IDIAP COMMUNICATION

Handwriting Recognition Demo

Haiyan Wang
Alessandro Vinciarelli
Frank Formaz

IDIAP-COM-02-02

MARCH 2002

Dalle Molle Institute for Perceptual Artificial Intelligence • P.O.Box 592 • Martigny • Valais • Switzerland

phone +41 – 27 – 721 77 11
fax +41 – 27 – 721 77 12
e-mail secretariat@idiap.ch
internet http://www.idiap.ch

1. INTRODUCTION

Handwriting is very natural to interact with computers. Nearly everyone learns to use a pen at an early age, and writes every day of his life. During the last twenty years, handwriting recognition has become a very important research area, which is attracting more and more scientists. In fact, the advances in the field of data acquisition technology and the promising results of the research, nowadays make possible the development of commercial systems or potential application in many areas such as office automation, bank check processing, recognition of postal addresses and ZIP Codes, signature verification, and document and text recognition.

This *Handwriting Recognition (HR)* Demo aims at presenting a state-of-the-art off-line handwriting recognition system developed at IDIAP. Off-line handwriting recognition is the automatic transcription by computer of handwriting, where only the image of the handwriting is available. The great variability of handwriting styles and the fact that the letters are connected are the major difficulties of the problem. To read cursive handwritten words is a difficult task even for humans. This makes us ask ourselves the question: If people sometimes can't read their own handwriting, with which they are quite familiar, what chance does a computer have? Fortunately, there are powerful tools that can be used that are easily implemental on a computer nowadays.

This demo shows the prototype of an office management system that provides pen-driven access to IDIAP people information, such as names, telephone numbers and nationalities through a digital tablet or a standard mouse. From a future perspective, speech recognition technology will be well integrated, thus make this application could be enabled by speech or pen.

2. ABOUT THE HR DEMO

2.1 HR Demo description

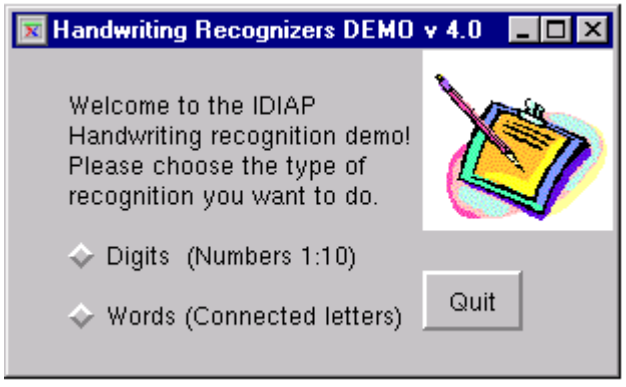


Figure 1 HR Application Welcome message

The demo starts by showing a popup window. *(See Figure 1)* It displays a welcome message and holds two *Radio buttons* that allow you to choose between cursive or digits recognition.

The button "Digits" starts the "Handwriting Digits Recognition" dialog window shown in *Figure 2*. The white area on the top of the window is the input area. You can draw directly the number with the electronic pen or mouse. The width of the pen can be modified by right clicking on the draw zone, and by selecting a new size. The recognition is executed by clicking "Recognize" button.

Recognition results are displayed in the white field at the right-bottom. The confidence levels and the absolute levels plot are shown in the middle. The "confidence" levels measure the distance between the scores of the two best ranking hypothesis, the "absolute" levels mean the score of the best ranking class.

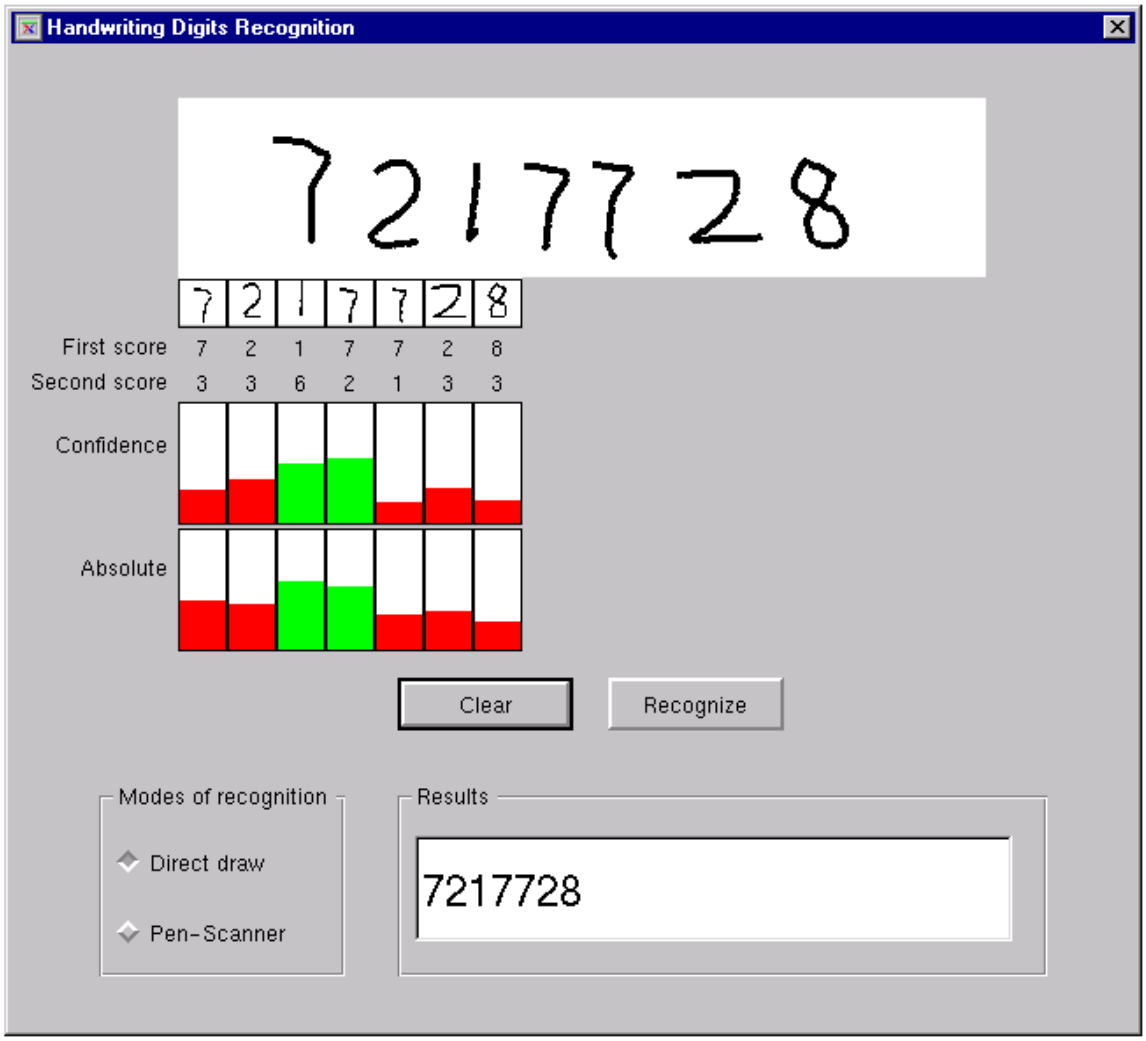


Figure 2 Digits Handwriting Recognition Application Outlooks

The button *"Cursive" (see Figure 1)* starts the "Handwriting Cursive Recognition" dialog window, the same as in "Handwriting Digits Recognition" dialog. The input area is on the top. You can use *Wacom* electronic pen or mouse to write words in the white blank zone. The system recognizes single word. Before performing recognition, you must choose the content of recognition from the IDIAP people's family names or countries *CheckBox*. Click the "RECO" Button, the recognition result appears in the "Cursive Recognition Results " field. *(See Figure 3)* To get prompt information of the HR Cursive application, click the "Help" button. Click the *"Exit" Button*, you will quit the HR Cursive application.

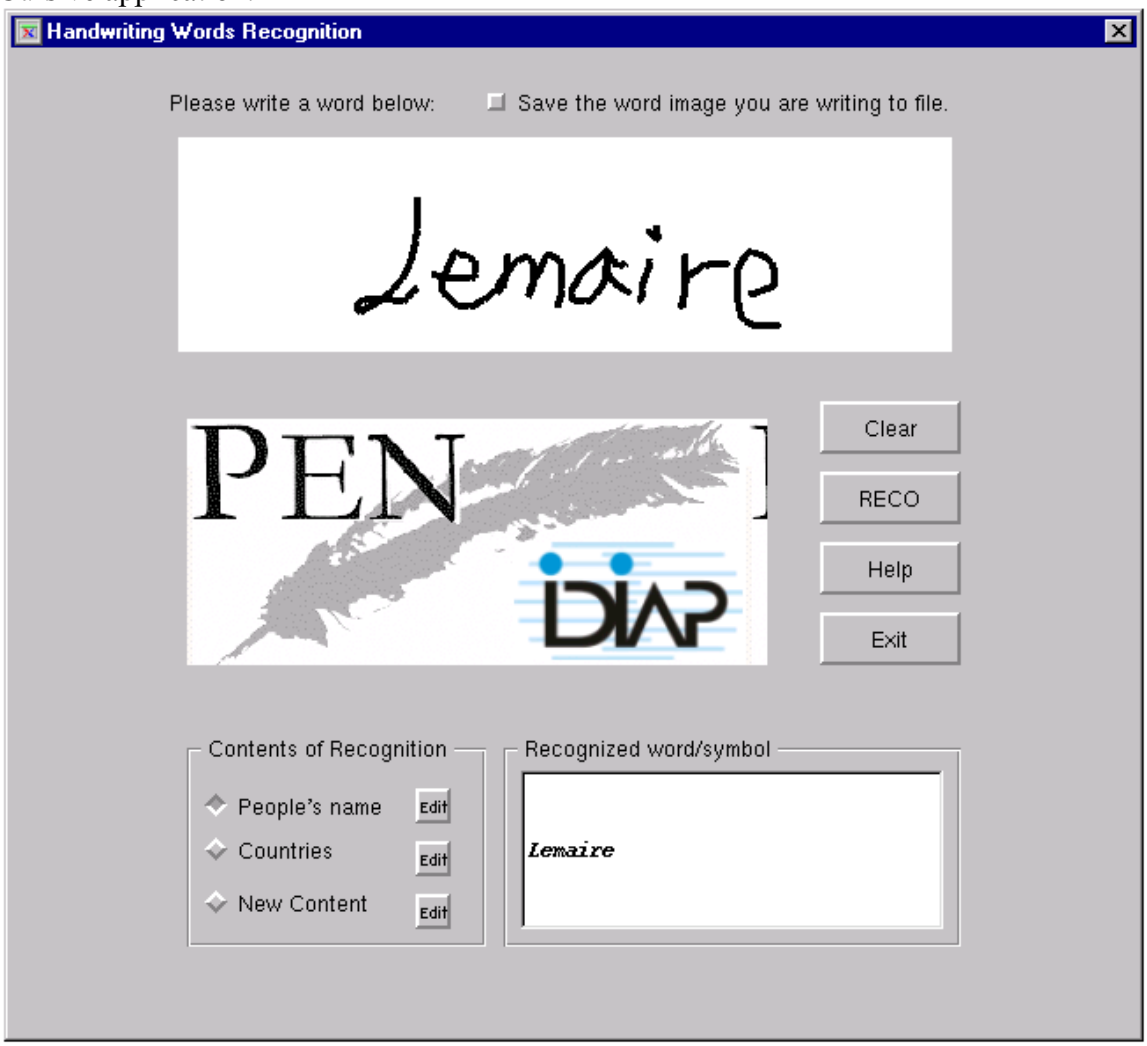


Figure 3 Cursive Handwriting Recognition Application Outlooks

If you are not family with this IDIAP people's information, press the small square *Buttons "Edit"* to get the names or countries list. Hereafter is an example of people's name list. From this modify window, the user can add a new name to the list, press *"Ok"* to quit the small window and update dictionary and syntax files, afterwards you can write the new name and recognize it. For example, the user writes a new name on the *"Modify name"* Box, click *"Add"* to add new name into the list, click *"Help"* to get help information, click *"Cancel"* to ignore the present operation and quit the small window.

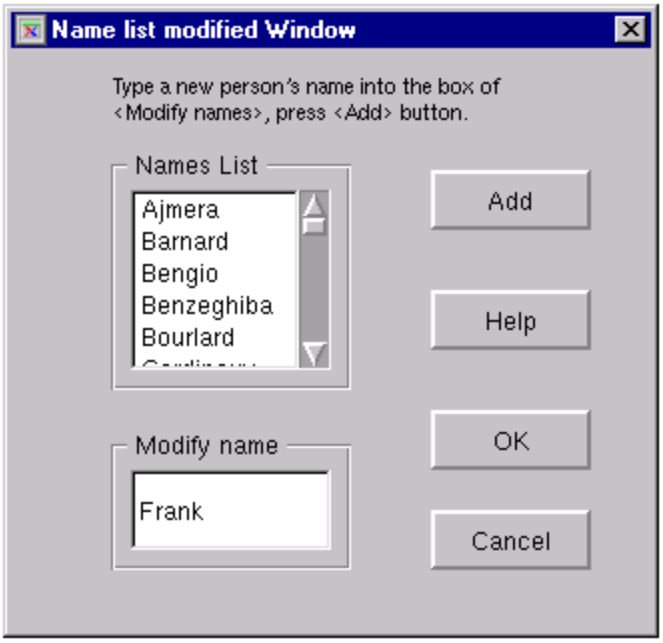


Figure 4 Name list modified window Outlooks

The following is another window for user to generate a new contents list. In this window you can edit new word which is going to be recognized, only the words in the dictionary can be recognized. To add a word to the dictionary, type a person's name into the <Edit new word> box, and then press <Add> button. To delete a word, select from the list, and press <Delete>. When you have finished editing the dictionary. Click <OK> to exit, meantime the new dictionary and syntax file are generated for recognition, press <Cancel> to ignore the operations and exit.

Figure 5 New lists edit window Outlooks

2.2 Development Environment

Software – *"Linux"*, "*Windows NT*", C/C++ language under *"KDevelop"* and "*Microsoft Visual Studio*", *Qt* library

Hardware – PC, Wacom *"Graphire2"* tablets

The *Wacom Graphire2* is a high-quality tablet that doesn't require batteries and connects to your machine's existing USB port. In essence, a pencil becomes your both mouse and keyboard. This one was chosen since its support on both windows and Linux environment.

2.3 HR Technology support

The recognition system used for the experiments is based on a sliding window approach. The image of the word is first normalized: slope (the angle between the horizontal direction and the direction of the line the word is aligned on) and slant (the angle between the vertical direction and the direction of the strokes supposed to be vertical in an ideal model of handwriting) are removed. Once the word is normalized, a window shifts column by column from left to right and, at each position, a feature vector is extracted.
The feature extraction process consists of partitioning the content of the window into 16 cells regularly arranged in a 4x4 grid and containing each one n_i foreground pixels.
The feature vector is obtained as follows:

$$\overline{f} = \left(\frac{n_1}{N}, \frac{n_2}{N}, \ldots, \frac{n_{16}}{N} \right)$$

where N is the total number of foreground pixels in the window. The sequence of vectors corresponding to each position of the window is modeled with continuous density HMMs having mixtures of Gaussians as emission probability. A model is built for each letter and, for each entry in a list of allowed transcriptions (called lexicon), a word model is obtained by concatenating the single letter models. This makes the system more flexible with respect to a change of lexicon since it is not necessary to have samples of the words to be recognized in the training set. For simplicity, the number of states and mixture Gaussians is the same for every letter model.
The training of the models is performed with the Baum-Welch algorithm and it is embedded. This means that the training algorithm is not applied to the single letters, but to the words. In this way, no segmentation into letters of the training data is necessary

and the single characters are modeled as parts of a word (a realistic condition in the cursive handwriting). The recognition is performed with the Viterbi algorithm.
This gives an estimate of the highest value of the likelihood that can be obtained by following a unique state sequence in the word model with the vectors extracted from the handwritten data.
The entry of the lexicon corresponding to the model giving the highest value of likelihood is selected as transcription of the handwritten word.

3. DEVELOPMENT AND IMPLEMENTATION

This demo is required to run on both Linux and Windows operating systems. To this end, we developed two versions for each module including recognition engine, GUI tools, and databases.

3.1 Recognition engine

The preprocessing and feature extraction modules are developed as isolated modules in C code on Unix and the HMM based training and recognition are performed by HTK software.

The Hidden Markov Model Toolkit (HTK) is a portable toolkit for building and manipulating hidden Markov models. HTK was originally developed at the Speech Vision and Robotics Group of the Cambridge University Engineering Department (CUED) where it has been used to build CUED's large vocabulary speech recognition systems. The software supports HMMs using both Gaussian mixtures and discrete distributions and can be used to build complex HMM systems. HTK was installed on windows and Linux, so the demo application could use HTK commands to process image features on both platforms.

The HR demo is a GUI application, which requires C++ definitions and functions. All the technological source code has been revised from C to C++ in the present HR recognizer. We transported the preprocessing and feature extraction modules into C++ on Linux and Windows individually. Besides, the code of Preprocessing and Feature-extract were bound together for optimization.

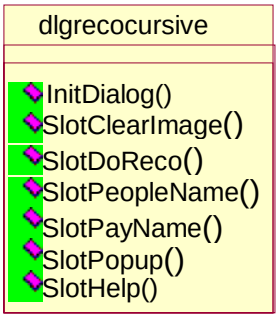
3.2 GUI Implementation

3.2.1 On Linux

The Graphical User Interface (GUI) of HR Demo was implemented using **KDE on the basis of Qt 1.X** library. KDE is a powerful Open Source graphical desktop environment. Qt is a fully object-oriented, easily extensible C++ application framework that enables rapid building of state-of-the-art GUI applications, which enables building professional, efficient, portable and maintainable GUI applications quickly and easily. Qt implements **standards GUI objects**, from the most basic button to a complex hierarchy view; It has his own implementation of **signal-slots communication mechanism.** It allows QObject

to send events (with parameters) at any time to others QObject. The Qt library simply replaces Motif, MFC, etc. Qt-based applications run at least as fast as applications written with those toolkits. The memory footprint is also the same.

The following example explains a class for Cursive Handwriting Recognition Dialog application on Linux.



The methods start with "Slot…" is developed as responses of the operations of GUI objects such as buttons and check boxes in our dialog windows. In Qt, any operation of a GUI object actives a corresponding signal, which should be processed by a specified method called slot. To connect a signal and a slot, you simply use QObject::connect(). This method asks four inputs: the object that sends out the signal, the signal that it should connect the slot to, the object that will receive the signal, and the slot that will be connected to the signal. For instance, if click the *"Clear" Button* object, the *"click" signal* sent by the *"Clear" Button* object should connect to the "SlotClearImage()" slot of the "dlgrecocursive" object.

3.2.2 On Windows

To implement HR Demo on Windows, we used "*Visual C*$^{++}$ " based on Microsoft Foundation Classes (MFC) library. MFC comprises excellent and copious graphical objects classes, which is greatly simplifies the accomplishment of our tasks.

The current HR demo on Windows is derived from an original Digital handwriting recognition demo developed by Eric Grand. The details can be found in *ref [1]*. In this demo application, we add a *"word recognition"* button into the main interface, and then connect with Cursive Handwriting Recognition software to function this button. When the user wrote a word, and click *"word recognition" button,* the recognition result would appear on the work frame below. Additionally the user can change the width of the pen with a right click on the drawing zone.

3.3 Building Databases

The GUI of this demo is not only for illustrating the functions of handwriting recognition but for building training databases. A database consists of a set of images of handwritten digital or words, their transcriptions, a dictionary and a syntax file.

3.3.1 Image File Generator

We developed an Image File Generator to collect images for training. *(See Figure 4)* Using this interface, it is possible to write cursive words in the white area on the top and input its transcription in the field below. The word id is used to distinguish different instances of the same word. The id of database specifies the name of targeting database. Image File Generator can save each cursive input as an image file with the labels of this word within the file name.

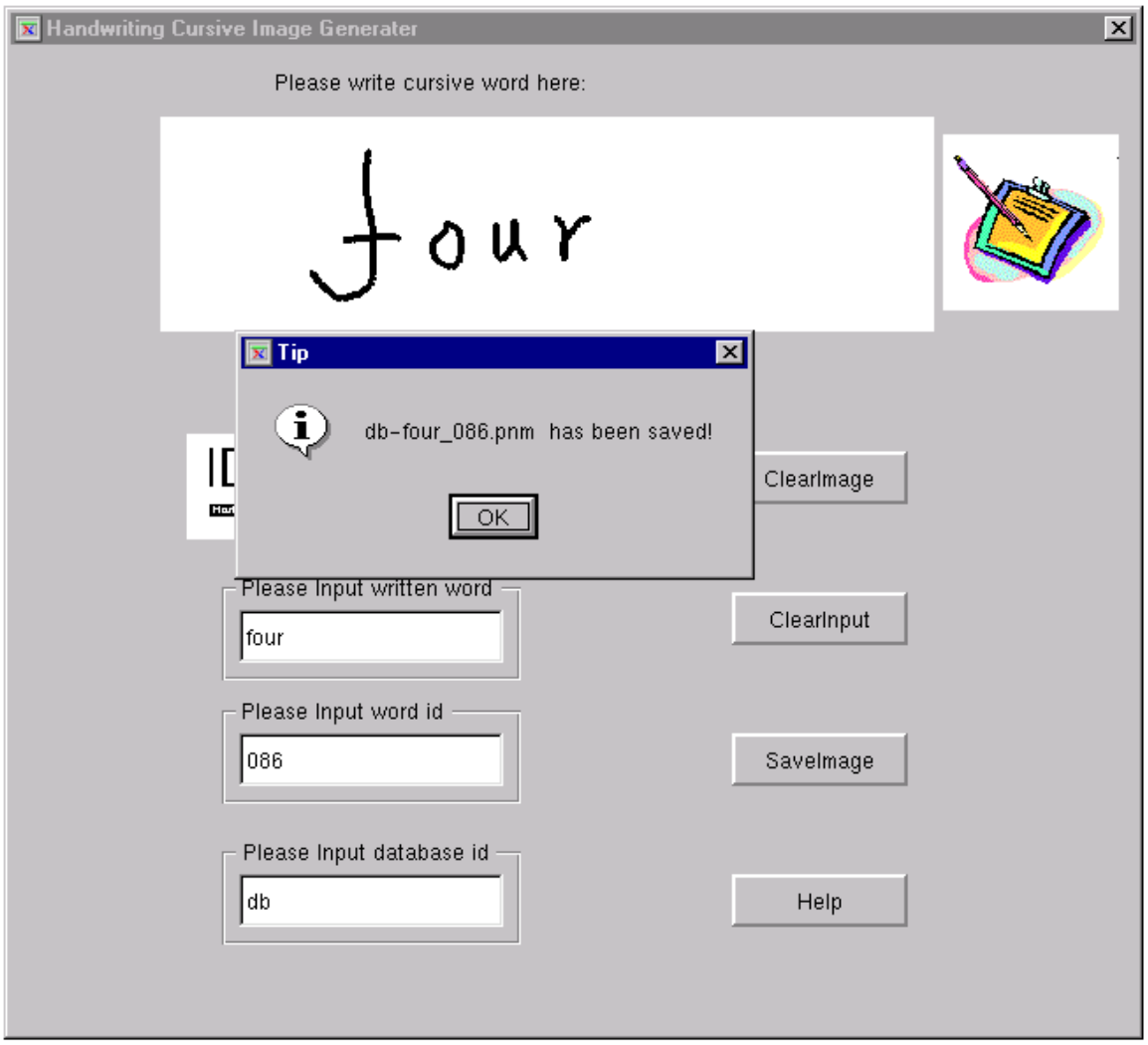


Figure 6 HR Image Generators

For instance, if the user writes the word “four” with the id “086”, and database is “db”, The image file “db-four_086.pnm” will be generated after pushing the *“SaveImage” Button*.

QImage class, one of the image mechanisms of Qt provides you actually the methods, which get the image data correctly in and out. **QImage** objects encapsulate the pixel data; also have a width, a height, and a depth.

Another essential point is to determine the image file format that you use within the HR Demo. Image file formats are the standards by which pictorial information is stored in and retrieved from files. The image file we used for HR Demo is a PNM (Portable Anymap) format image. PNM Set of image formats of increasing complexity - from bits to gray levels, to pixmaps of various sorts.

Hereafter is an image file shown by standard image display program--*“xview”*.

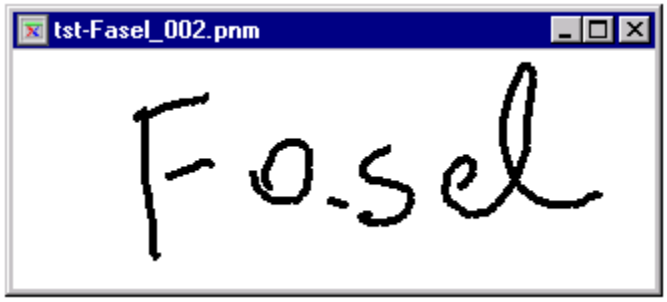


Figure 7 an example of HR Image file

Using the image file generator, we built the databases of IDIAP people’s names and nationalities that consist of over 600 examples written by 40 people.

3.3.2 Dictionary and Syntax file

The HR Demo is a content-depende recognition system. In this Demo, we use dictionary and syntax files to specify the recognition content. The goal is to recognize IDIAP people’s name and nationalities. If we would like to recognize other content, or add more words into the dictionary, hereafter we describe an example of how to create dictionary and syntax files for countries.

- Input the country names and save it – *“countryList.dat”*
- Run ~vincia/code/projects/createSyntax, to change the format of *“countryList.dat”* to suit the HTK Toolkit.
  >createSyntax countryList.dat countrySyntax

- HParse countrySyntax countryLat (Hparse is a HTK Command)

- Run ~vincia/code/projects/lexicon/CreateLexicon, to get dictionary.
  >CreateLexicon countryList.dat

So the *"dictionary"* and *"countryLat"* are individually new created **Dictionary** and **Syntax files**.
In the HR demo, the dictionary file consists of 43 names and 29 countries

4. CONCLUSION AND FURTURE WORK

At present, the main functions of the handwriting recognition demo have been successfully developed on Linux and Windows. You can write cursive words or digits and get recognition results through a graphic interface. Meanwhile, each word image can be saved as training material for the future versions of the demos.

In the future development and improvement, we could focus on to the following points:

- Apply database property, for instance: display the people's information (group name, telephone num, e-mail address etc) when this people's name is recognized out.
- Improve the accuracy of the recognizer by applying more advanced technologies.
- Integrate with the *InfoVOX* system, to combine speech and handwriting recognition.

5. REFERENCES

[1] *Eric Grand , "Handwritten Digits Recognition",* IDIAP-RR 00-07, 2000

[2] *Alessandro Vinciarelli and Juergen Luettin*, ***"****Off-Line Cursive Script Recognition Based on Continuous Density HMM",* in "Proceedings of 7th International Workshop on Frontiers in Handwriting Recognition", 2000.

[3] *"Programming with **Qt**"*

[4] *Frédéric Bressoud and Haiyan Wang*, *"Personal Voice Dialing over PC",* IDIAP-Com 00-05, 2000

[5] *"Programming with Windows 98/NT"*

[6] http://www.trolltech.com/

[7] http://www.kdevelop.org/index.html

[8] http://www.research.ibm.com/handwriting/

[9] http://www.wacom.com/graphire/index.cfm

[10] MSDN Library Visual Studio 6.0

[11] http://www.linuxgraphic.org/dcforum/DCForumID1/93.html
[12] http://sem.best.vwh.net/linux/wacom/