Image-based interactive tools for entering ground truth data

Jun Liu

University of Nevada, Las Vegas

Follow this and additional works at: https://digitalscholarship.unlv.edu/rtds

Repository Citation

This Thesis is brought to you for free and open access by Digital Scholarship@UNLV. It has been accepted for inclusion in UNLV Retrospective Theses & Dissertations by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
IMAGE BASED INTERACTIVE TOOLS FOR ENTERING GROUND TRUTH DATA

by

Jun Liu

Master of Science
Department of Chemistry
University of Nevada, Las Vegas
2000

A thesis submitted in partial fulfillment of the requirements for the

Master of Science Degree
Department of Computer Science
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
August 2002
The Thesis prepared by

Jun Liu

Entitled

Image Based Interactive Tools For Entering Ground Truth Data

is approved in partial fulfillment of the requirements for the degree of

Master's of Science in Computer Science

Examination Committee Chair

Dean of the Graduate College

Examination Committee Member

Examination Committee Member

Graduate College Faculty Representative
ABSTRACT

Image Based Interactive Tools for Entering Ground Truth Data

by

Jun Liu

Dr. Kazem Taghva, Examination Committee Chair
Professor of Computer Science
University of Las Vegas, Nevada

We report on the design and implementation of an interactive system for capturing the logical description of textual documents. This system is equipped with a graphical user interface which guides the user in defining components of the logical structure of the document while viewing document page images. The system is expected to enable faster and more accurate ground truth document data collection.
# TABLE OF CONTENTS

ABSTRACT ................................................................. iii

LIST OF FIGURES ................................................... v

ACKNOWLEDGEMENTS ............................................. vi

CHAPTER 1  INTRODUCTION ......................................... 1

CHAPTER 2  BACKGROUND ........................................... 6
  Java Language ......................................................... 6
  TIFF Image Files .................................................... 8
  JAI API ........................................................................... 8
  XML .............................................................................. 10
  XML Parser ............................................................. 11

CHAPTER 3  INTERFACE DESIGN AND IMPLEMENTATION ..... 13
  The Input Document Of The Image Based Interactive Tools .... 14
  Interface building, Image Loading and Interface Manipulation ... 17
  Ground Truth Data Collection ........................................ 23
  Output File ...................................................................... 27

CHAPTER 4  RESULTS AND DISCUSSION ............................. 33

CHAPTER 5  FUTURE WORK ........................................... 43

BIBLIOGRAPHY .......................................................... 45

VITA ............................................................................. 48
LIST OF FIGURES

1.1 Autotag System ................................................................. 3
1.2 The Hierarchical Tree Of The Document's Physical Representation 4
1.3 The Hierarchical Tree Of The Document's Logical Representation 5
3.1 The Main Parts of The Image Based Interactive Tools ............... 14
3.2 Example output of OCR device in JDB format ......................... 15
3.3 The Bounding Box of One String ........................................ 16
3.4 The Example of XML File for EXML Parser ............................. 18
3.5 The Input Into The Interface ................................................ 19
3.6 The Design of Interface ...................................................... 20
3.7 The Graphic Interface ......................................................... 22
3.8 The "Total Pages" of the Selected Document ............................ 23
3.9 Choosing A File From The File Menu .................................. 24
3.10 The TIFF Image with All Strings in Their Bounding Boxes ........... 25
3.11 String Collection ............................................................. 28
3.12 Finish the Selection of One Sentence .................................. 29
3.13 Tag Selection ................................................................. 29
3.14 The Ground Truth Data ..................................................... 30
3.15 Finishing the Collection on One Page .................................. 31
3.16 Example of An Output File ................................................ 32
4.1 The Example Of A Misrecognized OCR Error ......................... 35
4.2 The Example Of One TIFF Image ........................................ 38
4.3 The String ID of "2000" ..................................................... 39
4.4 The String ID of "Department" ............................................. 39
4.5 The String ID of Zip Code .................................................. 40
4.6 The String ID of Footer ...................................................... 40
4.7 The Last String ID on Separated Paragraph ......................... 41
4.8 The First String of Separated Paragraph ............................... 42
5.1 Use Input String to Correct an OCR Error ............................ 44
ACKNOWLEDGEMENTS

I am grateful to Allen Condit, Steve Lumos for their many helpful suggestions on this thesis, and permission to reference some useful sources about "Autotag" system. I would like to thank Ron Young for solving so many unexpected system problems for me, Jeffrey Coombs for his great help on the Latex format and grammar checking of my thesis.

Especially, I would like to thank my advisor, Dr. Kazem Taghva for his instructions and guidance in this research and his financial support, which makes it possible to finish this research.

I also wish to thank all of my family members' spirit supports in all these years' study and research.
CHAPTER 1

INTRODUCTION

With the fast development and growing use of the internet, more and more hard-copy documents must be transferred to their electronic form. Usually there are two possible ways to convert a paper document to its electronic form: entering from the keyboard or scanning with Optical Character Recognition (OCR) technology. The former is very slow and time-consuming. What's more, it is very easy to make manual mistakes. The latter is much faster, more economic and efficient, and less error-prone, and time-consuming. Therefore, most printed documents use OCR technology. In spite of these advantages, OCR technology still has two main problems as keyboard entry method.

- OCR devices are not free of errors. The effect of OCR errors to information retrieval is an important topic. Lots of studies have been done, and some results were obtained. In paper [18], it was proved that "average precision and recall is not affected by OCR errors across systems". In paper [9], it was concluded that high quality OCR devices won't affect accuracy of retrieval too much, but low quality devices used with databases of short documents can result in significant degradation. But paper [19] showed that OCR errors have little effect on average precision not only for full text collections, but also for short documents with the SMART retrieval system.

- After conversion to its electronic form, the document is always in its physical representation format, which means it only keeps its physical structure, but loses
the logical structure and the original features (such as: hierarchical structure, artwork, points of emphasis, etc.) made by its author.

The Information Science Research Institute (ISRI) at the University of Nevada, Las Vegas (UNLV) built a document processing system “MANICURE” [24] to detect and correct OCR errors and to mark the logical components of the text automatically. “MANICURE” consists of four modules: a physical document parser (“Doc_parse”), the logical document tagger (“Autotag”), the post processing system (“PPSYS”), and the semi-automatic user interface (“Rummage”). The “PPSYS” [17] is the module used to correct as many OCR errors as possible. The “Autotag” system [23] automatically captures the logical structure of documents, from the document’s physical representation as seen in Figure 1.1 [20]. This logical representation can be passed to rummage for display on the original document’s images.

The “Autotag” module starts with the physical representation of documents which include objects such as character strings, geometrical information, etc. and produces a logical structure of the document. This logical structure identifies objects such as authors, title, sentences, and paragraphs. In identifying these objects, “Autotag” relies on some intelligent routines which are heuristic in nature. In order to test the performance of systems, such as “Autotag”, we need to prepare a test collection which is the subject of this thesis. Figure 1.2 [22] and Figure 1.3 [21] show the detailed structures of the input and output of “Autotag”.

The aim of this study is to create a graphical user interface (GUI) to display and manipulate the information of a document’s physical representation directly, and collect the data with logical meaning for future information retrieval. It takes the physical representation as the input, parses it by Electric XML Parser (EXML) and stores the useful information (such as ids and geometric information of all strings) into arrays, then builds the user-friendly interface to display its TIFF image. On the interface, the user is able to choose any string (“data”) by mouse clicking to
Figure 1.1: Autotag System

construct the logical information he needs, such as the content of running headers and footers, title, subtitle, author, abstract, etc., then output this information into a file for future retrieval.

This interface is a semi-automatic tool that provides a very convenient, less time-consuming and less error-prone way for the user to collect ground truth data from the documents, and to build the components of the logical structure of the document.

In the next chapter some related background of this research will be introduced, and in the third chapter the detailed design and implementation of this interface will be explained. The conclusion is made in chapter four, and possible related future work will also be mentioned.
Figure 1.2: The Hierarchical Tree Of The Document's Physical Representation
Figure 1.3: The Hierarchical Tree Of The Document’s Logical Representation
CHAPTER 2

BACKGROUND

The goal of this research is to design and build the image based interactive tools, which mainly includes a graphical user interface designed with Java Swing and JAI (Java Advanced Image) API (Application Programmer Interface). This interface is able to load the image document, which is in TIFF file format, and let the user collect ground-truth data, obtained from the parsed document’s physical representation in XML format, and then build the logical structure of tags and store this logical information in an output file. In this chapter, some concepts related to the design of the tools are described, such as: the reason to choose Java Swing as the programming language and JAI as the imaging API in the design of the graphical user interface and some background about the input of this interface, such as: image formats, XML format and XML parsers.

Java Language

Java, a new, general purpose, object-oriented programming language from Sun Microsystems, was first designed for embedded consumer-electronic applications in 1991. It was initially called “Oak” by James Gosling, the designer, because of the tree outside of Gosling’s office window, and then hastily renamed “Java” because of trademark conflicts [12]. The Java 1.0 programming environment was announced on January 23, 1996 [16]. For the past few years, the Java programming language has experienced tremendous success and explosive growth because of the following advantages.
Based on the C and C++ programming languages, Java language differs from them in several important ways.

- In Java, all development is done with objects and classes. Java is object-oriented programming language that models the real world. this is the main difference. Object-oriented methodology (OOM) is an attempt to design computer programs based on intuitive, real-world objects. Since Java is entirely based on this methodology, it follows all the principles of OOM [7]:

  - classification: grouping the "objects" based on shared traits.
  - inheritance: each subclass can "inherit" its state (in the form of variable declarations) from a "super" class, and also may "override" the inherited methods.
  - polymorphism: the ability of sibling classes to share a common "super" class ancestor.
  - identity: correlating a real object with its class.
  - aggregation: building a class from other classes.

- Java uses automatic memory allocation and garbage collection, so it is a much simpler programming language compared to C++.

- Java is architecture neutral, which means it is platform independent. It runs on most major hardware and software platforms, including Windows 95 and NT, the Macintosh and UNIX. therefore. it is more portable than C-based languages.

- Java is one of the first programming languages to take security into consideration as part of its design. The compiler, interpreter, and Java-compatible browsers all contain several levels of security measures to deal with security problems, such as: loss of data and program integrity, etc.
• Java has packages of classes which provide extensive multimedia facilities, such as: images, sounds, and animation, etc..

In short, Java is “A simple, object-oriented, network-savvy, interpreted, robust, secure, architecture-neutral, portable, high-performance, multi-threaded, dynamic language” [26].

**TIFF Image Files**

In this research, all hard-copy documents are scanned and saved as TIFF images. All image files can be categorized into two kinds: bitmap-based and vector-based files. A bitmap is an array of “pixel” (picture elements) values. Each bitmap-based image is mapped into a grid of squares or pixels, where each pixel is either black or white, on or off. The size of the grid is based on the image’s resolution. There are three common bitmap standard files: TIFF (Tag Image File Format), GIF (Graphic Interchange Format) and JPEG (named by the Joint Photographic Experts Group in 1987) [1]. The main difference among these three files are: JPEG files are lossy compression with file data losses. *Lossy compression* reduces a file by permanently eliminating certain information, and not all original data can be recovered when the file is uncompressed. GIF files are limited to 256 colors. There are color losses, but the file format itself is lossless. Usually on the web if lossless compression is needed, then GIF files are used. Compared with JPEG and GIF files, TIFF files are lossless and uncompressed [6], they won’t throw away any image data. Therefore, they are the best possible file formats for graphic designers.

**JAI API**

In Java’s early versions, only a limited number of image files (such as: GIF and JPEG) could be read in by its Toolkit object and displayed, but there are no image processing operators in Java AWT. However, in this research, all documents are saved...
as TIFF images, which can not be read in by the Toolkit object. Java 2D API extended AWT by adding support for more general graphics and rendering operations. JAI [14] is a set of classes that offers imaging functionality beyond that of Java 2D and the Java Foundation classes. By incorporating sophisticated, high-performance image processing into Java applets and applications, the Java Advanced Imaging (JAI) API further extends the Java platform (including the Java 2D API). Compared to other imaging solutions, JAI has many advantages: cross-platform imaging, distributed imaging, device independence, flexibility and extensibility, interoperability, etc. Therefore, JAI is a very powerful image processing API and it provides a simple programming model that can be readily used in Java applications without a lot of mechanical programming overhead. JAI provides a core set of operators that support basic imaging technologies and offer an extensible framework that allows customized solutions to be added to the core API for specialized applications.

This research is using JAI API for reading in and processing the TIFF images, in that JAI is able to read in several kinds of images, such as TIFF, JPEG, GIF, BMP, PNG, PNM, etc. The method “readImage” in the “JAII mageReader” class [3] is used to read in the document TIFF image, and the constructor of the “IconJAI” class [2] is used to load the TIFF image on an icon.

Since TIFF format is uncompressed, it is too large to display the whole page on the screen, therefore, it needs to be resized when being loaded on the interface. In the JAI API, geometric transformations provide the ability to reposition pixels within an image. There are several geometric transformation operations: Affine, Rotate, Scale and Translate. Scaling (Scale), also known as “minification and magnification” or “enlarges or shrinks” an image, is used in this research to shrink the image. The interpolation method of “add” in the “ParameterBlock” class [4] and the method “JAI.create” [5] are used in this design to scale the TIFF image files.
XML

For this research the output of OCR scanned documents was parsed into the physical representation, and then stored as XML files. The Extensible Markup Language (XML) is a markup language for documents containing structured information. It is a set of rules for designing text formats that let anyone structure his data [10]. Although XML looks quite new: as a project of the W3C (World Wide Web Consortium), its development started in 1996 and has been a W3C Recommendation since 1998. Its technology isn't that new. XML is defined as an application profile of SGML, the Standard Generalized Markup Language defined in ISO standard 8879:1986, which was developed in the early '80s and has the ISO standard since 1986. SGML requires users to provide a model of the document being produced, in order to let the computer do as much of the work as possible. This model, called a Document Type Definition (DTD), describes each element of the document in a form that the computer can understand. The DTD shows how the various elements that make up a document relate to one another.

One major shortcoming of SGML is its complexity, even though it is very powerful. XML is an abbreviated version of SGML, no less powerful and more regular than SGML, and much simpler to use. XML may or may not have DTD, but SGML must have a DTD.

There are two categories of XML documents: well-formed and valid. A well-formed document means it obeys the syntax of XML and follows certain rules on tags, attributes, entities, etc [15]. A valid XML document means that the well-formed XML contains a proper document type declaration (DTD), and the document obeys the constraints of that declaration. An XML document may be invalid, but it must be well-formed. If a document is not well-formed, then it is not XML.

XML is a family of technologies, which offer useful services to accomplish various tasks, such as XLink which shows a standard way to add hyperlinks to an XML file.
XPointer and XFragments are syntaxes in development for pointing to parts of an XML document. XSL which is the advanced language for expressing style sheets.

**XML Parser**

The Document Object Model (DOM) APIs [25] and the Simple API for XML (SAX) APIs [8] are currently the two most popular APIs for manipulating XML documents of “JAXP” (Java API for XML Processing) in the Java XML Pack Summer 02 Release. The most important difference between SAX and DOM is that SAX is the event-based parsing strategy and DOM is the tree-based parsing strategy. SAX presents the document as a serialized “event stream” (a sequence of calls to a handler function as each chunk of XML syntax is recognized), while DOM builds a tree representation in memory. Because a major disadvantage for SAX is that it does not support random-access manipulation of the document, therefore, in the early stages of this research, DOM API was tried.

In a pdoc file, all “strings” are embedded among “Lines”, “Lines” among “zones”, and “zones” among “pages” (shown in Figure 1.2). The strategy that DOM builds a parsing tree in memory makes it more complicated to extract the information of all strings. The recursive method of “post-order tree-traversal” has to be used to navigate all strings in different lines, zones, and pages. What's more, it is also not easy to grab the string texts from “strings” using DOM. Therefore, another simple XML parser: Electric XML Parser was used to replace DOM in this research. The Electric XML Parser is a small (packaged as a stand alone 47K JAR file), simple XML parser, with the native support for the W3C DOM interfaces, which means EXML natively implements the DOM interface, and can be passed to third party libraries that operate on DOM documents. It is much faster than other XML parsers, and flexible, easy to use XML pattern to match algorithms with support for constraints. In short, “Electric XML is a simple, fast, and comprehensive Java toolkit for parsing
and manipulating XML documents" [11]. This study uses *Electric XML* to parse the input files, grab and store the information of strings for future use.

In this chapter, some background knowledge of this research is discussed. The detailed design and implementation of the image based interactive tools will be described in the next chapter.
CHAPTER 3

INTERFACE DESIGN AND IMPLEMENTATION

Ground truth data (GTD) are the original data information existing in the document's physical representation (pdoc file), which are built by parsing the output of the OCR device. The important information of GTD that this research needs includes: document identification, page number, total pages in one document, especially, the information of the smallest elements - "strings" in pdoc files: string's id, geometric coordinates, text, etc.. The selected information of "string text" is used to construct the logical, descriptive structure information, such as the contents of title, abstract, author, header, footer, etc..

Usually, there are two ways for the user to collect GTD: either type in the data manually or use some interactive tools. Two notorious disadvantages exist for the keyboard entry: it is time-consuming and error-prone. The ideal way to collect GTD is to build image based interactive tools: a graphical interface to display the document page by page, and then let the user choose any needed data with a mouse click directly on the displayed page, which is also selected by user. Meanwhile, the components of the logical structure, constructed from the selected ground truth data, should be further confirmed by the user and then stored automatically to an output file at run time.

The design of the image based interactive tools consist of three main parts:

- Parse the input document file with EXML, and store all necessary information. The stored information will be used when user manipulates on the interface, such as choosing a document, selecting certain page, collecting strings or making

13

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
confirmation.

- Build the graphical user interface with Java Swing and JAI API. the interface provides the user many functions which enable him to choose any document file, select any page number to load this page’s TIFF image, and then collect the GTD with the mouse click, as well as choose the tags for the collected data, and make further confirmation as well.

- Write the collected results of the components of the logical structure to an output file in XML format.

The whole design of the image based interactive tools is shown in Figure 3.1. This chapter gives the detailed explanation of each part’s design and implementation.

The Input Document Of The Image Based Interactive Tools

The OCR engine transfers the scanned document into its electronic form (saved in JDB format). JDB is “a package of commands for manipulating flat - ASCII databases from shell scripts.” [13]. JDB files are the files filled with columns of
numbers, which are easy to output from OCR program, and easy to feed to other programs. An example of JDB output appears in Figure 3.2.

However, this output is hardly readable for those who do not understand the meanings of all columns, and even for those who understand them, it is hard to remember. Therefore, some work is done to use an OCR-dependent module parser to extract necessary information and then put information from every page together to build a physical representation of that document (saved as the pdoc file). This physical representation is in the form of a hierarchical tree, with "document" as the root, and "string" as the leaf, which is shown in Figure 1.2 in Chapter 1.
The purpose of this research is to find an easy way to construct the logically
structured information (such as: document title, abstract, author, paragraph subtitle,
running header and footer, etc.), which is done by the user's selection of strings on
the image. Thus, the geometric information of string is extremely important. The
pdoc file provides this information for every parsed string, which is actually the
information of two pairs of coordinates: the upper left coordinate \((x_1, y_1)\) and the
lower right coordinate \((x_2, y_2)\). This constructs a bounding box of this string. One
string example with text of “OneString” is shown in Figure 3.3.

The pdoc file contains all ground truth data which are important for the con­
struction of logically structured information. The design idea for extracting and
manipulating the data is as follows:

1. Extract the bounding boxes information from the pdoc file with Electric XML
Parser, and save it.

2. While the user selects a string with a mouse click, the current coordinate of
the mouse is compared with all the bounding boxes already saved, and then
the string text and the string id of the matched bounding box on the image are
returned for the user's further confirmation.

As shown above, the first part is processed in “FileParser.java” class, and the
second part is manipulated on the interface in "ParseLoad.java" class. The physical representation of the document in XML format is the input of this interface. The "FileParser.java" class uses Electric XML to parse the physical representation (pdoc) which should be stored as XML format. In ISRI for the research of "Autotag" system [23], lots of pdoc files are already stored as SGML form, and it is known that XML is an abbreviated version of SGML. Therefore these SGML files can also be used in this research. As long as a SGML file is well-formed, it can be converted to an XML file. The example of one pdoc file output saved in XML is shown in Figure 3.4.

The "FileParser.java" file returns the important information to the main file "ParseLoad.java": such as the document id number, the page number of that image, total pages of this document, and the information of all strings on one page as well. In particular, it returns this page's TIFF image path. Every page's scanned image is saved as TIFF image file, and consecutive page images are stored under the same document directory. All these ground truth data are needed during the interface manipulation in the "ParseLoad.java" file. Figure 3.5 shows the outline of the input for this research.

Interface building. Image Loading and Interface Manipulation

"ParseLoad.java" is the main file to build the graphical user-friendly interface and provide many functions for user to manipulate on the interface and collect the ground truth data. As mentioned in Chapter 2, the Java Swing language has multiple levels in its containment hierarchy. In this interface design, the top level container is a JFrame. There are two intermediate containers: JScrollPane and JPanel. In the design of this interface, JScrollPane is used as a container for the image file loading, and as a superclass of MouseControl class, which also means that MouseControl is a subclass of JScrollPane. Mouse manipulation, such as a mouse click, is done in this container. Then JScrollPane is added to another intermediate container: JPanel,
Illegibility does not impact the MOL.

Figure 3.4: The Example of XML File for EXML Parser
which is then added to the top level container: JFrame. The atomic components, including the “file” menu, several buttons, and two textfield areas, are added to JPanel intermediate container. The hierarchy diagram of this interface is shown in Figure 3.6, and the graphic interface is shown in Figure 3.7.
Figure 3.6: The Design of Interface
The user selects a physical representation of the document file from the file list in the “file” menu. The selected file is parsed automatically by “FileParser.java” class. As described early, the important information (such as: document id, string geometric coordinates, etc.) parsed in XML file is saved and would be returned to this “ParseLoad.java” class whenever it is needed. The selected document’s id is returned and shown in the textfield area beside DOCID. its first page’s TIFF image is loaded on the screen, and other information such as: total pages, string ids, string contents, string coordinates, are also returned and available for the user’s selection. For example, the user can get the information about the total pages of the selected document by clicking the “About” in “File” menu. one small window will pop up to show the total pages in this document (shown in Figure 3.8).

The buttons of “prev”, “next” let the user choose the previous page or next page of that document. The “addBox” button displays all strings in their own bounding boxes, their geometric coordinates are already in XML file after OCR scanning and stored after parsing. The “clearBox” clears these bounding boxes. The user may also select any page image as he likes by typing in the page number in the textfield. Figure 3.9 shows this graphical interface with “File” menu clicked. Figure 3.10 shows the image when “addBox” button is clicked. The image when “clearBox” is clicked looks the same as Figure 3.7. The user may also load any page image of this document by typing in the page number in the textfield at the “page”. If the page number that user entered is larger than the total number or smaller than 1, the word of “WRONG” will be shown in the textfield of “page”.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CONCLUSIONS

Installation of a similar surveillance system should be considered at the cost of other systems. The video-surveillance system at Victoria, St. Helen has proven to be useful, efficient, and reasonably reliable and is easy to use by reducing exposure in risk and hazardous areas.

DESCRIPTION AND SCHEMATIC LAYOUT OF COMPONENTS OF VIDEO SURVEILLANCE SYSTEM

The camera system, like the 160B Sony Model 160B PTSC color camera equipped with a standard pan-and-tilt mechanism. The lens provides a focal length of 10 to 1 from a focal length of 35 to 70 mm. The camera was designed for short-term surveillance applications and therefore, was enclosed in a FibreDev NEMA-12 cabinet for protection from the elements. The camera lens and housing are mounted to a pedestal and tilt axis. The pedestal and tilt axis are mounted on a utility trailer to increase the height and visibility of the camera and the mobility of the equipment. The trailer is electrically and physically protected for the cameras and the electric equipment. The trailer is electrically and physically protected for the cameras and the electric equipment. The trailer is electrically and physically protected for the cameras and the electric equipment.

Figure 3.7: The Graphic Interface

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Ground Truth Data Collection

The collection of ground truth data is done semi-automatically. The “MouseControl” class is written for user to be able to click any area on the loaded image page. Thus it is helpful to select any string in this page. If the string is being processed by the OCR engine, which means it is one of the leaves in the hierarchical tree, then this string’s id and geometry information are the outputs of OCR engine and saved in XML file. When the page is loaded, this information is parsed by Electric XML Parser in “FileParser.java” class, and stored in the global array. If the (x, y) coordinate of the image where the mouse button is pressed is within the range of this string’s, it means this string is chosen by user.

In the data collection part, we use three mouse buttons. The left mouse button is used to select the individual string, the right mouse button is to finish the string’s selection, it also means the whole sentence of these strings is constructed, and the middle mouse button is to finish the data collection on this whole image page.

We uses several JOption popup windows for each mouse button. These popup windows are used to let the user confirm the information he just selected, or choose some information which will be used later on. After the user selects any one string by clicking the left mouse button, a “confirmation” dialog window of JOptionPane would
INTRODUCTION

Visual monitoring of Mount St. Helens became critical because of the threat of explosive eruptions. In 1980, the U.S. Geological Survey and the U.S. Forest Service began monitoring the volcano. The TV surveillance system was installed to allow visual monitoring of the volcano by personnel at nearby observation posts. A system was selected, assembled, and installed near the volcano by July 15, and became fully operational on July 20.

ADVANTAGES OF THE TV SURVEILLANCE SYSTEM

In addition to those mentioned above, some of the advantages are:

1. Eruption events can be observed as they occur, and their potential dangers can be immediately assessed.
2. The volcano can be examined any time during daylight hours, even during periods of cloudy weather (as could not have occurred previously).
3. The video system presents events at the volcano (e.g., volcanic eruptions, lahars) to be correlated with other monitoring data received by telemetry at the Emergency Coordination Center in Vancouver.
4. Weather conditions can be viewed each morning prior to planning the day's activities and near the volcano.
5. The video system can be recorded on magnetic tape to provide a record of events for subsequent research.

ACKNOWLEDGMENTS

We thank Gary J. Gunkelhof, Reynolds Parcels, and Engineering Co., Inc. for providing the equipment and schematic layouts of the video surveillance system.
THE SOUTHWEST RISK OF MOUNT ST. HELENS, WASHINGTON

VOLCANO MONITORING BY CLOSED-CIRCUIT TELEVISION

By C. Dan Miller and Richard Hoblit

INTRODUCTION

Closed monitoring of Mount St. Helens was successful in 1980. The monitoring efforts led to a sequence of events that resulted in the decision to resettle the town of 119. In May 1980, a Computer Science program at the University of Washington was first monitored using a single camera. The program was then expanded to include two cameras monitoring the volcano. The computer program monitored the activity of the volcano and transmitted the images to a computer center located in Seattle.

Monitoring the volcano was made possible by the installation of a closed-circuit television system. The system was installed after the evacuation of the town of 119. The system was then monitored and transmitted to the computer center in Seattle.

ACKNOWLEDGMENTS

We wish to thank S. T. and J. B. for their generous support of the volcano monitoring project.
pop up, asking the user to make a confirmation (see Figure 3.11). After receiving a “Yes” answer, this string is stored to an array. The user keeps selecting strings in this way. After he is done with string collection, he clicks the mouse right button, and another option dialog window of JOptionPane pops up, asking him to confirm that he has finished the selection of strings (see Figure 3.12), because the user might click the wrong button during selection. Once the user gets the confirmation, an option window pops up, asking the user to select a tag among: title, author, abstract, subtitle, header, footer, content (see Figure 3.13). After the user chooses the tag name, a confirm dialog window pops up again, showing the whole sentence consisting of all strings selected for that tag (see Figure 3.14). If the user confirms that he has collected everything he wants, that information will be stored to a XML file, or database for future reference. Then the user may start to collect another tag’s contents with the mouse click. After he finishes collecting the whole page’s information, he clicks the middle button, and a confirm dialog window pops up, asking him whether he wants to continue collecting on the next page (see Figure 3.15). If he says “no”, this means he has finished this document’s collection, and will either do another document or exit the interface.

Since this interface loads the scanned TIFF image, this image file is saved as an OCR output in XML format. Before loading on the interface, this XML file is processed by EXML Parser, and the original information of data are already saved in the “FileParser.java” file. On the interface, the string “text” and its “id” number of string chosen by the user are actually returned from the parsing information already saved before, therefore, the information of that selected string is the original data value of OCR output. It might contain the OCR errors, which will be described in Chapter 4, but absolutely, it won’t contain any human error, such as typing mistakes. Besides, it saves the user much time from keyboard entry. In short, this way to collect data is less error prone, and much faster than typing data in through the keyboard.
Output File

The “xmlFile.java” class deals with the output of data collection. In this part’s design, we use several “static” methods (such as “setTitle()”, “setAuthor()”, “setHeader()”, “setFooter()”, etc.) to set the tags chosen by the user. As shown on Figure 3.13 and 3.14, the user selects the tag, and confirms the whole sentence is correct, then this string is passed into this tag’s set method in “xmlFile” class. The result is separated to be printed out as two parts: “print_xmlBody()” prints the XML file’s body, which contains all tags and their contents selected on a certain page, and “print_xmlEnd()” to print the end tag, which contains the close document tag. There are two methods to print the output, one is directly to print on the screen, and the other is to write the output into a file.

After the user finishes collecting all data on the certain page, he clicks the middle mouse button. Then he confirms on the popup window (see Figure 3.15) whether he wants to continue collecting on the next page or not. If he wants to continue, only print_xmlBody() method is called. the current information on this page is printed on the screen and also sent to the output file. The result however has no closing document tag, which means it does not end yet. When the user does not continue to the next page’s collecting, he clicks “NO” on the popup window, then print_xmlEnd() method is called right after print_xmlBody() method, which ends the whole document’s collection work with a closing document tag.

In this chapter the image based interactive tools for entering ground truth data are built with three main parts: processing the input document, building and manipulating the graphical user interface, and writing the results to an output file. In the next chapter, the advantageous results of this research will be discussed.
Figure 3.11: String Collection

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 3.12: Finish the Selection of One Sentence

Figure 3.13: Tag Selection
Figure 3.14: The Ground Truth Data
Figure 3.15: Finishing the Collection on One Page
Volcano Monitoring by Closed-Circuit Television

C. Dan Miller Richard P. Hoblitt

Visual monitoring of Mount St. Helens volcano by closed-circuit television allows eruptive events to be observed as they occur and an immediate evaluation of potential hazards to be made. Use of the remotely controlled TV system also reduces risks to personnel during eruptions by eliminating the need for close-in observers on the north side of the volcano, and reduces the need for continuous observation from aircraft.

Figure 3.16: Example of An Output File
CHAPTER 4

RESULTS AND DISCUSSION

The result of the image based interactive tools for entering ground truth data is shown in the output file in Chapter 3. It not only provides a quick and precise way to collect the ground truth data, but also a possible method to prevent several potential problems due to OCR output, such as: separated words, misrecognized words, non-consecutive ids of consecutive strings, etc.

- Unexpectedly Separated Words

After it is scanned by OCR devices, one string may be separated into two parts. For example, the string “of” may be split into two strings “o” and “f” by an OCR engine, and it won’t be detected by the “Autotag” system, which will still take it as two separated words in its logical representation. Thus it might cause problems for future information retrieval.

Image based interactive tools offer an easy and direct way to collect ground truth data. They display the data in its original value on the interface as it was presented in the pdoc file after OCR scanning. Therefore, in this research this kind of error will be detected easily and directly on the interface. Once the TIFF image of the document’s physical representation is displayed on the interface, the user may click the “addBox” button to display all bounding boxes of strings. If “of” has two bounding boxes instead of one, then it is known that “of” is separated. This kind of error can also be corrected by entering the ground truth data to an output file: through selecting these two strings successively, their texts and ids are stored together as one part of the logically structured
information, thus no matter how many parts one word may be separated by OCR errors, it will be constructed again as one logical word in the output file.

- **Misrecognized Words**

  As discussed before, this way of collection will not introduce any human typing errors. What's more, it may detect some OCR errors during collection. OCR errors are usually generated during OCR scanning, since OCR techniques are not free of errors. OCR error rates are highly variable based on the quality of images, font types, etc. There may be many typical OCR errors such as: "c" to "e", "i" to "l", etc., and some "random" misrecognitions as well in the pdoc files. One example of such misrecognized error is shown in Figure 4.1.
Figure 4.1: The Example Of A Misrecognized OCR Error

The string “Management” is misrecognized as “M&najtement”. There is no relationship between these two words. The errors are “random” errors generated from OCR scanning, not a “typical” OCR error, thus it is very difficult to detect it with many automatic OCR-error correction systems, such as the “PPSY” system in the “Manicure” system, which is introduced in Chapter 1. In most cases, it is not necessary to detect and correct it, as long as it won’t have big effects on the information retrieval.

In this research, if this kind of error exists in the part of important data that construct the logically structured information, such as in the title, author, abstract, etc. then it can be easily detected with the user semi-automatical selection on the interface. Since this research focuses on the “ground truth data”, there is no need to correct it. It provides a possible way to detect this kind of “random” misspelling errors for future’s correction.

- Ids of Consecutive Strings Are Not In the Consecutive Order
In most cases of the document's physical representation, the consecutive strings are in the consecutive order based on ID number. But there also exist some situations that after OCR scanning, the consecutive strings have the non-consecutive ID orders in the input document. They are not in the order shown on the image, or in the order that the user read the file.

If constructing the logically structured information of those "no order" string ids from the keyboard entry, one problem would happen: where to find its successive string in the whole image page? With the collection tools of image based interface, this problem is solved easily. Since the image is displayed on the interface, the user just needs to choose the next string with the mouse clicking, and after further confirmation, store this string's id and its text right after its previous string in one piece of logically structured information. The non-consecutive string id does not have any bad effect on the collection as it would in the keyboard entry, and there is no time to waste in order to search the consecutive string in the whole image file.

Usually, there are three cases of non-consecutive string ids that the user may meet in the image file are as follows:

- **One image page**

  The ids of consecutive strings appear in the non-consecutive order on one image file. The Figure 4.2 shows a typical example of one scanned image page with such problem. On the first page of document "mol20010419.0251", the header's string "id" is "1", but the id of "Department" is "421", right after the last second string "2000" on that page, whose id is "420". The footer is the last string on that page, but its string id "443" is not right after "2000", but after the zip code on the top of page (as shown in Figure 4.3, 4.4, 4.5, 4.6).
- One page is separated in the middle

The page is separated in the middle. The OCR scanned output may have put these paragraphs in the different zones, which will cause the string ids to be non-consecutive. In the Figure 4.7 and 4.8, the last string id on the left column is “753”, while its consecutive string, the first string on the right column has the id “788”.

- Consecutive strings on two consecutive pages

Since some pages have footers and/or headers, the string ids of the last word on the previous page and the first word on the next page may not be consecutive.

In this chapter, several potential problems existing in the image file result from OCR scanning technique are discussed, and it shows that the image based interactive tools provide the better and easier way for entering the ground truth data in an output file.
Figure 4.2: The Example Of One TIFF Image
Figure 4.3: The String ID of "2000"

Figure 4.4: The String ID of "Department"

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 4.5: The String ID of Zip Code

Figure 4.6: The String ID of Footer

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
surveillance system replaces one daily, early morning
USFS observer flight and, during periods of poor
or marginal weather, some other flights as well.
The system is used regularly to examine fluctua-
tions in gas emission, avalanche frequency, local
wind conditions blowing ash in potential work areas,
and eruptive events. The system is also useful for
briefing scientists and public officials without vis-
ting the volcano.

When combined with seismic, tilt, and other real-
time monitoring data delivered into Vancouver,
the video system allows USFS and USGS personnel
at ECC to make accurate and rapid assessments of the
state of the volcano and associated hazards.

Figure 4.7: The Last String ID on Separated Paragraph
Surveillance system replaced one daily, early morning USFS observation flight and, during periods of poor or marginal weather, some other flights as well. The system is used regularly to examine fluctuations in gas emission, avalanche frequency, local wind conditions blowing ash in potential work areas, and eruptive events. The system is also useful for briefing scientists and public officials without visiting the volcano.

When combined with seismic, rth, and other real-time monitoring, data telecentered into Vancouver, the video system allows USFS and USGS personnel at ECC to make accurate and rapid assessments of the state of the volcano and associated hazards.

Figure 4.8: The First String of Separated Paragraph
CHAPTER 5

FUTURE WORK

This research is aimed at preparation of the ground truth data (GTD) from the interface. So even if there are some OCR errors in the GTD, it is not necessary to correct them. As discussed in Chapter 4, this interface provides a good method to detect the OCR errors in collected logically structured information, which make it quite easy to correct any possible OCR error in sentences.

In future work, the tools of this research may be used for the correction of OCR errors: after the user select any string, the confirmation window pops up. If “Yes” button is clicked, it means that the string is correct and it is stored to the output file. If “NO” button is clicked, it means that there are some errors in this string. An input window of “showInputDialog” method of “JOption” class would then pop up, asking the user to type in the correct string, and the string will be stored into the output file, instead of the misspelled string. As to the former example in Figure 4.1, the “Manager” is misspelled to “M&najement” after being scanned from OCR device. The Figure 5.1 shows the example for the correction of this OCR error.

In future work, this system can be connected to a database, entering data directly to the database using JDBC. Besides, the interface can also be improved according to the user’s requirement in future, such as: adding some other functions, like “scale” to load images of various size.
Figure 5.1: Use Input String to Correct an OCR Error
BIBLIOGRAPHY


VITA

Graduate College
University of Nevada, Las Vegas

Jun Liu

Local Address:
1600 E. University Ave #125
Las Vegas, NV 89119

Home Address:
3025 Via Della Amore
Henderson, NV 89012

Degrees:
Master of Science, Chemistry, 2000
University of Nevada, Las Vegas

Thesis Title:
Image Based Interactive Tools For Entering Ground Truth Data

Thesis Examination Committee:
Chairperson. Kazem Taghva. Ph. D.
Committee Member. Dr. Thomas Nartker. Ph. D.
Committee Member. Dr. Wolfgang Bein. Ph. D.
Graduate Faculty Representative. Dr. Klaus J. Stetzenbach. Ph. D.