56], 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, including 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 through matching techniques [53, 56] that compare the input pattern with stored reference patterns or evaluate it with a classifier, such as a neural net [35], a hidden Markov model [48, 49], or a stochastic network [5, 29].

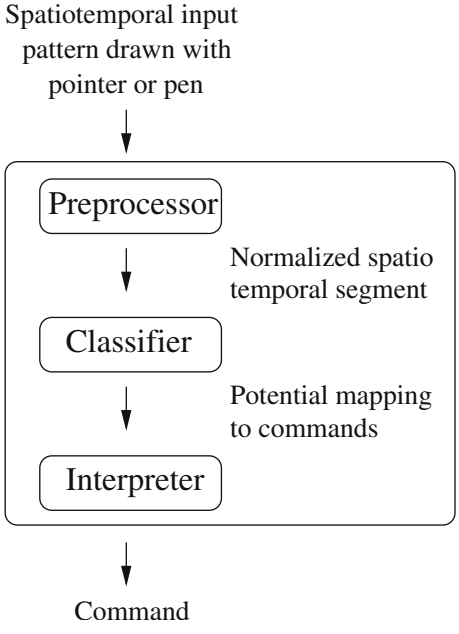

Fig. 1 Overview of the three system components, and their inputs and outputs, of an interface created by the SymbolDesign method

The core of the SymbolDesign system, first introduced in [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], a new method was developed in the context of the work presented in this paper that creates one-hidden-layer neural networks dynamically and efficiently, i.e., the network accommodates the three requirements listed above.

The paper is organized as follows. Section 2 provides 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 Sect. 5. The experiments and results are presented in Sect. 6. The paper concludes with an in-depth discussion in Sect. 7.

2 System overview

An interface designed using the SymbolDesign method consists of three major components: a preprocessor, a classifier, and an interpreter (Fig. 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 and 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.

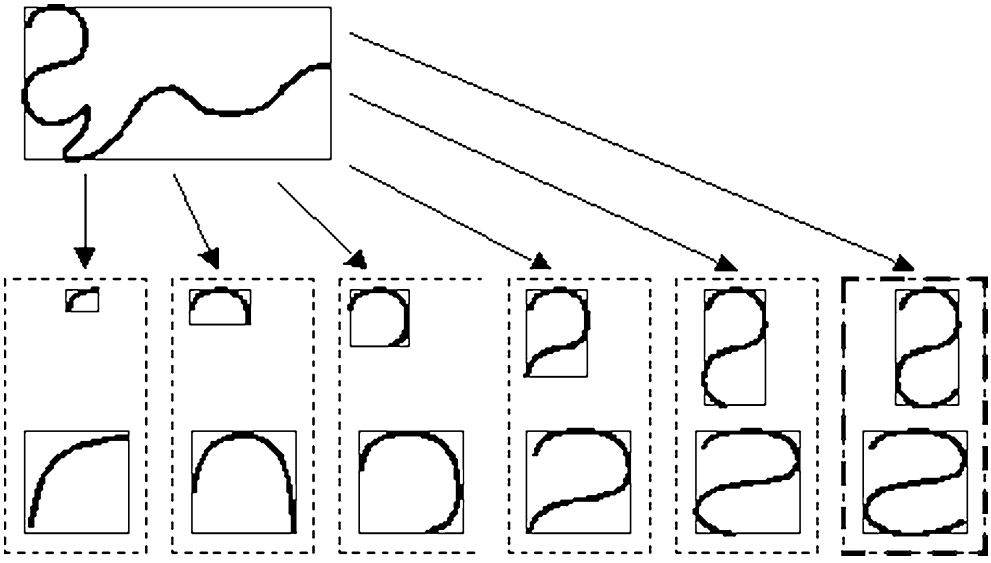

Fig. 2 Preprocessing of spatio-temporal patterns: The input pattern (*top*), which was drawn from left to right, is processed in segments (*bottom row*). Each segment includes the previous segment plus additional positions on the input path. The bounding boxes of the segments are resized to fit a fixed 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 postprocessing steps were performed

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 (Fig. 2).

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 that it does not exceed a length limit. Moreover, positions older than a certain age limit (e.g., 1 s) 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 those 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 Fig. 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

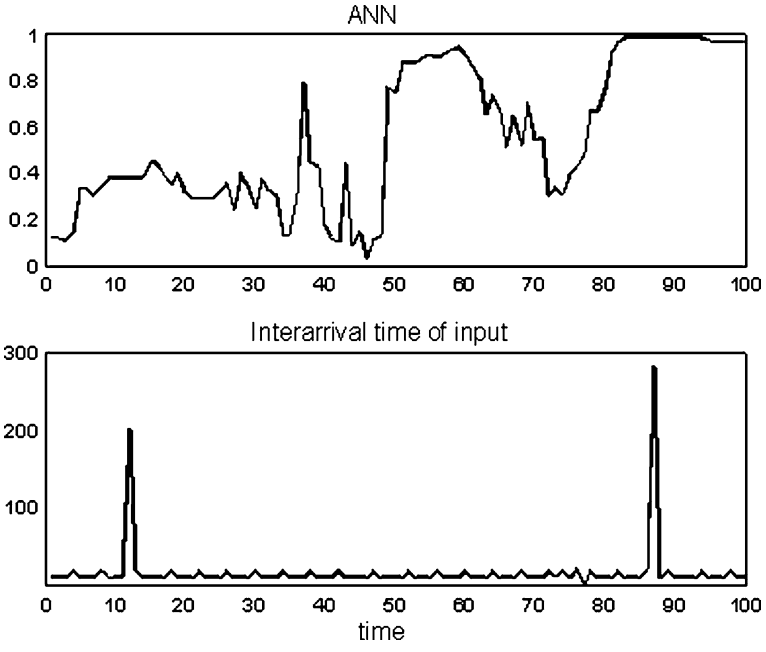

Fig. 3 Using the rate of input events to choose a spatio-temporal pattern (STP). *Top*: Artificial neural network (ANN) output over a period of 100 time units. *Bottom*: Inter-arrival time of input events over a period of 100 time units. At time $t = 12$, a significant pause in pointer movement was registered, 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

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 in 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 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 end-point of an input path segment with a high classifier score corresponds to a peak in the inter-arrival time of input events (Fig. 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 (Fig. 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.

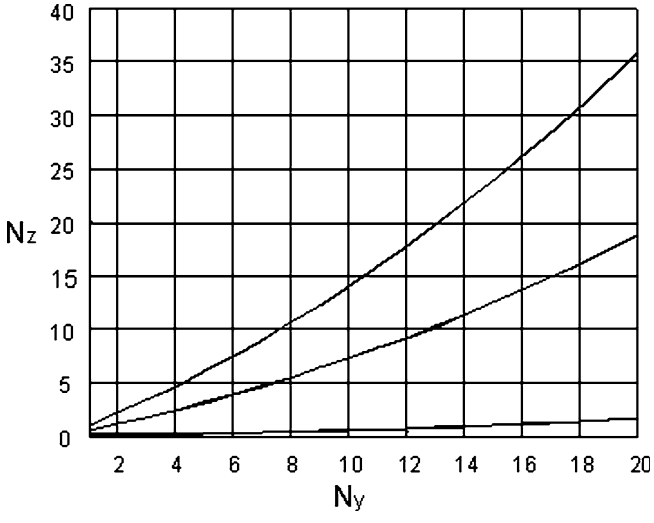

Fig. 4 Minimum, average, and maximum number of hidden nodes N_z as a function of $N_y = 1, \ldots, 20$ classes, 10 examples per class ($N_p = 10N_y$), and $N_x = 256$ input nodes

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 may also include in his or her symbol alphabet. For this reason, it is necessary to complete the processing of 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, 55]. 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., the "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. An adaptive network design has been adopted 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 be minimized. 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 has been elaborated. 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.

The elaborated 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

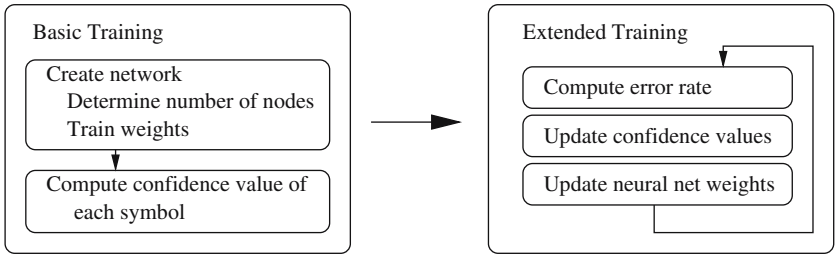

Fig. 5 Overview of the two training stages. In the basic training stage, a few instances of each symbol are used to create and train the neural network. For each symbol, a value is computed which represents the system's confidence that the network can reliably recognize the symbol. In the extended training 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

layers of the network [16]. The number of training examples used by this method is unfeasible for the current 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 [58]. The 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 Sjogaard's approach and the SymbolDesign 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 the latter, the size of the hidden layer is selected according to the complexity of the classification task. The SymbolDesign method thus avoids testing architectures that are unlikely to deliver the potential required for the given task. The way in which the SymbolDesign method determines the size of the hidden layer for the initially tested network is described below.

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_ω of weights [57]:

$$\frac{N_y N_p}{1+\log_2 N_p} \leqslant N_\omega \leqslant N_y\left(\frac{N_p}{N_z}+1\right)\left(N_x+N_y+1\right)+N_y \qquad (1)$$

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

$$N_z = \frac{N_\omega}{N_z+N_y} \qquad (2)$$

Minimum and maximum values for the number of hidden nodes can be computed by applying the respective bounds in Eq. 1 to N_ω in Eq. 2. The minimum value is needed for "easy" classification tasks, the maximum value for "hard" tasks. Since apriori 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. The number of hidden nodes was therefore initialized to the average of maximum and minimum values (Fig. 4).

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. The 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]:

$$N = \frac{\ln\left(1-F_w(\alpha)\right)}{\ln\left(1-F_x(\alpha)\right)} \qquad (3)$$

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 (Fig. 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.

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 decreased 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 initial experiences acquired working with two subjects with severe disabilities and provides some qualitative results.

6.1 Testing methodology of quantitative experiments

The 20 human subjects who participated in the quantitative experiments belonged to two groups. The 10 subjects in group A were sophomore college students with strong computer skills. The 10 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. The time it took each subject to configure the system, i.e., the ''average adjustment time'', was recorded. 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, the number of correctly performed actions out of the 50 attempted actions was recorded. The average processing time to recognize each symbol, i.e., the ''response time'', was also measured.

In the second experiment, the subjects were asked to expand their existing alphabets to include 10 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. The time it took each subject to configure the system, i.e., the average adjustment time, was recorded. For the test phase, the subjects were asked to launch the word processor and produce all 10 digits in consecutive order, using the mouse as a pointing device. In this phase, 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, were also recorded. 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 Fig. 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.

The 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 application.

6.3 Initial experience with users with quadriplegia

The SymbolDesign system was tested with two subjects with severe disabilities (see Fig. 7). Subject 1, a 28-year-old man, and subject 2, a 40-year-old woman, were both born with severe cerebral palsy that resulted in quadriplegia. The subjects were nonverbal, 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.

The primary method of communication used by the subjects' small head nods to select letters spelled out by an assistant or to directly answer the assistant's questions. Both subjects had experience using 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 the symbol recognition system. The subjects were asked to help with the design of symbols that they could draw comfortably. The starting point was the symbol O, which is used in the other experiments described in this paper as a selection command. The 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.

Simpler patterns were then tried, containing straight-lines, such as triangles. Finally the subjects were asked to draw symbols that just contained one line–either a horizontal, vertical, or diagonal line. The lines that the subjects were able to draw with the Camera Mouse were not straight, as can be seen in Fig. 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 on average 19.0 seconds for subject 2 to draw a line, with a standard deviation of 15.6 seconds. 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. They were asked 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 the symbol recognition system.

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	Av. recognition accuracy		Av. adjustment time	
Group	Task 1	Task 2	Task 1	Task 2
A (strong computer skills)	95 %	87 %	3 min	12 min
B (some computer skills)	85 %	75 %	6 min	18 min

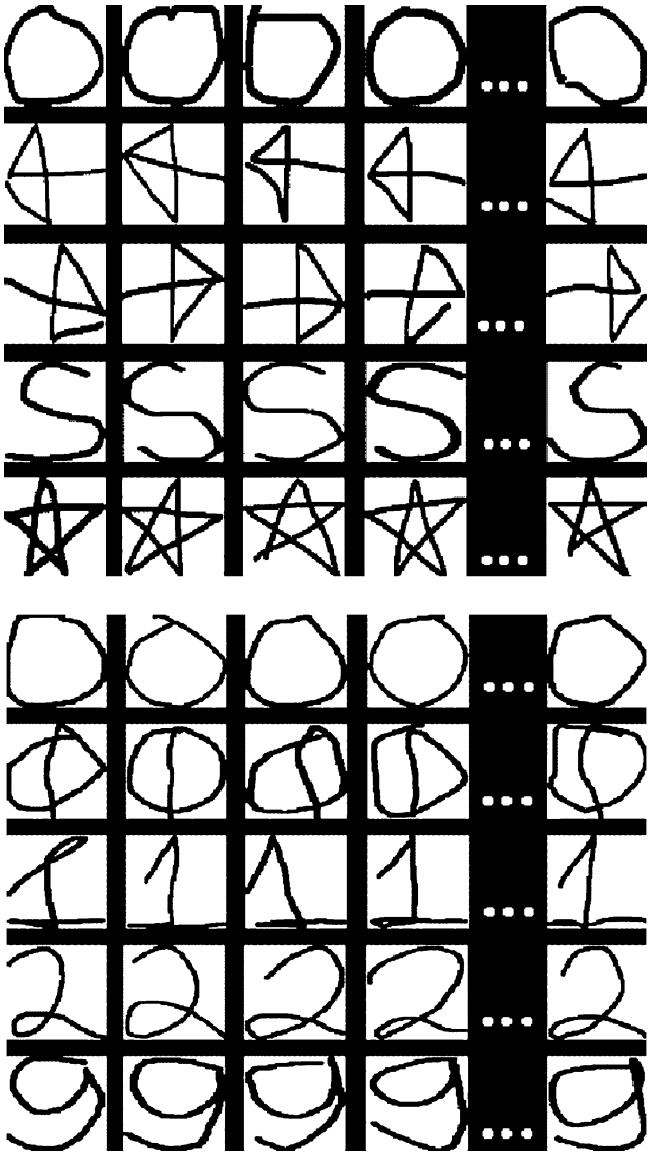

Fig. 6 *Top*: Instances of five symbols as produced by a user in Group B who was asked to perform Task 1. The user was asked to assign the symbol O to the "select" command ("left mouse click" command). The remaining four symbols were chosen by the user. Symbol $\rightarrow$ was chosen for the "forward" command, symbol $\leftarrow$ for the "backward" command, symbol *S* for the "stop" command, and symbol * for selecting "favorites." *Bottom*: 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 O

7 Discussion

The main contribution of the work presented in this paper 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 user 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. The proposed 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. Interfaces with alphabets of up to 15 symbols (10 digits and 5 browser commands) were tested. The number of symbols was 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 the conducted experiments, it can be concluded 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 the modular approach adopted 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.

Little prior work has been done to enable users to create their own interfaces, i.e., design symbols or gestures and map them to commands [52]. To the best of the authors' knowledge, there are no prior user studies where subjects were asked to invent symbols for computer input. The user studies conducted with the SymbolDesign method were aimed at evaluating whether it indeed followed the paradigm of user-centered, accessible design. The analysis focused on evaluating the errors of the interfaces in recognizing symbols, the amount of training time needed, and the average recognition time during use. The 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. Error rates were larger for less experienced

computer users than for experienced computer users, likely because they may have been hesitant and may have 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 this 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, it was impossible for the two subjects with severe cerebral palsy who participated in the experiments to train an interface and use it to surf the web. Nonetheless, the authors believe that developing interfaces that learn to recognize gestures that the users choose themselves is particularly important for people with severe disabilities. Flexible interface design may enable users with less severe motion impairments to select gestures that they have the abilities to perform.

The initial experiments showed that physical limitations precluded the subjects from gesturing spatio-temporal patterns such as lines or circles, but not from moving the pointer to certain regions on the screen and keeping it in these regions for a while. Therefore, a potential solution could be to further discretize the input patterns 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].

Future work will be divided into two areas of investigation: (1) further improvement the SymbolDesign technology, and (2) additional user studies to evaluate the interfaces created by the SymbolDesign technology. To improve the SymbolDesign technology, work will be conducted 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. It will also be tested 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 will be added 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, this use of screen space had been excluded, since screen space is so valuable for miniature computers. In future work, this idea will be revisited 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.

Fig. 7 Subject 1 (*left image*) and subject 2 (*middle image*) are using the Camera Mouse as a mouse-pointer substitution interface. Subject 2s attempt to draw a vertical line is shown in blue on the left monitor in the *right 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 idea to use a "mask" during the training process will also be investigated. 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 input errors during regular use, which is an important consideration in the context of the design paradigm proposed in this paper. 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 the SymbolDesign 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 spatiotemporal 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 sub-patterns that can be individual characters by themselves, as it occurs in Japanese. In the SymbolDesign application, a similar issue with sub-patterns 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, the size invariant, pause-dependent recognition method would identify the prefix instead of the full pattern and issue an unwanted command. In the conducted experiments, the users only chose prefix-free single-stroke patterns. To allow the user to work with patterns that contain prefix subpatterns, the system could be extended with the candidate lattice approach [42].

The 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.

It is planned 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 Fig. 6.

The users involved in the 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. User acceptance of pen- or pointer-based interfaces will be further investigated in order to establish whether it indeed increases when users are allowed to choose their own symbols. It will also be studied whether some users might be overwhelmed by the task of having to design their own distinct symbols and would prefer to work with a predefined alphabet. The issue of gesture similarity arises for anybody who designs a gesture alphabet [36]. An advantage of the 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. It will be investigated 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, nonverbal 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 for 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, the students of the Boston University Video Game Creators Consortium, and the anonymous reviewers for their 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. Bauby J-D (1997) The diving bell and the butterfly. Vintage Books
3. Betke M, Gips J, Fleming P (2001) The Camera Mouse: visual tracking of body features to provide computer access for people with severe disabilities. IEEE T Neur Sys Reh 10(1):1–10
4. CameraMouse, Inc. (2005) Hands-free computer control. http://www.cameramouse.com Abilene, TX, USA

5. Cho SJ, Kim JH (2001) 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, pp 86–90
6. Cloud RL, Betke M, Gips J (2002) Experiments with a camera-based human-computer interface system. In: 7th ERCIM workshop on user interfaces for all, pp 103–110, Paris, France
7. Crampton SC, Betke M (2003) 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), pp 1357–1361 Crete, Greece
8. Crane HD, Ostrem JS, Edberg PK (1985) Method for distinguishing between complex character sets. US Patent 4 531 231
9. Cun YL, Bottou L, Orr G, Muller K (1998) Efficient backprop, neural networks: tricks of the trade. In: Lecture notes in computer sciences 1524, pp 5–50. Springer-Verlag
10. Diamond Bullet Design (2005) Usability first. http://www.usabilityfirst.com, Ann Arbor, MI, USA
11. DiMattia P, Curran FX, Gips J (2001) An eye control teaching device for students without language expressive capacity–eagleeyes. The Edwin Mellen Press. See also http://www.bc.edu/eagleeyes
12. Don Johnston, Inc., Penny & Giles HeadWay, infrared head-mounted mouse alternative. http://www.donjohnston.com
13. Duda RO, Hart RE, Stork DG (2001) Pattern classification, 2nd ed. John Wiley & Sons, New York
14. Evgeniou T, Pontil M, Poggio T (2000) Regularization networks and support vector machines. Adv Comput Math 13:1–50
15. Fagiani C, Betke M, Gips J (2002) Evaluation of tracking methods for humancomputer interaction. In: IEEE workshop on applications in computer vision, pages 121–126, Orlando, Florida
16. Fahlman SE, Lebiere C (1990) The cascade-correlation learning architecture. In: Touretzky DS (ed) Advances in neural information processing systems, vol 2. Denver, CO, USA, Morgan Kaufmann, San Mateo, pp 524–532
17. European Research Consortium for Informatics and Mathematics (ERCIM) (2005) User interfaces for all. http://ui4all.ics.forth.gr/workshop2004/technical-description.html
18. Forsberg A, Dieterich M, Zeleznik R (1998) The music notepad. In: Proceedings of the 11th annual ACM symposium on user interface software and technology (UIST '98), ACM Press, pp 203–210
19. Frankish C, Hull R, Morgan P (1995) Recognition accuracy and user acceptance of pen interfaces. In: Proceedings of the SIGCHI conference on human factors in computing systems (CHI '95). ACM Press/Addison-Wesley Publishing Co., pp 503–510
20. Goldberg D, Richardson C (1993) Touch-typing with a stylus. In: Proceedings of the SIGCHI conference on human factors in computing systems (CHI '93). ACM Press, pp 80–87
21. Grauman K, Betke M, Lombardi J, Gips J, Bradski GR (2003) Communication via eye blinks and eyebrow raises: video-based human-computer interfaces. Univ Acc Inform Soc 2(4):359–373
22. Gusyatin O, Urinson M, Betke M (2004) A method to extend functionality of pointer input devices. In: Stary C, Stephanidis C (eds) Proceedings of the 8th international ERCIM workshop on user interfaces for all, Revised Selected Papers, Lecture notes in computer science 3196. Springer-Verlag, Vienna, Austria, pp 426–439
23. Hecht-Nielsen R (1987) Kolmogorov's mapping neural network existence theorem. In: Proceedings of IEEE first annual international conference on neural networks, vol 3. San Diego, CA, pp 11–13
24. Hutchinson T, White KP Jr, Martin WN, Reichert KC, Frey LA (1989) Human-computer interaction using eye-gaze input. IEEE T Sys Man Cybernet 19(6):1527–1533
25. Isokoski P, Raisamo R (2000) Device independent text input: a rationale and an example. In: Proceedings of the working conference on advanced visual interfaces (AVI '00), Palermo, Italy. ACM Press, pp 76–83
26. Iyer MS, Rhinehart RR (1999) Method to determine the required number of neural network training repetitions. IEEE T Neural Networ 10(2):427–432
27. Ji Q, Zhu Z (2004) Eye and gaze tracking for interactive graphic display. Mach Vis Appl 15(3):139–148
28. Kang H, Lee CW, Jung K (2004) Recognition-based gesture spotting in video games. Pattern Recog Lett 25(15):1701–1714
29. Kang K-W, Kim JH (2004) Utilization of hierarchical, stochastic relationship modeling for Hangul character recognition. IEEE T Pattern Anal 26(9):1185–1196
30. Kapoor A, Picard RW (2002) Real-time, fully automatic upper facial feature tracking. In: Proceedings of the fifth IEEE international conference on automatic face gesture recognition, Washington, DC, pp 10–15
31. Kim K-N, Ramakrishna RS (1999) 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
32. Kristensson P-O, Zhai S (2004) 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
33. LC Technologies, Eyegaze System. http://www.lctinc.com
34. Lee HK, Kim JH (1999) An HMM-based threshold model approach for gesture recognition. T Pattern Anal 21(10):961–973
35. Leung W, Cheng K (1996) 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. ACM Press.
36. Long AC Jr, Landay JA, Rowe LA, Michiels J (2000) Visual similarity of pen gestures. In: Proceedings of the SIGCHI conference on human factors in computing systems (CHI '00). The Hague, The Netherlands. ACM Press, pp 360–367
37. MacKenzie IS, Zhang S (1997) The immediate usability of graffiti. In: Proceedings of graphics interface '97, pages 129–137, Toronto, Canada. Canadian Information Processing Society.
38. Madentec. http://www.madentec.com
39. Magee JJ, Scott MR, Waber BN, Betke M (2004) EyeKeys: A realtime vision interface based on gaze detection from a low-grade video camera. In: 2004 Conference on computer vision and pattern recognition workshop (CVPR'04), vol 10, Workshop on real-time vision for human-computer interaction (RTV4HCI), Washington, DC, IEEE Computer Society, pp 159–166
40. Meyer A (1995) Pen computing: a technology overview and a vision. ACM SIGCHI Bulletin 27(3):46–90
41. Morimoto CH, Koons D, Amir A, Flickner M (2000) Pupil detection and tracking using multiple light sources. Image Vision Comput 18(4):331–335
42. Murase H (1988) Online recognition of free-format Japanese handwritings. In: Proceedings of the 9th international conference on pattern recognition, vol 2. Rome, Italy, November 1988. IEEE Computer Society Press, pp 1143–1147
43. Oka K, Sato Y, Koike H (2002) Real-time fingertip tracking and gesture recognition. IEEE Comput Graph 22(6):64–71
44. Plamondon R, Srihari SN (2000) Online and offline handwriting recognition: a comprehensive survey. IEEE T Pattern Anal 22(1):63–84
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. Riviere C, Thakor N (1996) Assistive computer interface for pen input by persons with tremor. IEEE Eng Med Biol 15(3):29–36
47. Sears A, Arora R (2002) Data entry for mobile devices: an empirical comparison of novice performance with jot and graffiti. Interact Comput 14:413–433
48. Sin BK, Kim JH (1993) 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, pp 147–150

49. Sin B-K, Kim JH (1997) Ligature modeling for online cursive script recognition. IEEE T Pattern 19:623–633
50. Sjogaard S (1991) A conceptual approach to generalization in dynamic neural networks. Neural Comput 2:198–209
51. Steriadis CE, Constantinou P (2003) Designing human-computer interfaces for quadriplegic people. ACM T Computer-Human Interact (TOCHI) 10(2):87–118
52. StrokeIt software, developed by Je. Doozan. Website accessed in April 2005. http://tcbmi.com/strokeit
53. Tappert CC, Suen CY, Wakahara T (1990) The state of the art in on-line handwriting recognition. IEEE T Pattern 12(8):787–808
54. Tash solutions. Website accessed in April 2005. http://www.tashinc.com
55. Vapnik VN (1998) Statistical learning theory. John Wiley & Sons, New York
56. Wakahara T, Murase H, Odaka K (1992) On-line handwriting recognition. Proceedings of the IEEE 80(7):1181–1194
57. Widrow B, Lehr MA (1990) 30 years of adaptive neural networks: Perceptron, madaline, and backpropagation. Proceedings of the IEEE 78(9):1415–1442
58. Yeung D-Y (1991) Node splitting: A constructive algorithm for feed-forward neural networks. In: Moody JE, Hanson SJ, Lippman RP (eds) Advances in neural information processing systems, vol 4. Morgan Kaufman Publishers, San Mateo, CA, pp 1072–1079
59. Yoon HS, Soh J, Bae YJ, Yang HS (2001) Hand gesture recognition using combined features of location, angle and velocity. Pattern Recog 34(7):1491–1501
60. Young L, Sheena D (1975) Survey of eye movement recording methods. Behav Res Meth Instrument 7(5):397–429
61. Zhai S, Kristensson P-O (2003) Shorthand writing on stylus keyboard. In: Proceedings of the conference on human factors in computing systems (CHI '03), ACM Press, pp 97–104
62. Zhai S, Kristensson P-O, Smith BA (2005) In search of effective text input interfaces for off the desktop computing. Interact Comput 17(3):229–341
63. ZyonSystems (2005) i-Pen–Presentation Digital Pen / Optical Pen Mouse. http://www.zyonshop.com/product/ipen.htm Seattle, WA, USA.