XPEN: An XML Based Format for Distributed Online Handwriting Recognition
A.P.Lenaghan, R.R.Malyan,
School of Computing and Information Systems, Kingston University, UK
{a.lenaghan,r.malyan}@kingston.ac.uk

Abstract
Architectures for integrated and distributed handwriting recognition systems are discussed. An XML (eXtensible Markup Language) based representation for online handwriting data, referred to as XPEN, is proposed. XPEN is based on the earlier UNIPEN format. The flexibility of the new format is illustrated with an example of the use of XSLT (XSL-Transformations) to translate XPEN into the Scalable Vector Graphics (SVG) format for visualisation and the processing of XPEN using a programming language via the Document Object Model (DOM) Application Programming Interface.

1. Introduction
Improvements in cellular and wireless LAN technologies increasingly mean that handheld devices are networked. Where these devices incorporate online handwriting recognition as part of their user interface, network connectivity offers the possibility of delegating recognition to dedicated servers in the network. This potentially keeps the requirements for storage and computation on the mobile device to a minimum without compromising the sophistication of the algorithms used for recognition. Moreover, distributing the recognition task naturally extends to the use of multiple expert classifiers, a trend already seen to offer gains in some recognition systems [1].

Central to the issue of distribution is the format in which data is exchanged. We propose a new XML (EXtensible Markup Language) based format derived from earlier work on the UNIPEN format [2] to fulfil this role.

The remainder of the paper is structured as follows. In section 2 integrated and distributed handwriting recognitions systems are discussed. Section 3 explains the origins and rational behind the new XPEN format. Section 4 introduces the details of the XPEN format. Section 5 illustrates the flexibility of the format with examples of it being manipulated with a generic XML tool and from a programming perspective. Finally we close with a brief discussion and summary.

2. Distributed Handwriting Recognition
A recognition system can be logically subdivided into three components responsible for; capturing data, classifying data, and a knowledgebase holding models of words or characters known to the system. Figure 1 shows the architecture for an integrated recognition system in which all three components are hosted within the same machine.
the components. There is little need to incur the overhead of adopting a formal standard for such exchanges. With a distributed system, however, an agreed format in which to exchange data over a network becomes necessary. This needs to be platform and programming language independent because separate components may be developed by differing teams and in different languages. The separation emphasises the need for a common, unambiguously defined yet easily manipulated format.

The second issue that distributing the components of a recognition system brings is that it becomes natural to consider making use of multiple recognition engines. The user agent may submit requests to several remote engines and combine their responses. The user agent need have no knowledge of the classification algorithm employed on its behalf providing an agreed protocol for the exchange is followed.

3. Rational for new format

Offline handwriting recognition based on images has always been able to draw upon a range of general-purpose image formats. Lossless image formats such as GIF (Graphics Interchange Format) and PNG (Portable Network Format) can be used to represent bitmaps or greyscale data suitable for OCR and offline recognition systems. By contrast, online handwriting recognition has required more specialised formats to represent vector-based information. At the core of such representations are sequences of (x,y) coordinates produced by tracking a pen tip as symbols are written.

An early format for online data adopted within the research community was the UNIPEN format. This provided a common representation that could be interchanged between researchers and opened the way to the independent benchmarking of recognition systems against established data sets. In addition to the basic (x,y) sample points UNIPEN also provided a mean for capturing experimental details about:
- the writers,
- the data capture equipment used
- the outcome of the segmentation and classification processes.

In several ways UNIPEN foreshadowed the trends for XML based data formats that followed it. It was an open format available for all, ASCII based for readability and simplicity, and has a formal definition against which data sets can be tested for their validity.

With the existing UNIPEN format the question is: Why propose a new format for online data? The issues are i) what limitations of the existing format does a new format address, and ii) what additional potential benefits does a new format offer?

In developing an XML based format we have adopted a pragmatic stance. The new format, which we refer to as XPEN, is based on the existing UNIPEN format. It is an evolution of the format as it came to be used (rather than its original definition). A minimal number of changes have been made where necessary to bring it into an XML based format. The benefits are the opening up of the format to a range of tools, specifically:
- Generic tools to view, edit, and validate the format against a formal definition.
- Tools to transform the format into other XML applications for visualization.
- Platform and language independent API’s to parse and manipulate XML.

In short, a great deal of effort has been expended in developing tools for manipulating XML, translating online data into XML opens the way to use these general purpose tools. This frees developers to focus on specialised tools for those aspects of the handwriting recognition such as extracting features and classifying symbols that are unique to the domain.

The UNIPEN format was developed against a background need for archiving datasets and supporting research, which it does well. It was not, however, developed for the real-time exchange of data required for distributed recognition system or to be extensible.

4. Translating UNIPEN to XPEN

The translation of the UNIPEN format to XPEN is shown in Figure 2. A script developed using the Python Programming language performs this. Directives under the UNIPEN format are mapped to elements and attributes...
in XML. Kassel’s data set [4], which makes use of a range of the facilities of the UNIPEN format, was taken as the basis of the XPEN format.

**Figure 3 Translating UNIPEN to XPEN**

From the empirically generated examples of XPEN a Document Type Definition (DTD) formally describing the format is generated automatically. The DTD can subsequently be used to validate instances of files containing the XPEN format.

For example, the snippet of UNIPEN describing the beginning of a pen stroke:

```
PEN_DOWN
527 1302 0
527 1291 5
538 1250 10
...
PEN_UP
```

is translated to corresponding XPEN format:

```
<?xml version="1.0"?>
<xpen>
<header> ... </header>
<body>
<stroke>
  <sample time = "0">
    <x>527</x>
    <y>1302</y>
  </sample>
  <sample time = "5">
    <x>527</x>
    <y>1291</y>
  </sample>
</stroke>
...
</body>
</xpen>
```

Whilst the core sample data remains the same there are several changes to note about the XPEN formatting.

Firstly, whereas UNIPEN is a flat file format with its directives existing at the same semantic level, XPEN is inherently a hierarchical format. It includes elements defined by start-tags and end-tags that can be nested. The single top most element is the `<xpen>`...`<xpen>` element. The line preceding the `<xpen>` element is a processing instruction defining the version of XML used to encode the data [5]. The `<header>` element contains all information usually captured in UNIPEN relating to the writers identity and the sampling equipment (not shown). The `<body>` element contains the sample data describing the handwriting.

Secondly, at a stroke level, there is a change of perspective from describing events during writing to a hierarchical data structure. Whereas UNIPEN uses .PEN_UP and .PEN_DOWN writing events derived directly from the sampling process, XPEN explicitly defines a `<stroke>` element to contain the data describing a pen stroke.

Finally, at the level of an individual sample point, whereas the UNIPEN format implicitly interprets the meaning of the data (in this case (x,y,t) tuples are delimited by new lines) under the XPEN format the meaning of individual values is explicitly indicated. A `<sample>` element is introduced to contain all the information relating to an individual sample. The `<sample>` element has an attribute ‘time’ representing the timestamp and contains sub elements for the data corresponding to the sample. In this case `<x>` and `<y>` elements for the Cartesian coordinates of the sample location are show. This could be extended to include additional sampled information, example writing pressure or pen angle.

**5. Manipulating XPEN data**

shows two examples of processing of the XPEN format are briefly described to illustrate the flexibility of the format.

The first example (on the left) uses standard XML processing tools. The second example (on the right) shows the format being manipulated using a programming language.
5.1 Use of generic XML tools to manipulate XPEN

The XSLT (eXtensible Stylesheet Language – Transformations [6]) defines a document transformation language. This allows an XML document in one format to be mapped to another format. An XSLT processor was used to translate XPEN into the Scaleable Vector Graphics (SVG) standard [7]. SVG is the application of XML to a language for describing drawings constructed from simple graphics primitives such as lines, shapes and text. The transformation was specified by a XSLT stylesheet defining how the elements and attributes in XPEN are mapped to SVG entities. Figure 5 shows the result of rendering the output via Adobe’s SVG tool.

```xml
<?xml version="1.0" encoding="utf-16"?>
<svg width="20cm" height="20cm">
<polyline
  fill="none"
  stroke="blue"
  stroke-width="10"
  points="527 1302 &&xD;&&xA;
  527 1291 &&xD;&&xA;
  538 1250 &&xD;&&xA;
  538 1187 &&xD;&&xA;
  ...
"/>
</svg>
```

Each <stroke> element in XPEN is mapped to a <polyline> under SVG. The contents of the <x> and <y> elements in XPEN are mapped into the ‘points’ attributes of a <polyline>. The ‘&&xD;&&xA;’ represent the hexadecimal values for carriage return and new line.

This is an example of generic XML tools being used to process and in this case visualize online handwriting recognition data.

5.2 Programming with XPEN

The second example of XPEN processing illustrates the use of an API (Application Programming Interface). The Document Object Model (DOM [8]) is an API for accessing the elements XML documents. It provides a standardized means to parse an XML based format into a hierarchical or tree based data structure. To assess the ease of use of the API some basic pre-processing operations and a simple viewer were implemented. This included reading in and parsing the data representing two characters, computing their bounding boxes and centres of gravity, normalising them, and plotting them overlaid on each other (Figure 6—shows two ‘4’s normalised by aligning their centres of gravity).

```
<?xml version="1.0" encoding="utf-16"?>
<svg width="20cm" height="20cm">
<polyline
  fill="none"
  stroke="blue"
  stroke-width="10"
  points="527 1302 &&xD;&&xA;
  527 1291 &&xD;&&xA;
  538 1250 &&xD;&&xA;
  538 1187 &&xD;&&xA;
  ...
"/>
</svg>
```

The SVG format generated to correspond to the XPEN snippet shown earlier is:
Figure 6 DOM API in use with the XPEN format

The python code:

```python
xmlfile1 = open('c:\prog\match\S001d05.xml')  # open file
filedoc = xml.dom.minidom.parse(xmlfile1)  # parse xpen
samples = filedoc.getElementsByTagName("sample")
```

is all that was needed to read in and extract the samples from a file in the XPEN format. The complexity of parsing the data and iterating over the tree structure to extract the samples is delegated to the library implementing the DOM API. The DOM API is available for several languages including, Java and Python.

6. Discussion

A potential use of distributed recognition systems that XPEN aims to enable is to overcome the traditional trade off between employing general models and personalised models of peoples writing. Personalized models offer better classification but are time consuming to create. Although tedious, such training can be justified over the lifetime of a regularly used device. However, devices that are used infrequently, or perhaps written on just once by a user must fall back to using generic models. A distributed approach offers a way to mitigate this problem. A user can train a single recognition engine and then when writing on any network connected device they can instruct the user agent to direct their writing to the remote engine containing their personalise model.

7. Summary

The trend towards network connectivity for mobile devices incorporating handwriting recognition interfaces brings the possibility of distributing the recognition task between a local user agent and (potentially several) remote recognition engines. The exchange of online data requires a protocol and central this is a format in which to send data to a remote server. An XML based format derived from the UNIPEN format has been proposed for this role. The flexibility of the format was illustrated by showing its translation by a generic XML tool into the SVG format for visualisation. The ease of programming was demonstrated using the DOM API by developing a simple application to pre-process and view the XPEN format.

References.


Recommendation, Nov. 2000,
http://www.w3.org/TR/DOM-Level-2-Core