AN OBJECT-ORIENTED CHARACTER RECOGNITION ENGINE

B. Klauer K. Waldschmidt
J.W.Goethe - Universität Frankfurt, Professur für Technische Informatik
Robert Mayer Straße 11 - 15 D - 60054 Frankfurt am Main
R. Heinrich
INFOS GmbH
Ludwigstr. 78, D - 63067 Offenbach

Tagungsband
Euro-ARCH '93
Informatik aktuell
Springer 1993

Abstract

This paper introduces the object-oriented character recognition engine AQUIRE which was originally designed as a general pattern-classifier under the academic aspects of object-orientation and parallelism together with low space and time complexity. When industry expressed interest in using AQUIRE to provide pen-based computers with a powerful character-recognition engine, AQUIRE was revised to meet the commercial requirements. Together with a brief discussion of some special pen-related problems, this paper contains the theoretical background for the recognition mechanism and a description of the methods used to perform a high quality and high speed recognition-process with a minimum of executable code. The proposed method has been implemented on an INFOS NotePad 386-SX pen-computer and on the associative processor AM^3 developed within the PROMETHEUS project at the Department for Technical Computer Sciences at J.W.Goethe-University. The acceleration of the recognition mechanism by using the associative type of parallelism will be shown.

1. Pen computing

Pen-computer environments offer a lot of opportunities to their users but also a lot of restrictions to hardware and software designers. The first and most obvious opportunity of pen computers on one hand and restriction on the other hand is their physical size and the fact that they are intentionally used with a pen instead of a keyboard as user-input device. From a user's point of view nothing is more familiar than processing data using pen and paper. Pen-based knowledge- and data-aquisition can help to lower the barrier between non-experienced users and computers and terminals. It can help experienced users to enter commands and data in a more natuaral way.

From a programmer's point of view the replacement of the keyboard by a pen requires pen-based user interfaces, providing a complete control mechanism for a pen as central input device. The User-Interface has to provide comfortable methods to handle the control and data streams to an application. The control stream can easily be handled using window and menu techniques which are the state-of-the-art in modern applications even in non pen-related environments. The data entry is more complex. Menu techniques, like so-called virtual keyboards are usable for data entry but less comfortable than data input via handprinted characters or digits.

This feature requires a highly sophisticated character recognition engine to be embedded into pen-based applications. The recognizer can be used to transform drawings like characters or gestures into their semantics. The semantics can then be used to trigger functions in case of gestures or as character or digit in case of character like entries.

The physical and logical restrictions and requirements as mentioned above raised academic and commercial interest in the developement of a powerful recognition engine for handwriten symbols with the following outlines:

- Small executable code
- Small database size
- High accuracy
- Directness (speed)

"Directness" means that users should have the feeling of controlling the system directly with the tip of the pen. Actions of the application should follow immediatly after a handprinted symbol has been completed. [RHY86] contains proposals how the completeness of handprinted symbols can be decided.

1.1 Design-decisions

Due to the above mentioned restrictions and requirements the following design decisions concerning the archtitecture of the recognizer and it's implementation have been made:

1.1.1 Online-Recognition vs. Offline-recognition

In comparison to off-line recognition, on-line recognition performs recognition while the pattern is beeing drawn on any kind of digitizing device. Directness, which means that user actions like gestures are immediately translated into machine action is an ergonimic requirement for pen-related applications. The directness usually implies online-recognition systems in pen-based applications.

1.1.2 Static recognition vs. dynamic recognition

Static character recognition evaluates patterns without looking on how the pattern has been created. Dynamic recognition is focussed on strokes, edges in strokes, drawing speeds and accelerations in the drawing history. AQUIRE is a heterogeneous recognition engine combining a static (PATTMA) and a dynamic (MAGIC) module.

1.1.3 Source Language

Due to the portability of the source-code the recognizer has been programmed in C++. This decision seems to be contradictory to the size and speed requirement but is actually not. Analyzing the code produced by the Borland C++ compiler used for the MS-DOS version, it can be observed that it generates efficient machine-code. The runtime-libraries and the startup-code are the most space-expensive modules due to their generality. Special startup and runtime routines have been designed to gain efficiency in size and speed.

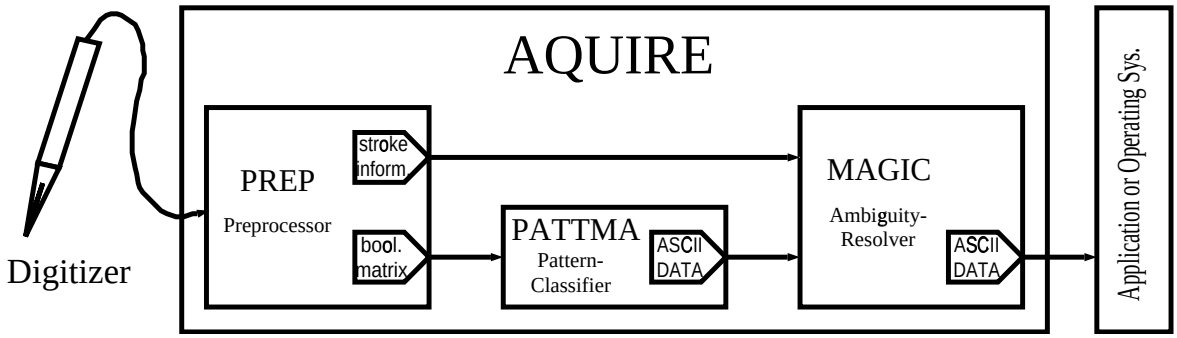

figure 1

1.1.4 Software architecture

The recognizer consists of three modules a preprocessor (PREP), a pattern-matcher (PATTMA) and an ambiguity resolver (MAGIC: **MA**nual **G**esture **I**nconsistancy **C**learance) as shown in figure 1. The PREP module contains an Application Programming Interface (API) and an extractor providing the PATTMA and MAGIC modules with significant information on the symbol to be recognized. PREP works "on-line" during the drawing phase. It passes a boolean pattern to the PATTMA pattern matcher and stroke-information to the MAGIC module. PATTMA is a very fast pre-classifier but is in case of optically similar characters such as "B" and "8" or "b" and "6" insufficient in recognition accuracy. MAGIC detects - and resolves ambiguities. The resolving mechanism is activated only in case that an ambiguous pattern has been detected.

2. Methods

2.1 The PREP module

PREP provides the Application Programming Interface (API) and an extractor to retrieve significant information from the input data. The API for the Intel 80x86 processors consists of an interrupt-service routine supporting one of the 80x86 processors software interrupts. The API for the AM^3 processor consists of a set of C++ functions to control the recognizer. It provides the following functions:

1. Setup-functions for the recognition engine
2. Aquisition and storage of handdrawn symbols as defined below
3. Functions to control the recognition of patterns

2.1.1 Recording and Scaling

The following expressions be informally defined:

Def 1.0 Handdrawn Symbols
Let a handdrawn symbol be defiened as a sequence of strokes

Def 1.1 Strokes
Let a stroke be defined as a sequence of coordinates

The strokes as defined above are passed to the PREP module coordinate by coordinate via the API. Strokes are separated by the coordinate ($ffff_{hex}$,$ffff_{hex}$) which is defined to be invalid for coordinates within a stroke. PREP connects adjacent coordinates whit a straight line. The minimum and maximum x and y coordinates are recorded online. All coordinates are then scaled to fit into into a 32*32-bit boolean matrix.The boolean matrix will later be converted into a 1024-bit vector to provide the PATTMA module with data.

2.1.2 Extraction of significant information

A mask and a search argument are computed by the extractor and sent to the PATTMA module. The Pattern matcher computes the semantics by retrieving the most similar pattern in it's database. The ambiguity resolver is also provided with data from the extractor. If envoked, MAGIC consumes the total of strokes, start- and end-coordinates, as well as the start- and end-directions of the strokes, to examine ambiguous characters.

2.2 Methods used in PATTMA

The description of the basic classification mechanism of PATTMA requires formal definitions of some important expressions:

Def 2.0 Patterns
Let a pattern be a boolean vector

P = $[p_i]$; $1 \leq i \leq L$;
L : length of the pattern

An element p_i of a vector P is called a pixel. The value of the pixel corresponds to it's color (e.g. black and white in case of boolean patterns). All pattern-data used in PATTMA is derived from the handprinted input symbols by the extractor of the PREP module. PREP converts handdrawn symbols (Def 1.0) into images (Def 2.1) with unknown semantics.

Def 2.1 Images
Let an image be a tupel I consisting of a pattern P and the symbol s indicating the semantics of the pattern P.

I = (P,s); $s \in S$

s is a member of the set S of all symbols. S must contain a symbol ϕ indicating that the semantics of a symbol is unknown. All well known images have the semantics $s \neq \phi$. The database used by PATTMA contains images with semantics $s \neq \phi$. Unknown images as derived from the PREP module from handprinted symbols have a semantic $s = \phi$.

Def 2.2 Hamming Distance
The Hamming Distance HD(A,B) of two patterns is the total of all pixels a_i,b_i with $a_i \neq b_i$.

$HD(A,B) = \Sigma (a_i, \oplus b_i)$
$\oplus$: boolean XOR

Def 2.3 The Masked Hamming Distance
The Masked Hamming Distance MHD of two patterns A,B and a mask M is defined as follows:

$MHD(A,B,M) = \Sigma ((a_i \oplus b_i) \; m_i)$
: boolean AND
: boolean NOT

MHD behaves as HD if M = $\underline{0}$. In this case MHD computes the total of all different elements a_i,b_i with $a_i \neq b_i$ which are not hidden by m_i. HD as well as MHD can be used as similarity indicators in recognition systems. The following method can be used to find the semantics of an unknown image U = (X,ϕ):

Let W = $\{(P_1,s_1),(P_2,s_2), ...\}$ be a set of well known images.

1. Compute the mask argument M
   (A method to compute the mask will be shown in the following paragraph)
2. Compute MHD(X,P_i,M) for all i
3. Replace ϕ with s_i if MHD(X,P_i,M) is minimal for all i

2.2.1 The mask-argument of MHD

As one might expect the computation of the mask-argument is the key to obtain good recognition results using the MHD function.To classify handprinted characters using the MHD classifier the following informal method can be used to compute the mask:

Let X be the pattern of an unknown image
Let M be the mask argument to be computed

A function BOLD(PATTERN A, INTEGER N) returning a boolean pattern, be defined as follows:

1. if N>0 set R:=A else R:=0
2. repeat N times
   for all a_{ij} with $a_{ij} = 0$ do
   if an adjacent element of $a_{ij} = 1$ then
   set r_{ij}:=1
3. return(R)

The Mask M can then be computed as follows:

1. set Bold_X :=BOLD(X,ntimes)
2. set Bold2_X := BOLD(BX,mtimes)
3. set M := Bold2_X-Bold_X

The result of the computation is shown in figure 2. The mask has been derived from pattern A with the parameters ntimes=1 and mtimes=0.

2.2.2 Hamming-Distance vs. Masked Hamming- Distance

The Hamming-Distance classifier is very popular to compute a similarity indicator for boolean patterns [KOH87]. The Masked Hamming-Distance distinguishes from the pure Hamming-Distance in the simple fact that specific areas of the patterns to be compared can be hidden. The MHD classifier computes the Hamming-Distance of all bits which are not masked. This special feature can be used for fault-tolerant classification.
Example:

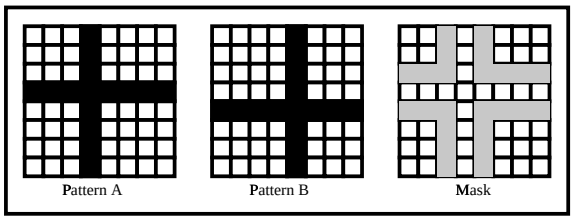

figure 2

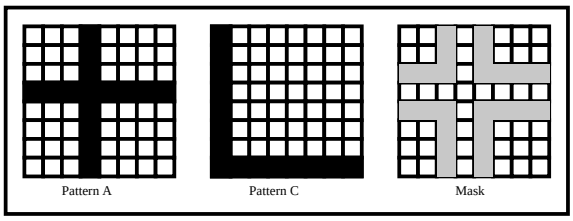

figure 3

Figure 2 shows two "+ " symbols which are similar but not equal. A recognition system holding one of both symbols in it's database should classify them to be similar.The Hamming-Distance of pattern A and B is 26, indicating that both patterns are different even if they are not from a human point of view. The masked Hamming-Distance of A and B and a mask argument as shown in figure 2 is 13, indicating that both patterns are similar. This result is more adequate to the human impression. The next example (figure 3) shows that the MHD classifier works also in case that both symbols are different. In this case MHD(A,C,Mask) = 24, indicating that both symbols are much more different. Figure 4 shows a statistical evaluation of 135 handdrawn symbols to find optimal values for the parameters "ntimes" and "mtimes" for 1024-bit patterns (32*32-bit). It shows that the pure Hamming-Distance

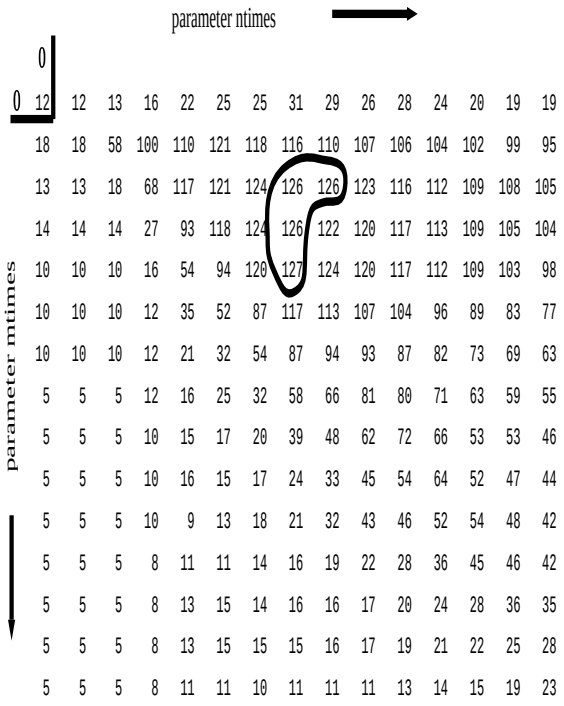

figure 4

(ntimes=mtimes=0) is not optimal for the pattern-classification of human handprinted characters. It shows an optimum at ntimes=7 and mtimes=4. The parameters depend on the database-size, the type of symbols, the cardinality of the alphabet and on the dimension of the boolean matrix used to store the patterns.

2.3 Methods used for MAGIC

The ambiguity resolver MAGIC consists of a set of routines to detect and resolve ambiguities. Handprinted characters from 102 Persons from different countries on different continents have therefore been statistically evaluated to find ambiguities in the human style of handwriting. Ambiguities will be denoted as follows:

$(A_1,A_2,...A_n)$

The intended meaning of the above expression is that symbol $A_1,A_2,...A_n$ cannot properly be distinguished by their optical image as represented by the 32*32 bit matrix.

Example:

(Z,2)

(O,0)

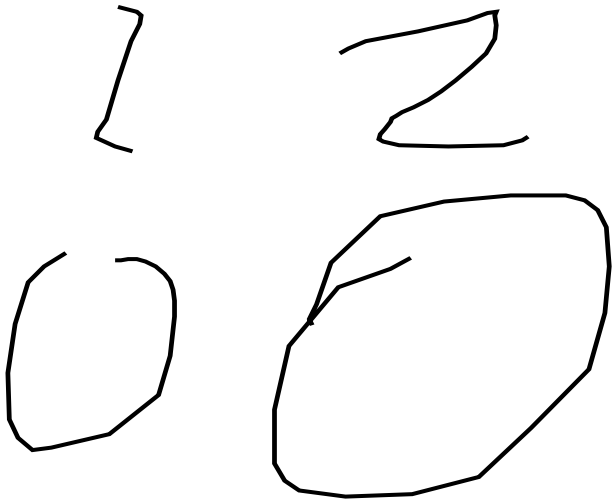

figure 5

All ambiguities detected within the evaluation have been classified to be *hard* or *soft*. Soft ambiguities can in most cases be resolved by counting and tracing the strokes as in the (B,8) ambiguity. Hard ambiguities cannot be resolved by stroke-evaluation as in the (Z,2) ambiguity. Figure 5 shows two symbols of a 2 and a Z and two examples of an O and a 0 recorded from different writers. Even a human can not classify them correctly without any context. To solve the above mentionened problems rules have been stated for the writers to gain proper recognition. E.g. a Z must always be drawn with a horizontal stroke at it's center. MAGIC contains a subroutine for each ambiguity which has been statistically detected. The routines to resolve the hard ambiguities require that specific rules be obeyed by writers.

3. Implementations

The first version of AQUIRE has been programmed in C++. The three major modules PREP, PATTMA and MAGIC have been programmed as separate C++ classes. The PATTMA module has been derived from a module called CAM (s. figure 6). CAM performes a retrieval of the best matching database item using the MHD classifier.

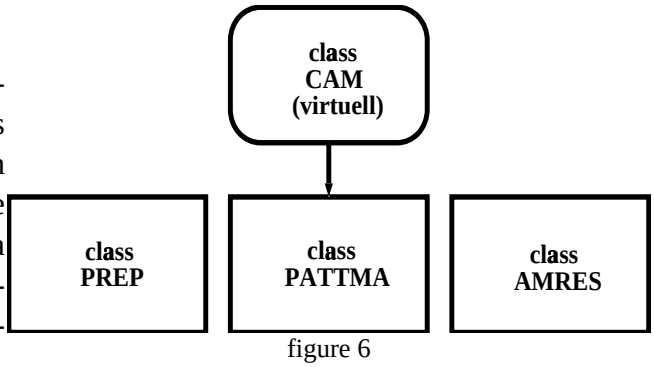

figure 6

3.1 The MS-DOS implementation

The MS-DOS implementation uses a CAM module which computes the MHD classifier for all database items sequentially. It provides functions to handle the database containing the set of known images.The runtime-behaviour has then been analyzed and some critical routines have been reprogrammed in assembly language. Portability has been lost in the optimized assembly language version but speed was significantly increased. The size of the executable program was reduced. Application programmers have access to the recognizer via the software interrupts of the Intel 80x86 processors. This provides an interface which is completely independent of languages and developement environments. The Application Programming Interface (API) provides the following functions:

INITIALIZE
This function presets all global variables with proper initial values

INSERT COORDINATE
This function is used to pass a coordinate of a stroke to the PREP module for preprocessing.

INSERT STROKE SEPARATOR
This function is used to declare a stroke consisting of previously entered coordinates to be complete. It is envoked by inserting the coordinate ($ffff_{hex}$,$ffff_{hex}$) with the INSERT COORDINATE function. ($ffff_{hex}$,$ffff_{hex}$) is defined to be invalid within a stroke.

ENABLE/DISABLE MAGIC
The MAGIC module has been designed to resolve ambiguities in the recognition process of handprinted latin characters. It can not optimize the recognition process for arbitrary symbolsets. To recognize arbitrary non-latin character sets MAGIC should be turned off.

SELECT DRAWING AREA
A drawing area is an invisible ribbon specified by it's upper and lower bounds. It is used by MAGIC to distinguish ambiguous characters like (Ww). AQUIRE also supports a so called free-drawing mode. In this mode the screen is divided into two halfes. Characters drawn into the lower half of the screen are always converted into lower-case. Characters drawn into the upper half are recognized as they are in upper- or lower-case with upper-case preference in case of (Ww)-like ambiguities.

RECOGNIZE
This function starts the recognition process for previously inserted and preprocessed data. It returns an integer value representing the semantics of the unknown symbol as result. It has been derived from the most-similar database-item. An important problem to be mentioned in this context is the so called CLOSURE PROBLEM. It discribes the decision of the question *Is the symbol complete now?* occurring after an arbitrary stroke has been drawn. [RHY86] contains a discription and solutions to the closure problem. Good results have been made with the following informally discribed methods.

1. Time method
A symbol be complete if a pen-up condition for more than 500ms has been detected.

2. Space method
A visible grid be defined on the screen. Characters are assumed to be drawn into the grid spaces. A symbol be complete if the user starts a stroke in a new grid space.

3.2 The AM^3 implementation

The AM^3 processor [SCH89,SCH92, DAR90] is an associative 32-bit processor with a modified Harvard-architecture. It has one bus to handle the data and instruction streams and a separate addressless associative bus-system to provide a link to a varity of associative memory components. One basic function which is supported by all memory components within the AM^3 is the masked search function. The masked search function has been described by Kohonen [KOH87] as one of the basic associative functions.The above mentioned character recognition method was mapped onto the AM^3 processor by replacing the CAM module implemented as virtual base class in C++ by a compatible module which uses directly the associative functions of the AM^3 processor.

3.3 Performance parameters

The most important performance parameters of a character recognizer for pen-computers are code-size, database-size, speed and accuracy. The recognition speed and the size of the executable code are machine and operating-system dependent while database-size and accuracy are pure method-related parameters.

3.3.1 MS-DOS implementation

executable code size: ~18KB

recognition speed: ~ 80 s per character in the database on a 486 machine running at 33MHz

3.3.2 AM^3 implementation

One goal of the AM^3 implementation was to show that the method as proposed is suitable to be computed and to be accelerated by associative hardware. The academic intention was to show a sub-linear run-time behaviour with a growing database-size. Due to the memories used within the AM^3, a logarithmic run-time behaviour could be measured. Due to the low system clock of the machine (4 MHz) the AM^3 is much slower for reasonable database-sizes than the MS-DOS implementation.

3.3.3 Machine independent parameters

Database-size and accuracy are tighly related parameters. They depend on each other and on the cardinality of the alphabet to be recognized. The parameters as presented below require some comments on how they have been measured since recognition accuracy of recognizers can in general be very good with cooperative users and very bad with users intentionally trying to fool the system. The following reasonable experiment has been performed to measure the recognition quality: The handprinted character sets (upper-case, lower-case and numeric) of 102 people have been evaluated. A character set of 600 characters has then been extracted from the complete set as database. The other characters have been used as test-patterns. The recognition rate has been measured under 2 conditions. In the first test only rule consistant characters (see Methods used for MAGIC) have been used. In the second test all characters have been exposed to the recognizer. The following recognition rates under the above mentioned conditions could be measured:

97% Recognition-rate with rule consistant characters

92% With an arbitrary character set

4. Applications

4.1 Commercial application

In tight cooperation with INFOS GmbH AQUIRE has been optimized for the INFOS NotePad 386-SX computer. It is commercially available with the INFOS Notepad computer.

4.1.1 The INFOS NotePad 386-SX computer

The INFOS NotePad computer is standard 386-SX machine running at 20MHz under the operating system MS-DOS. It has a backlit LCD screen with a pen-digitizer and two PCMCIA slots. It is provided with 4MB RAM and serial and parallel interfaces. Via it’s keyboard interface it can be connected with an optional physical keyboard. The complete machine fits into a 28mm(H)*335mm(L)*270mm(W) package.

4.1.2 The pure recognition engine

The recognition-engine can be loaded as TSR (terminate and stay resident) module into the standard DOS memory area or into the high-memory area. Applications can then access the recognizer via it’s API. The recognition engine comes together with a training-program to improve the recognition-rate for individual styles of handwriting or to create customized symbol sets.

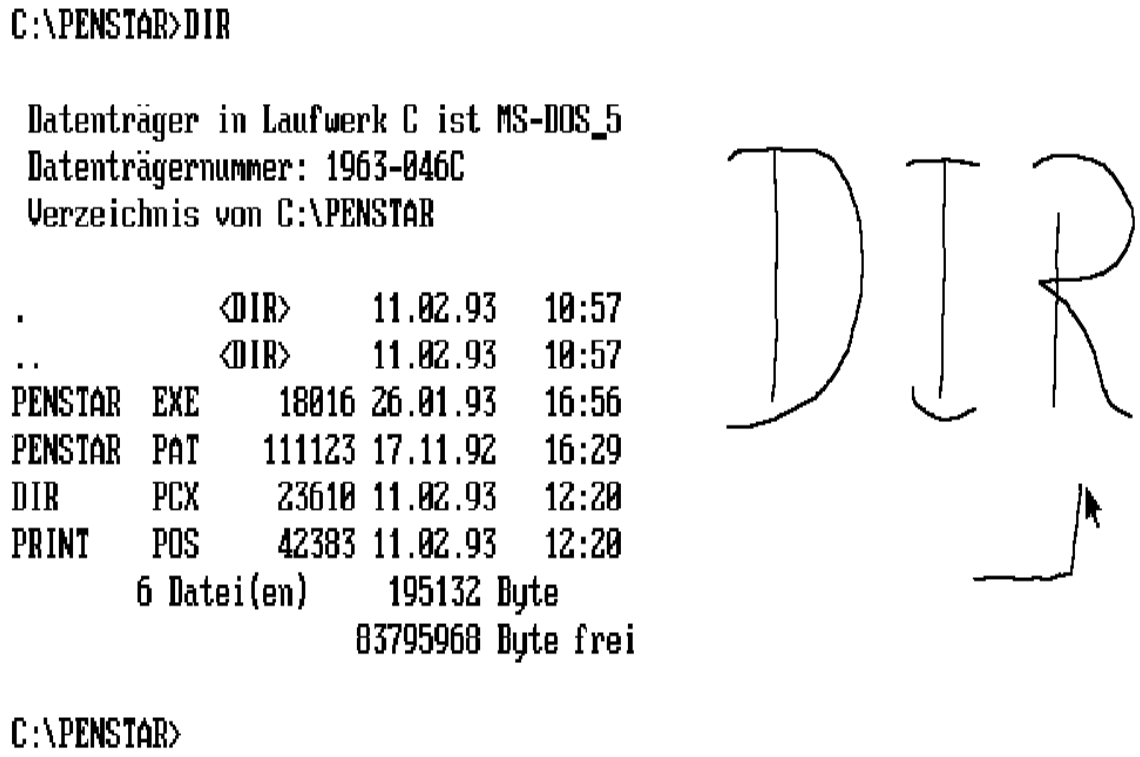

figure 7

4.1.3 The PEN-COMMANDER for DOS

The PEN-COMMANDER is a special application for the INFOS 386-SX notepad computer. It has been designed to replace the keyboard in standard keyboard-oriented DOS applications by a pen. All handprinted data is converted into keyboard-like data. Therefore the PEN-COMMANDER emulates the standard keyboard by recognizing and converting handprinted pen input data into keyboard data. The operating-system can not distinguish input data provided by the real hardware keyboard or by the emulation. Standard DOS applications can be used with the notepad computer even if they are not provided with a pen-based user interface. Figure 7 shows a handprinted DOS DIR command. It has been converted into ASCII and sent to the operating-system. The operating system shows the command at the DOS-prompt. The "⌋" symbol is a guesture used as RETURN key. After drawing the "⌋" the command is executed and the directory shows up. The PEN-COMMANDER is restricted to applications getting keyboard data via DOS interrupt 21hex. Applications using their own keyboard driver are not suitable since they are observing the physical keyboard hardware directly. The PEN-COMMANDER cannot be used with mouse oriented applications since it behaves logically like a keyboard and not like a digitizing device. Therefore the PEN-COMMANDER shuts down and passes control to the mouse-driver if an application request mouse specific functions.

4.1.4 Scientific application

A scientific application is the implementation on the AM^3 Processor. The recognizer is used to evaluate guestures of a car-driver, drawn on a touch-panel with a fingertip to control certain

functions of a car and a car-co-pilot. An important scientific result was that the CAM module of AQUIRE can be accelerated using the associative type of parallelism. The linear run-time behaviour on a sequential processor of the module became approximately logarithmic on the associative processor. The logarithmic runtime-behaviour is due to the logarithmic access-time of the emulated memories of the AM^3 processor.

5. Future work

We expect that the base-method of PATTMA is suitable to support a speech-recognizer. The opportunity of associative parallelism supports this investigation since online-speech-recognition requires fast retrieval mechanisms for large databases. A special associative memory derived from the experiances with associatively supported pattern recognition is currently under developement to compute the MHD classificator in constant time.

References

[DAR90] M. Darianian, Ch. Schönfeld, K. Waldschmidt, Ein Assoziativspeicherfeld hoher Kapazizät im Bit-Slice-Prozessor AM^3 , Tagungsband der 11. GI/ITG-Fachtagung: Architektur von Rechensystemen, VDE 1990
[KOH87] T. Kohonen, Content Addressable Memories, Springer, 1987
[RHY86] J.R.Rhyne, Dialogue Management for gestural interfaces. Computer Graphics, 21 (2), Workshop on User Interface Software, April 1987
[SCH89] M. Schulz et al. An Associative Microprogrammable Bit-Slice-Processor for Sensor Control, Proceedings of the 3rd CompEuro, Hamburg 1989
[SCH92] M.Schulz, An Object-Oriented interface in C++ to an associative processor, Proceedings of the 6th CompEuro, The Hague 1992
[WAL92] K. Waldschmidt, M. Schulz, Der assoziative Universalprozessor AM3: Architektur, Befehlssatz und objektorientiertes Programmierinterface, Tagungsband der 12. GI/ITG-Fachtagung: Architektur von Rechensystemen, Kiel 1992

Special thanks to Fred Schuchard who collected some MBs of handprinted data for testing and verification, to Reiner Heinrich who provided us with the INFOS 386SX Notepad Computer and to Ronald Moore.

AN OBJECT-ORIENTED PEN-BASED RECOGNIZER FOR HANDPRINTED CHARACTERS

B. Klauer K. Waldschmidt
J.W.Goethe - Universität Frankfurt
Professur für Technische Informatik
Robert Mayer Straße 11 - 15
D - 6000 Frankfurt am Main

++ 49 69 798 2121
Fax ++49 69 798 2351
email klauer@ti.informatik.uni-frankfurt.de

R. Heinrich
INFOS GmbH

Ludwigstr. 78
D - 6050 Offenbach

++ 49 69 228131 0
Fax ++49 69 228131 31

KEYWORDS

Handprinted character recognition, Pen computing, Associative memories, Fault tolerant computing, Pattern classification

ABSTRACT

This paper introduces the object oriented character recognition engine AQUIRE which has originally been designed as general pattern-classifier under the academic aspects of object-orientation and parallellism together with low space and time complexity. When industry raised interest in using AQUIRE to provide pen-based computers and applications with a powerful character-recognition engine, AQUIRE has been revised due to commercial requirements. Together with a brief discussion of some specific problems of pen-computing the paper contains the theoretical background for the recognition mechanism and a description of the methods used to perform a high quality and high speed recognition-process with a minimum of executable code (~18KB on IBM-AT kompatible machines under MS/DOS) an a small database (~50KB for a complete character-set containing all upper-case, lower-case characters together with punctuation symbols and numbers). The paper contains a brief overview on the implementations of the proposed method on an INFOS 386 SX pen-computer and on the associative processor AM^3 (Associative Microprogrammable Multipurpose Monoprocessor) developed at the department for technical computer sciences of J.W.Goethe-University as contribution to the PROMETHEUS project. Within PROMETHEUS AQUIRE has successfully been used as recognizer for finger-printed guestures on a touch-panel to provide a comfortable user (driver) - interface to a car-co-pilot.

IN CASE OF ACCEPTANCE...

...this paper will be presented at the Euro - ARCH ’93 by the author.
... a camera-ready version of the paper will be provided by July 2. 1993.