2009 10th International Conference on Document Analysis and Recognition

PENTOOLS - A MATLAB Toolkit for On-line Pen-based Data Experimentation

R.M. Guest
Department of Electronics, University of Kent, UK
r.m.guest@kent.ac.uk

Abstract

MATLAB provides a powerful environment for rapid prototyping of research methods and techniques. Across the wide range of on-line pen computing applications there exists a series of common methods for the capture, storage, manipulation and measurement of handwritten data which are used as the basis for novel research and system architecture implementation. This paper outlines a new open-source Toolkit for the MATLAB programming environment containing routines and data structures to perform common functionality manipulating 'on-line' hand drawing and writing data captured in the form of a time series sequence. The routines provide capture device interrogation, feature extraction and data manipulation, enabling full integration within MATLAB, thereby providing an extensible platform for experimentation and research.

1. Introduction

"On-line" data within the pen-computing community refers to writing or drawing data captured in the form of a time-series. Each sample within the time-series comprises a number of channels relating to data returned from the capture device (for example, absolute pen position and downwards pressure/force on the capture surface). On-line pen-based computing encapsulates a wide range of applications and research, including biometric signature verification, document forensics and handwriting and drawing analysis [1, 2]. Across these application areas there exists a series of common techniques for the processing, feature extraction and analysis of time-series data [3]. During research and development in these areas, a programmer/researcher often wishes to design a new algorithm or technique to process data but does not wish to reprogram (the often tedious) lower level functionality such as file reading and basic feature extraction.

The MATLAB programming environment provides an extensive collection of routines for rapid prototyping and experimentation of novel implementations [4]. Alongside basic programming structures and standard off-the-shelf toolboxes, adding functionality such an image processing and statistical processing, the academic, research and industrial communities have developed a series of public domain toolkits to provide common methods within particular applications areas. For example, PRTOOLS [5] is widely used to provide standard pattern recognition techniques for data stored in MATLAB variables (and hence interface to other toolboxes and functions). These toolboxes enable a developer to concentrate on the novel algorithmic development by simply calling common support routines without the necessity to recode.

This paper documents a novel toolbox containing routines for the processing of on-line time-series based pen data. The Toolkit contains around 130 routines which manipulate two individual data structures, the contents of which are accessible by the programmer. The first structure contains the "raw" time-series channel data (for example x and y coordinate pen positions at successive sample points) while the second structure holds derived temporal and dynamic data (such as pen velocity) for each of the sample points in the first data structure. Alongside routines to manipulate and interrogate these structures, the toolbox provides a MATLAB interface to lower level capture device interrogation using the standard WINTAB protocol [6].

The concept behind the Toolkit is to provide the pen-computing community with a series of routines to enable the rapid implementation of common methods that can interface to the extensive range of routines within MATLAB thereby reducing development time. By making the Toolkit freely available and open-source, it can be enhanced, modified and expanded to the needs of the user and the community.

978-0-7695-3725-2/09 $25.00 © 2009 IEEE
DOI 10.1109/ICDAR.2009.35

2. Data Structures

At the heart of the Toolkit are two data structures. The first *pen_sample_points* contains the raw channel data directly from the pen capture device. The structure comprises a *variable.type* containing the string "pen_sample_points" and *variable.data* which contains a 15 x *size* array where *size* is the number of individual channel sample points. Figure 1 shows the structure of *pen_sample_points* and details the contents of data at each sample point and the column in the array where each data item is stored. Data from writing devices is typically captured by polling the device at a regular interval and receiving a status packet containing multiple channel data. The *pen_sample_points* array has been designed to encapsulate both a wide range of capture devices and the channel data returned from a device (in line with the ISO/IEC 19794-7 format [7]). Not all channels will be supported by an individual device – unused channels are zero-loaded to indicate a non-return. Other routines within the PENTOOLS Toolkit which use this structure only rely on the timestamp, *x* coordinate, *y* coordinate and a pressure channel (of device specific range) being mandatorily present within the data structure. Other channels can be accessed by researchers own routines.

One of the issues within the pen computing community is a lack of standardisation in formats for storing pen data. While not replacing formats such as UNIPEN [8], InkML [9] or the recently devised ISO/IEC 19794-7 format [7], the *pen_sample_points* structure provides a unified approach to storing data for use within MATLAB. Typically this structure would be filled by a routine reading in a common format for pen-based storage format. PENTOOLS contains a routine 'open197947' which reads in an ISO/IEC 19794-7 formatted file and parses the data in a pen_sample_points format. The open-source nature of the Toolkit means that mapping proprietary file storage formats into the *pen_sample_points* structure is easily achieved.

The second structure, *pen_dynamics,* contains features at each sample point derived from data contained in a *pen_sample_points* structure. The structure is shown in Figure 2 with *variable.type* containing the string "pen_dynamics". The contents of the *variable.data* array (with one row for each sample point in the corresponding *pen_sample_points*) are also shown in Figure 2. This array contains a number of commonly derived instantaneous and accumulative temporal features which may be used by researchers to supplement their own functions. Data contents in *pen_dynamics* are generated by executing a routine 'dynamicsconstruct'. This function also contains a series of user selectable automated drawing/writing segmentation methods, the output of which (for example numeric segment membership) will be stored in the resulting *pen_dynamics* structure.

3. Pen-based Functions

Function utilising or servicing the above data structures can be divided into six broad areas:

3.1. WINTAB Compliant Device Interrogation

WINTAB provides a low-level C-structure API to a range of pen-input capture devices and has become the de-facto standard for device communication within the Microsoft Windows environment. Routines provided within PENTOOLS allow for configuration parameters (including reporting ranges) to be read from the currently connected pen device and stylus. Table 1 lists the functions that allow integration within the MATLAB environment. Currently pen status information (pen position, etc.) via WINTAB during a capture process is not implemented – future work will address this omission.

3.2. Structure Interrogation

PENTOOLS contains over 40 routines to enable the extraction of an individual array value within the *pen_sample_points* and *pen_ dynamics* structures at a specified sample point. Routines are also provided to extract all data from a particular column or row within the two data structures. These latter two routines may be used, for example, to plot a feature against time enabling of an assessment of values across the drawing process.

3.3. Dynamic/Temporal Features

Central to its functionality, PENTOOLS contains a number of standard features extraction routines using data stored in the *pen_sample_points* and *pen_dynamics* structures. Although not comprehensive (or, indeed, leading-edge), these features are commonly used within the community and enable researchers to utilise, modify and enhance for their own purposes. The range of features includes:

- Mean, standard deviation and maximum values of velocity, acceleration, jerk and pressure:
    - overall in both axes

    - in the *x* axis
    - in the *y* axis
- Starting and ending drawn coordinates.
- Routines to indicate intra-drawing quadrant of starting and ending position.
- Drawing height, width and area in pixels.
- Number of pen lifts within drawing.
- The mean *x* and *y* coordinates (the centroid) of the drawing.
- Routines to return line numbers of sample points immediately prior to the pen crossing specified *x* and *y* coordinates and sample points within a defined active 'box'.
- Routine to produce a chain code containing the 8-point quantised pen direction
- Routine to produce a string containing sample point pen travel distance.
- A binary test for pen movement.
- Total drawing time and ratio between pen drawing, off-tablet movement and pen stationary times.
- The pen-on-tablet to pen-off-tablet ratio within a drawing.
- Slant estimate within the drawing.
- Routine to produce the nearest sample point to given time.

3.4. Image-Based Features

Alongside features utilising temporal data (Section 3.3), the PENTOOLS toolbox contains a number of routines to produce images and extract information from image based representations of time-series data. These routines are detailed in Table 2. The MATLAB Image Processing Toolkit provides a natural complement to the functionality within these routines.

3.5. Support Functions

PENTOOLS also contains a number of support functions which are utilised by other routines (Table 3). These functions can also be used as stand-alone common methods within pen-based computing development. As can be seen in Table 3 two methods have been implemented to automatically segment a time-series data file. By adhering to the PENTOOL data structures, it is possible for a researcher to implement their own state-of-the-art or application specific method whilst using the other functions within the Toolkit. This same concept applies to the development of routines to read in data files stored in a proprietary format.

Table 2. Image-Based Functions.

Function	**Description**
CONTOUR	Extract upper/lower contour of a binary image.
IMCENTROID	Finds x and y coordinate centroids of a binary image.
IMMOMENT	Computes raw moment from binary image.
LOOPS	Calculates the number of enclosed pixels in binary image.
PLOTPENFILE	Draws an X, Y plot of sample data.
ROTATE	Rotate a binary image.
SHEAR	Introduce shear into a binary image/segment.

Table 3. Support Functions.

Function	**Description**
AUTOSEGMENT1	Segments a drawing by pen-up actions.
AUTOSEGMENT2	Segments a drawing by local velocity minima.
BEARING	Calculate bearing between two coordinates.
BRESENHAM	Generates a line using Bresenham's algorithm
DYNAMICSCONSTRUCT	Calculates dynamic properties at sample points.
EUCLID	Calculate Euclidean distance between two coordinates.
FILTERCOORDS	Smooths pen coordinates within a sample matrix.
OPEN197947	Open an ISO/IEC 19794-7 data file and fill *pen_sample_points* structure.
SPATIALINTERPOLATE	Interpolation of raw data by coordinate.
TEMPORALINTERPOLATE	Interpolation of raw data by sample time.

4. Conclusions

In this paper a new Toolkit has been described to facilitate common pen-based computing operations within the MATLAB programming environment. Initial functionality includes separate data structures for holding raw pen channel data from capture devices and pen dynamics, alongside routines for the extraction of on-line and off-line features, image conversion and WINTAB interrogation. As source code is available it is possible for the Toolkit to be expanded and enhanced by the research community according to application and state-of-the-art.

The Toolkit is available via email from the author.

Acknowledgements

The author acknowledges the generous support of the INTERREG IIIA EU Programme, GOSE, Kent County Council, UK and Conseil Régional de Haute Normandie, France in the development of this work.

References

[1] R.Plamondon and S.N. Srihari, "On-line and Off-line Handwriting Recognition: A Comprehensive Survey", *IEEE Transactions on Pattern Analysis and Machine Intelligence* (T-PAMI), Vol. 22, No. 1, Jan. 2000, pp. 63-84.

[2] R. Plamondon, G. Lorette, Automatic signature verification and writer identification – the state of the art, *Pattern Recognition*, 1989, 22, 2, pp. 107-131.

[3] A. Jain, F. Griess, S. Connell, On-line Signature Verification. *Pattern Recognition*, Vol. 35, Issue 12, pp. 2963-2972, 2002.

[4] MATLAB - www.mathworks.com/products/matlab/

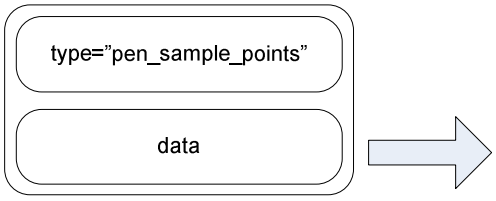
[5] R.P.W. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. de Ridder, D.M.J. Tax, S. Verzakov, *PRTools4.1, A Matlab Toolbox for Pattern Recognition*, Delft University of Technology, 2007.

[6] R. Poyner, *Wintab™ Interface Specification 1.1*, LCS/ Telegraphics, 1996.

[7] ISO/IEC Information technology – Biometric data interchange formats – Part 7: Signature/sign time series data. International Standard ISO/IEC 19794-7.

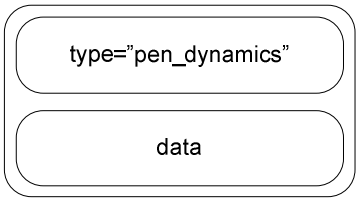
[8] Guyon, I., Schomaker, L., Plamondon, R., Liberman, M. & Janet, S. UNIPEN project of on-line data exchange and recognizer benchmarks, Proceedings of the 12th International Conference on Pattern Recognition, ICPR'94, pp. 29-33, Jerusalem, Israel, October 1994.

[9] Ink Markup Language (InkML), W3C Working Draft, www.w3.org/TR/2006/WD-InkML-20061023



Column	Channel	Description
1	Timestamp	the time offset in ms since the start of the capture.
2	X coordinate	the horizontal location of the pen.
3	Y coordinate	the vertical location of the pen.
4	Normal Pressure	the normal pen tip pressure.
5	Tangential Pressure	the tangential or barrel pressure.
6	Status	the cursor state (in/out of context etc.)
7	Cursor	the cursor type that generated the packet.
8	Context	the id of the context that generated the packet.
9	Buttons	the summation of button states (barrel button, tip etc.)
10	Azimuth	the clockwise rotation of the cursor about the z axis through a full circular range.
11	Altitude	the angle with the x-y plane through a signed, semicircular range - positive values specify an angle upward toward the positive z axis; negative values specify an angle downward toward the negative z axis.
12	Twist	the clockwise rotation of the cursor about its own major axis.
13	Pitch	the pitch of the cursor.
14	Roll	the roll of the cursor.
15	Yaw	the yaw of the cursor.

Figure 1. *pen_sample_points* structure.



Column	Channel
1	accumalative pen euclid distance
2	accumalative pen x axis distance
3	accumalative pen y axis distance
4	pen movement bearing
5	pen velocity
6	pen acceleration
7	pen jerk
8	pen x velocity
9	pen y velocity
10	pen x acceleration
11	pen y acceleration
12	pen x jerk
13	pen y jerk
14	drawing segment post-segmentation
15	pen on tablet

Figure 2. *pen_dynamics* structure.

Table 1. WINTAB Functions.

Function	**Description**
GETOPENMAXSAMPLE	Returns the maximum sample rate currently being received.
GETWTCURSORACTIVE	Returns status of the cursor.
GETWTCURSORBUTTONS	Returns the number of buttons on the WINTAB cursor.
GETWTDEVICECURSORMASK	Returns a bit mask indicating which packet data items are available when certain cursors are connected.
GETWTDEVICECURSORS	Returns the number of supported device cursors.
GETWTDEVICEFIRSTCURSOR	Returns the first cursor type number on the supported device.
GETWTDEVICEHWFLAGS	Returns device capability flags.
GETWTDEVICEMARGIN	Returns device context margins in X, Y and Z planes.
GETWTDEVICEPACKETMASK	Returns a bit mask indicating which packet data items are always available.
GETWTDEVICEPACKETMODEMASK	Returns a bit mask indicating which packet data items are physically relative.
GETWTDEVICERANGE	Returns device ranges in X, Y and Z planes.
GETWTDEVICEREPORTRATE	Returns the maximum sample rate of the connected device.
GETWTNOCONTEXTS	Returns the number of contexts supported by the current driver.
GETWTNOCURSORS	Returns the number of cursors supported by the current driver.
GETWTNODEVICES	Returns the number of devices supported by the current driver.
GETWTOPENCONTEXTS	Returns the number of contexts currently open.
GETWTVERIMP	Returns WINTAB Implementation Version Number.
GETWTVERSPEC	Returns WINTAB Specification Version Number.
GETWTWHOLESCREEN	Test to see if pointing available to whole screen.