Japanese HWR

Steven B. Poggel
steven.poggel@gmail.com

February 1, 2010

Contents

1 Handwriting Recognition Engine **5**

1.1 Data Capturing . . . 5
1.1.1 Writing Surface . . . 5
1.1.1.1 Writing Surface GUI . . . 5
1.1.1.2 Writing Surfaces Background Module . . . 5
1.2 Data Format . . . 6
1.2.1 Requirements of the Data Format . . . 6
1.2.2 Data Format Description . . . 6
1.2.2.1 Point Data Format . . . 6
1.2.2.2 Stroke Data Format . . . 6
1.2.2.3 Radical Data Formant . . . 6
1.2.2.4 Character Data Format . . . 6
1.2.3 Alternative Formats . . . 6
1.2.3.1 InkML . . . 6
1.2.3.2 The UNIPEN Format . . . 6
1.3 Database . . . 6
1.4 Recognition Architecture . . . 6
1.5 Stroke Recognition Process . . . 6
1.5.1 Advanced Point Lists . . . 6
1.5.2 Normalisation . . . 6
1.5.2.1 Boxing . . . 6
1.5.2.2 Scaling . . . 7
1.5.3 Curve Handling . . . 7
1.5.4 Dynamic Time Warping . . . 7
1.6 Radical Recognition Process . . . 7
1.7 Character Recognition Process . . . 7
1.8 Error Handling . . . 7
1.8.1 Error Recognition . . . 7
1.8.2 Error Processing . . . 7

Chapter 1

Handwriting Recognition Engine

1.1 Data Capturing

Each handwriting recognition process begins with the data capturing. The user's handwriting must be captured and fed into the system. The data capturing is therefore a crucial part of the whole process. In this system a GUI is used for the capturing of the pen movements on a writing surface.

1.1.1 Writing Surface

The writing surface module, the view is split into two parts. The writing surface GUI and the writing surface background module. The technical design of the data input GUI is described in section **??**.

1.1.1.1 Writing Surface GUI

The GUI works with pen-down and pen-up events. It has a cross in the middle in order to partition the writing surface the same way, character practicing paper sheets are usually partitioned. The GUI class is listening to *pen-down*, *pen-up* and *pen-move* events. These are the equivalent in the mobile world for regular mouse-down, mouse-up and mouse-move events. However, there is one difference - the pen-move event can only be captured between a pen-down and a pen-up event. An earlier conceptual idea for the HWR engine included using the mouse-move events during the input of a character between the strokes. This could not be realised, however, because the series of pen-move events can only be captured when there has been a previous pen-down event and no pen-up event yet. The GUI captures the events and passes the point coordinates on to the background class.

1.1.1.2 Writing Surfaces Background Module

When a pen-down event is detected, the background class of the GUI starts listening for the pen movement. All point coordinates and the time of their capturing are stored in two separate lists with the same indeces. An alternative solution would have been to store a number of instances of a custom-made encapsulated Point class including the time stamp in one list only, however, using two separate lists resulted in increased speed. Therfore, the point coordinates are stored in the frameworks Point class that does not account for time stamps. Therefore a separate list is used for the timestamps only.

The background module of the writing surface mainly administrates the capturing of the pen trajectory and sends it to the recognition module as soon as a stroke is finished. Therefore the system does not receive a separate signal indicating that the drawing of the character is finished. That design creates as segmentation problem that is solved with the *clear* button and the *clear* message. The segmentation of characters is left undetermined - only the beginning of it is determined through the *clear* message that is sent to the moblie view from the controller - or the clear message that is sent to the controller because the user clicked the *reset* button. After a stroke is finished the according point and timestamp sequences are passed to the main part of the HWR engine.

1.2 Data Format

1.2.1 Requirements of the Data Format

show requirements!

1.2.2 Data Format Description

A general description of the XML format - why it was used that way, given the requirements
data format of and representation of point, stroke, box, radical, character s. 20-23, 25f
show process of how I came to current data formats. include character models 1-4 on pages. - however, not everything in one go, but rather in the individual sections if possible. in the end, a radical has its own format that is unchanged, even if internal structure of a stroke is changed. (unipen is only text based, inkml does not help here, but the system allows for exchange of the custom format with those)

1.2.2.1 Point Data Format

1.2.2.2 Stroke Data Format

1.2.2.3 Radical Data Formant

1.2.2.4 Character Data Format

1.2.3 Alternative Formats

as opposed to other possible formats like unipen and inkml.

1.2.3.1 InkML

1.2.3.2 The UNIPEN Format

1.3 Database

1.4 Recognition Architecture

1.5 Stroke Recognition Process

1.5.1 Advanced Point Lists

what happens to the points? nothing, really - the magic happens when normalisation and the other stuff starts. why this section? what's the purpose? oh, right - angles and vectors instead of simple points. from one point to the next, or rather from on point to ten points down the line, to get a rougher direction. vectors make it interesting. impacts on curve handling! gradient and stuff can be measured in the vector representation (even without any boxes) making the point list a cool mathematical object! show code samples in pseudocode if necessary. report about the cool stuff.

what's the similarity measure for points and strokes? show requirements. what alternatives were there to consider?

1.5.2 Normalisation

what is N? why do N? show requirements. how is N performed here? why is it performed like that?

1.5.2.1 Boxing

how is boxing done? show requirements. what alternatives were there to consider? is it useful to have a similarity measure for bounding boxes? yes! but why? explain! size of the boxes! - think of characters that only have two strokes.

1.5.2.2 Scaling

s. 42-45 how is scaling done? show requirements. what alternatives were there to consider?

1.5.3 Curve Handling

S 14, 16, 17 how is curver handling done? show requirements. what alternatives were there to consider?
stroke matching with angles instead of point position. s. 24

1.5.4 Dynamic Time Warping

what's the similarity measure for points and strokes? show requirements. what alternatives were there to consider?

s. 51 how is dynamic time warping done here? pointer to papers or hwr - chapter, don't explain DTW here. show requirements why DTW? what alternatives were there to consider? none - it is the alternative. to all the other stuff I've been doing. however, what about 3D time warping?

1.6 Radical Recognition Process

1.7 Character Recognition Process

1.8 Error Handling

see section **??** in chapter **??** for possible sources of error

1.8.1 Error Recognition

why this section? to demonstrate own achievements of error recognition. the reader should know how it is done technically.

what goes into this section? the aspects of finding errors. finding errors is not a straightforward trivial task - whenever something does not match it is an error - doesn't work like that. instead, firstly, it needs to be made sure that it actually is an error. meaning - not a recognition error, but a user error. secondly, the type of error needs be identified. see section **??** (or handwritten page 58) for sources of error.

how will this section be written? technical - first describe how the error recognition integrates into the recognition process, then how errors are identified.

1.8.2 Error Processing

why this section? actually the 'handling' or 'processing' aspect could be described in the recognition section 1.8.1 as well. so this section is only for a better overview, for document structure, thematically they are the same section. thus they are put together under Error Handling 1.8.

what goes into this section?

List of Figures

List of Tables

References