LipiTk: A Generic Toolkit for Online Handwriting Recognition

Sriganesh Madhvanath, Deepu Vijayasenan, Thanigai Murugan Kadiresan

LipiTk: A Generic Toolkit for Online Handwriting Recognition

Sriganesh Madhvanath *Deepu Vijayasenan* *Thanigai Murugan Kadiresan*
Hewlett-Packard Laboratories
Bangalore, India
{srig, deepuv}@hp.com

Abstract

This paper describes Lipi Toolkit (LipiTk) - a generic toolkit whose aim is to facilitate development of online handwriting recognition engines for new scripts, and simplify integration of the resulting engines into real-world application contexts. The toolkit provides robust implementations of tools, algorithms, scripts and sample code necessary to support the activities of handwriting data collection and annotation, training and evaluation of recognizers, packaging of engines and their integration into pen-based applications. The toolkit is designed to be extended with new tools and algorithms to meet the requirements of specific scripts and applications. The toolkit attempts to satisfy the requirements of a diverse set of users, such as researchers, commercial technology providers, do-it-yourself enthusiasts and application developers. In this paper we describe the first version of the toolkit which focuses on isolated online handwritten shape and character recognition.

Keywords: Online Handwriting Recognition, Shape Recognition, Toolkit, Linguistic Resources, API

1. Introduction

There are still large parts of the world characterized by the extensive use of paper and handwriting in all facets of society, and poor penetration of traditional PCs and keyboards. In India for instance, only 12 in 1000 persons have PCs, as compared to 785 in the US, and over 500 in most Western European countries[1]. In this setting, products and solutions with pen and/or paper-based interfaces may play an important role in making the benefits of Information Technology more pervasive. HWR is an important technology in order to research appropriate user interfaces, and create innovative products, solutions and services for these markets. For example, the Gesture Keyboard [14] is an experimental desktop peripheral that uses recognition of handwritten gestures for Indic text input, and form-filling solutions in local languages also appear to have potential [15].

[1] 2004 figures, Source: Gartner Report

Unfortunately for many of the languages in these parts of the world - such as the Indic languages - no commercial handwriting recognition technology exists. The central problem being addressed in this paper is: how can we simplify the creation of HWR technology for a new script, and how can we simplify its integration into real-world applications? This problem has been addressed by sister language technology communities working on speech recognition, speech synthesis, and machine translation through the creation of toolkits comprised of tools and algorithms that can be used to create language technology for a new language [1,2]. The issue of integration has been addressed by the creation of standard interfaces/protocols such as MRCP for speech recognition engines [3]. However, to the best of our knowledge, no generic toolkit or standards exist for online handwriting recognition.

1.1. The Lipi Toolkit

There are many different challenges involved in developing a "generic" toolkit for online handwriting recognition. The first is that the toolkit should provide good enough generic components to perform reasonably well on simple as well as complex scripts, while providing the flexibility to tune, extend or even replace them with more suitable components, to meet the challenge of a particular script or application. A second challenge is to balance the needs of different classes of potential users (Figure 1). For researchers in handwriting recognition, the toolkit should serve as a

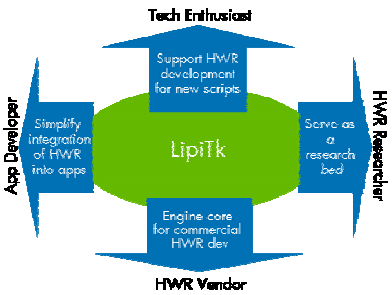

Figure 1. Design goals of Lipi Toolkit

research bed to experiment with new algorithms. For a certain class of do-it-yourself enthusiasts, it should allow the creation of engines for new shapes and

scripts out of the box, without requiring much knowledge of the algorithms. For a potential vendor interested in building commercial HWR engines, it should support the building of robust engines for new scripts. Finally, for the application developer, it should allow easy integration of engines built using the toolkit into any pen-based application.

The Lipi Toolkit ("lipi" being Sanskrit for "script") is our effort to create a generic toolkit whose components can be used to build an online HWR engine for a new script, while addressing the challenges described. There are three distinct stages in the lifecycle of the toolkit. In the first stage, standard tools and algorithms are packaged into a "downloadable toolkit". In the second stage, an intermediate user (such as a researcher or vendor) downloads the toolkit source and binaries, experiments with and potentially modifies the algorithms or adds new ones in order to build a custom engine for a new script and/or application context. In the third stage, an end-user (such as the application developer) integrates the built engine into a pen-based application. The three stages can clearly be separated temporally and spatially and involve very different sets of users. While designing the toolkit, our emphasis is primarily in the first stage, while ensuring that the toolkit supports the second and third stages in the hands of the respective user groups.

The first version of the toolkit (LipiTk 1.0) supports the recognition of isolated handwritten shapes and characters, as well as boxed words, captured as digital ink. The components of the toolkit make it possible to carry out all the steps involved in building a character recognition engine. These components are represented in Figure 2 and described in Section 3. But first we take a look at some of the salient features of the toolkit in the light of previous efforts.

2. LipiTk: Salient Features

While toolkits such as Sphinx from Carnegie Mellon University [1] and Festival from University of Edinburgh [2] exist for problems such as Automatic Speech Recognition and Speech Synthesis respectively, we believe LipiTk to be one of the first to address the problem of online HWR. Open source implementations of online gesture and character recognition such as Rosetta [4], XStroke [5] and WayV [6] are not primarily intended for experimentation with HWR algorithms, which is one of the (many) core goals of LipiTk.

LipiTk is designed to support a data-driven methodology for the creation of recognition engines. This implies tools for handwriting data collection and annotation, standard script-independent algorithms for preprocessing and feature extraction, algorithms for training and pattern classification, and tools for subsequent evaluation and error analysis.

While many handwriting recognition algorithms are script specific, LipiTk is intended to provide robust implementations of generic features and classifiers that are expected to perform reasonably well on any given set of symbols by learning the statistical shape properties of that set. This allows a reasonably performing recognition engine to be built with a minimum of effort.

One of the characteristics of handwriting recognition is that no single approach or set of features/classification algorithms is known to work optimally for all scripts. Also, the nature and quality of the input (digital ink) tends to vary widely with the capture device. LipiTk has been designed to accommodate different tools and algorithms specific to the device, script and/or application.

LipiTk 1.0 uses open standards such as UNIPEN [17] for the representation of digital ink and its annotation, facilitating the creation of shareable linguistic resources within the community. Future versions may use W3C InkML [18] and UPX [19] for digital ink and annotation respectively.

There is a focus on robust and efficient implementation of algorithms in LipiTk, in order to facilitate the integration of engines created using the toolkit into real-world applications.

LipiTk also specifies standard shape and word recognition interfaces for recognition engines. All engines built using the toolkit hence expose a standard interface, simplifying their integration into pen-based applications. The shape recognition API is less ambitious in some aspects than previous efforts [7], but is distinct from them in that it includes training and online adaptation of shape recognizers.

3. LipiTk Architectural Design

The Lipi Toolkit was intended to support both Windows and Linux platforms, hence its design and implementation considers portability related issues. Most of the algorithms and tools are implemented using C++ & STL. Only ANSI functions are used for portability. Some of the utilities and scripts are written in Perl. The major components of the toolkit are described below.

3.1. Generic Class and Utilities Library

The generic class library includes classes to store and manipulate ink traces, such as *Trace* and *TraceGroup*, and classes to store device and screen

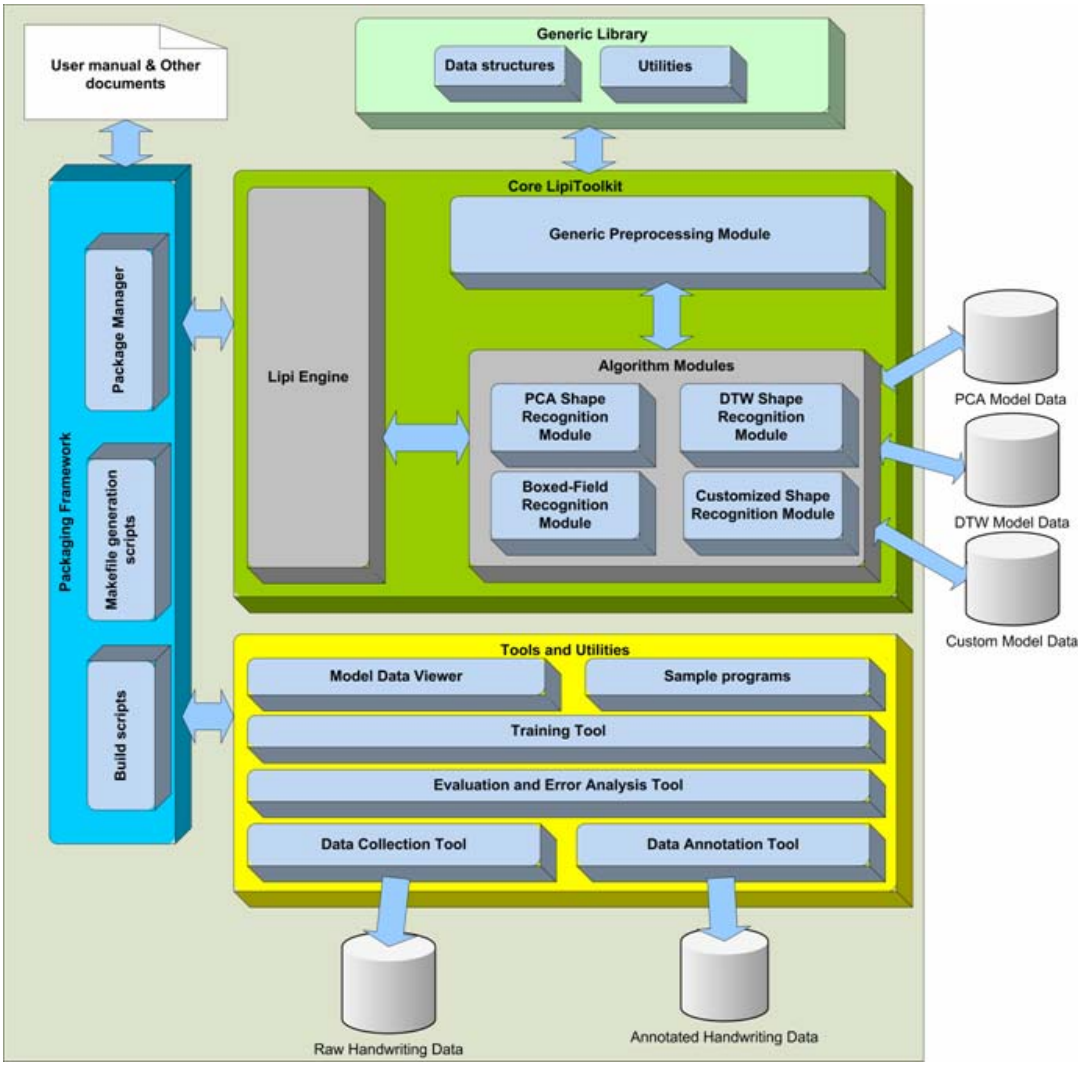

Figure 2. Lipi Toolkit Architecture and Components

context. These classes are shared by different algorithm and tool implementations. The design of these classes reflects a tradeoff between a conceptually intuitive and object-oriented data model, and efficient access to frequently accessed attributes, such as X and Y channels in the case of ink traces.

The utilities library provides utility functions to read and write LipiTk configuration files, read and write UNIPEN data files, and so on.

3.2. HWR algorithms

LipiTk 1.0 provides implementations of common preprocessing operations, common shape recognition algorithms, as well as a boxed field recognizer which iteratively calls one of the shape recognizers to interpret a boxed field of ink input. The preprocessing module, as well as the shape recognition and boxed field recognition modules are implemented as separate dynamic link libraries that can be loaded at runtime.

3.2.1. Generic Preprocessing

The generic preprocessing module provides implementations of commonly used shape/character preprocessing operations such as moving-average smoothing, size normalization, dehooking, and equidistant resampling. All of the operations have configuration options that can be varied using corresponding properties captured in a configuration file.

3.2.2. Shape Recognition

The two shape recognition algorithms bundled with LipiTk 1.0 are Subspace-based classification (PCA), and Nearest-Neighbor classification based on Dynamic Time Warping (DTW).

In subspace classification, each shape class is represented by a set of Principal Components computed from a fixed length representation of the online shape, obtained after size normalization and equidistant resampling [12]. The training method provided computes the Principal Components from

the training data, and stores them in a standard binary format.

The DTW implementation uses the same fixed length representation as the subspace classifier, together with a Nearest Neighbor classifier [10,11]. The training method exposed by the shape recognizer provides a choice of different prototype selection algorithms for prototype reduction.

Both shape recognizers expose a standard shape recognition API which allows the recognizer to be loaded, trained, and invoked on a *TraceGroup* (group of traces) corresponding to a single or multi-stroke shape or character.

In either case, important parameters (such as the number of Principal Components) as well as the sequence of preprocessing operations are externally configurable using configuration files.

3.2.3. Boxed field recognition

As mentioned earlier, the boxed field recognizer is useful for recognizing a boxed field of shapes, and in turn invokes a trained shape recognizer on each of the boxes, and uses a simple trellis for decoding the best strings based on the cumulative shape recognition confidences.

Significantly, the boxed field recognizer exposes a generic word recognition API, allowing the possibility of plugging in a connected word recognizer in the future in a backward-compatible manner.

3.3. Tools and Utilities

LipiTk 1.0 provides a number of tools and utilities to support the tasks of handwriting data collection, data annotation, and the training and evaluation of shape recognizers.

3.3.1. Data Collection and Annotation Tools

Collection and annotation of handwriting data is an important activity in the data-driven methodology for creating recognition engines. LipiTk 1.0 includes a generic TabletPC-based data collection tool capable of collecting isolated symbols, characters or words from writers [13]. We hope to include in subsequent releases, tools based on Digital Pen and Paper or PDAs, as well as tools running on Linux.

The annotation tool supports the tagging of digital ink with labels corresponding to ground truth, writing style etc. at different levels of an appropriate hierarchy of annotation. The tool included with LipiTk 1.0 is a generic tool written in C++/Qt with the ability to annotate entire documents of handwritten text, and supports plug-ins for segmentation and recognition to partially automate the annotation process [22].

3.3.2. Evaluation Tool

The Evaluation Tool computes statistics related to classification accuracy and performance of the built engine on the test data, and allows visualization of the results in ways that facilitate analysis of the errors. LipiTk 1.0 includes a basic evaluation tool written in Perl which renders top N accuracies and confusion matrices in the form of HTML pages.

3.3.3. Utilities

In addition to the above tools, LipiTk 1.0 also includes a number of scripts to facilitate tasks such as extraction of isolated character data from the annotated data (which may be words), and splitting the annotated data randomly into training and test sets.

3.4. Packaging framework

LipiTk provides build scripts to support the creation of specific engines from the source code. These scripts interpret project configuration files and build the necessary source code into libraries and binaries, using a hierarchy of static module-specific *Makefiles*.

LipiTk also provides scripts for packaging the built engine(s) for deployment, and integration into a pen-based application. The components of the package are fully user-configurable, and the packaging script creates a self-extract package file (or gzipped tar file in the case of Linux) that contains all the components selected for packaging by the user.

Finally, LipiTk provides sample code to assist the application developer in integrating an engine created using LipiTk into his or her application.

3.5. Lipi Engine

The Lipi Engine is the run-time component of engines created using the Lipi Toolkit. It is responsible for loading one or more shape/word recognition modules as specified in its configuration file, routing requests for recognition from the user application to the appropriate modules, and returning recognition results to the application.

4. Working with the Toolkit

In this section, we will take a brief look at the use of the toolkit to develop a recognizer for a set of shapes, for example, the 10 Indo-Arabic numerals. The first step in the process is the creation of a new shape recognition project, which we will call `numrec`. The associated project configuration file identifies the project as a shape recognition project (as opposed to a word recognition project), and the number of shape classes to be recognized as 10. Within the `numrec` project, one or more profiles may be defined. Each profile contains configuration settings for the recognizer corresponding to a particular user-defined mode of operation or “flavor” of the recognizer. Profiles may be used to distinguish settings for writer-independent recognition vs. specific writers, specific training data (e.g. US versus UK writers), specific recognition algorithms (PCA vs. DTW vs. Custom), specific preprocessing settings,

and so on – at the discretion of the researcher/developer. The profile also stores the results of training the shape recognizer (recognizer-specific model data) using the specific set of configuration settings.

The build scripts provided with the toolkit allow the building of a specific project and profile, and use the associated settings to determine which recognizers to build. For instance, if the `numrec` project's `default` profile specified PCA as the recognizer, and PCA's configuration file in turn specified a sequence of preprocessing operations, building the project and profile would cause the generic preprocessing as well as the PCA code to be built into dynamic link libraries. Building the project also builds a command line utility `runshaperec` that links with these libraries and supports training and testing of the shape recognizer on labeled numeral data.

The data collection tool provided with the toolkit may be used to collect samples of numerals from different writers, and organize them in the form of an annotated UNIPEN dataset. Alternatively an existing dataset may be used. A utility script supports the creation of training and test lists from the dataset using regular expressions to match file names.

The `runshaperec` utility may now be used to train the shape recognizer by providing the training list of samples as input. This causes subspaces for the 10 classes to be computed as stored as binary model data as part the `default` profile.

The same `runshaperec` utility may be used in evaluation mode to classify the test samples in the test list. The result file produced may then be provided as input to the evaluation tool to compute recognition accuracy and the confusion matrix.

Once the cycle of training and testing is completed, the packaging script may be used to package the entire `numrec` project and `default` profile into a self-extracting archive file or gzipped tar file for deployment on the target machine where the pen-based application is going to be deployed.

On the target machine, the numeral recognizer may be extracted and integrated into a pen-based application as per the sample code provided.

The above describes the simplest of scenarios wherein an existing recognition algorithm (PCA) is used for creating a shape recognizer more or less out of the box. Other scenarios address the extraction of isolated shape data for training from words or larger units of writing, adding and using new preprocessing methods, experimenting with different configuration parameters, writing a new shape recognizer, and using the Boxed Field recognizer to recognize a boxed field of characters.

5. Status

The first version of LipiTk has been implemented at HP Labs, Bangalore, India, by a team composed of GDIC engineers and HP Labs researchers, and was released internally in Dec 2005. The toolkit was subsequently released into the Open Source in April 2006 under a BSD-like license, and is now hosted on SourceForge at *http://lipitk.sourceforge.net*. As mentioned earlier, this first version is aimed at isolated shape and character recognition, and includes generic tools for data collection, annotation and evaluation, source code for common preprocessing and classification, build and packaging script, miscellaneous scripts, and sample data and code.

The included simple model-based shape classification algorithms based on Dynamic Time Warping and Principal Component Analysis have been shown to perform in the range of 80% accuracy on Indic scripts such as Tamil whose character set has many similar looking characters with complex shapes [11,12]. These algorithms have also been benchmarked on standard datasets such as Unipen [20] and IRONOFF [21]. In all these cases, it is clear that while these model-based methods offer a reasonable first level of classification, discriminative methods together with other features are necessary to achieve high accuracy. The toolkit is designed to support the addition of such methods.

Internally, the toolkit has been used to create the gesture-stroke recognition component of the Gesture Keyboard for Devnagari [23]. Specifically, this uses the PCA recognizer in a rank-based combination with a global gesture shape recognizer, and yields 97% accuracy on the constrained gestures used by the Devanagari Gesture Keyboard. The toolkit is also being used internally to explore other concepts in text input of Indic scripts, and solutions such as form filling in local languages [15,16].

6. Summary and Next Steps

In summary, the Lipi toolkit effort aims to facilitate development of online handwriting recognition engines for new scripts, and simplify integration of the resulting engines into real-world application contexts. The first version of the toolkit provides robust implementations of tools, algorithms, scripts and sample code necessary to support the entire process starting from the collection of handwritten data, to the deployment and integration of a robust engine, for a particular set of shapes or characters.

The design of the toolkit makes it possible to integrate new tools and algorithms (such as a data collection tool specifically for gestures, a different type of preprocessing, or classification algorithm) into the toolkit. For instance, to facilitate training of a classifier, LipiTk provides a common interface that calls the training module of a specific classifier, while hiding the specific details of its implementation from the user.

Given the need to support the potential LipiTk user community (whether researchers or application developers) across multiple operating systems and

computing platforms, the toolkit is also designed to simplify creation of versions for different platforms using a common code base.

As already indicated, there are several important research directions for the toolkit, including inclusion of discriminative classification algorithms, native support for emerging standards such as W3C Ink Markup Language, improved tools for data collection, annotation and error analysis, and even potential extensions to Offline HWR. However, our major focus at present is to validate the design and utility of the toolkit with different sets of users. Projects within HP Labs such as the Gesture Keyboard are the first "internal" users of the toolkit, and provide an opportunity to study the usability and cross-platform robustness of the toolkit at close quarters. We are also interested in collaborative projects with university research groups using the toolkit. We hope that some of these users can contribute by trying to use the toolkit and providing feedback, while others may contribute to the toolkit by way of new tools and algorithms.

Acknowledgements

The authors would like to acknowledge Sudhakaran K., Rajesh Pandry and Vijay Kumar of the LipiTk Development team, GDIC, and M. Dinesh, Sridhar Muralikrishna, V. Jagannadan and A. Bharath from the Pen-based Solutions and Handwriting Recognition team at HP Labs India, for their contributions towards the toolkit.

References

[1] The Carnegie Mellon Sphinx Project, http://cmusphinx.sourceforge.net.

[2] The Festival Speech Synthesis System, http://www.cstr.ed.ac.uk/projects/festival/.

[3] Shanmugham, S., Monaco, P. and B. Eberman, "MRCP: Media Resource Control Protocol", Internet Draft draft-shanmugham- mrcp-05, January 2004

[4] Rosetta - Multistroke / Full Word Handwriting Recognition for X, http://www.handhelds.org/project/rosetta/

[5] XStroke: Full-screen Gesture Recognition for X,

[6] WayV Project, http://www.stressbunny.com/wayv/

[7] HRE API: A Portable Handwriting Recognition Engine Interface, http://playground.sun.com/pub/multimedia/handwriting/hre.html

[8] C.C Tappert, C.Y Suen, and T.Wakahara, "The State of the Art in On-Line Handwriting Recognition," IEEE PAMI, vol.12, No.8, pp. 179-190, 1990.

[9] Rejean Plamondon and Sargur N. Srihari, "On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey," IEEE, vol. 22, No.1, pp. 63-84, 2000.

[10] S.D Connell, R.M.K Sinha, and A.K.Jain, "Recognition of Unconstrained On-Line Devanagari Characters", in Proc. 15th ICPR, pp. 368-371, 2000.

[11] Niranjan Joshi, G. Sita, and A. G. Ramakrishnan and Sriganesh Madhvanath, "Comparison of Elastic Matching Algorithms for Online Tamil Handwritten Character Recognition", Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition (IWFHR-9), Tokyo, Oct 2004

[12] Deepu Vijayasenan, A.G. Ramakrishnan and Sriganesh Madhvanath, "Principal Component Analysis for Online Handwritten Character Recognition", 17th Intl Conf. Pattern Recognition (ICPR 2004), Cambridge, United Kingdom, August 23-26, 2004.

[13] Experiences in Collection of Handwriting Data for Online Handwriting Recognition in Indic Scripts, Ajay S Bhaskarabhatla and Sriganesh Madhvanath, 4th Intl Conf. Linguistic Resources and Evaluation (LREC 2004), Lisbon, Portugal, May 26-28, 2004

[14] Gesture Keyboard - User Centered Design of a Unique Input Device for Indic Scripts , Ashish Krishna, Rahul Ajmera, Sandesh Halarnkar and Prashant Pandit, HCI International-2005, Las Vegas, Nevada, USA, July 22-27 2005.

[15] Indic scripts based online form filling - A usability exploration, Ashish Krishna, Girish Prabhu, Kalika Bali, Sriganesh Madhvanath, 11th International Conference on Human-Computer Interaction(HCI 2005), Las Vegas, July 22-27, 2005

[16] Coffei: Common Forms Framework for Electronic Ink, Sriganesh Madhvanath et al., HP Tech Con '05.

[17] UNIPEN 1.0 Format Definition, http://www.unipen.org/pages/5/index.htm

[18] InkML – The Digital Ink Markup Language, www.w3.org/2002/mmi/ink

[19] UPX: A New XML Representation for Annotated Datasets of Online Handwriting Data, Mudit Agrawal, Kalika Bali, Sriganesh Madhvanath, Louis Vuurpijl, 8th International Conference on Document Analysis and Recognition (ICDAR 2005), Seoul, Korea, Aug 29 - Sept 1, 2005.

[20] The UNIPEN collection release #1: train_r01_v07, http://unipen.nici.ru.nl/cdroms/

[21] C. Viard-Gaudin, P.M. Lallican, S. Knerr, P. Binter, "The IRESTE On/Off (IRONOFF) Dual Handwriting Database", ICDAR'99.

[22] Representation and Annotation of Online Handwritten Data, Ajay S. Bhaskarabhatla, Sriganesh Madhvanath, M. N. S. S. K. Pavan Kumar, A. Balasubramanian and C. V. Jawahar, Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition (IWFHR-9), Tokyo, Oct 2004.

[23] Handwritten Gesture Recognition for Gesture Keyboard, R. Balaji, V. Deepu, Sriganesh Madhvanath and Jayasree Prabhakaran. Proceedings of the 10th International Workshop on Frontiers in Handwriting Recognition (IWFHR-10), La Baule, France, Oct 2006.