PR 1726
pp: 1--15 (col.fig.: Nil)
PROD. TYPE: COM
ED: Indira
PAGN: Vidya – SCAN: Vinod

ARTICLE IN PRESS

PERGAMON

Pattern Recognition ▮▮▮ (▮▮▮) ▮▮▮–▮▮▮

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

The architecture of TrueViz: a groundTRUth/metadata editing and VIsualiZing ToolKit

Chang Ha Lee[a], Tapas Kanungo[b,*]

[a] *Department of Computer Science, University of Maryland, College Park, MD 20740, USA*
[b] *IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120, USA*

Received 5 December 2001; accepted 20 March 2002

Abstract

Tools for visualizing and creating groundtruth and metadata are crucial for document image analysis research. In this paper we describe TrueViz (TRUEVIZ User's Manual, August 2000; Proceedings of the SPIE Conference on Document Recognition and Retrieval, San Jose, CA, 2001, pp. 1–12), which is a tool for visualizing and editing groundtruth/metadata. We first describe the groundtruthing task and the requirements for any interactive groundtruthing tool. Next we describe the system design of TrueViz and discuss how a user can use it to create groundtruth. TrueViz is implemented in the Java programming language and works on various platforms including Windows and Unix. TrueViz reads and stores groundtruth/metadata in XML format, and reads a corresponding image stored in TIFF image file format. Multilingual text editing, display, and search modules based on the Unicode representation for text are also provided. This software is being made available free of charge to researchers.

Keywords: Annotation; Groundtruth; Visualization; Multilingual; Multiplatform; Java; XML; OCR

1. Introduction

In the document image analysis (DIA) research area, the term 'groundtruth' refers to various attributes associated with the text on the image—bounding box coordinates of words, lines, characters; font type; character size; direction of text; etc. Groundtruth data is crucial for document image analysis because it is impossible to train and test optical character recognition (OCR) algorithms without it. Since groundtruth is created manually in most cases, tools for annotating and visualizing groundtruth are very important. In fact, at the MLOCR99 international workshop [1–3] the consensus in the corpus working group was that our community needs (i) a protocol for groundtruthing documents, (ii) an XML-based groundtruth representation format, (iii) a public-domain multilingual/multiplatform visualization and data-entry tool, and (iv) a consortium for managing and distributing datasets.

In this paper we address two of the four issues raised by the working group: (i) we describe an XML-based groundtruth representation format, and (ii) we describe TrueViz, which is a public domain [1] annotation tool that we have developed at the University of Maryland.

This paper is organized as follows. In Section 2 we describe various existing annotation tools used in document image analysis and in related areas such as speech recognition, linguistics, and information retrieval. The desirable features of a document image groundtruthing tool are described in Section 3. In Section 4 we discuss design and implementation issues related to editing, visualization, and search. The XML data format for groundtruth is discussed in Section 5, where we also provide representative samples of XML files. The multilingual data entry, visualization, and

* Corresponding author. Tel.: +1-408-927-2487; fax: +1-408-927-3215.
E-mail address: kanungo@almaden.ibm.com (C. Ha Lee).

[1] TrueViz is available at http://www.cfar.umd.edu/~kanungo/software/software.html.

0031-3203/02/$22.00 © 2002 Published by Elsevier Science Ltd on behalf of Pattern Recognition Society.
PII: S0031-3203(02)00101-2

search features of TrueViz are quite unique and are discussed in Section 6. Finally, in Section 7 we list the things that we hope the international DIA community will add to the public domain system.

2. Previous work

There are many annotation and visualization tools in various domains. In this section we describe a few annotation tools commonly used in document image analysis, speech recognition, linguistics, information retrieval, video analysis, geographic systems, and statistics. In Table 1 we provide a comparison of these tools.

2.1. Document image visualization tools

Visualization tools for displaying or editing a document image and groundtruth metadata have been developed for evaluating algorithms, creating document groundtruth, or browsing documents.

Pink Panther [4] is an environment for creating segmentation groundtruth files and for page segmentation benchmarking. Page segmentation is the process of decomposing a document page image into structural and logical units, such as images, paragraphs, headlines, tables, etc. The performance of a page segmentation algorithm is evaluated by running the algorithm on a set of document images, and comparing the output for each document to corresponding groundtruth metadata. Pink Panther consists of two parts: Grounds-Keeper and Cluzo. Grounds-Keeper is a tool for creating groundtruth metadata. It visualizes a document image and the corresponding metadata, and also allows users to zone the document image and specify the information for each zone. Groundtruth metadata created by Grounds-Keeper is stored in an ASCII file format. Cluzo is a benchmarking tool for collecting the locations, types and severities of segmentation errors on a page as well as information on segmentation performance. Pink Panther is implemented on the Unix and X Windows platforms and is written in C. While Grounds-Keeper allows the user to enter segmentation groundtruth, entering text groundtruth is not possible.

Illuminator [5] is an editor developed by RAF Technology, Inc. for building document understanding test and training sets, for correction of optical character recognition (OCR) errors, and for reverse-encoding the essential information and attributes of a document. Illuminator visualizes or edits a document image and its entities, which are specific regions of the image and the associated metadata. It is configured to handle text in major European languages and Japanese. Illuminator uses the document attribute format specification (DAFS) file format [5] to store the document image and metadata. DAFS provides a format for breaking down a document into entities which have hierarchical structure, and for defining entity boundaries and attributes.

Table 1
Comparison of visualization tools

Name	Platform	Data format	Domain
PinkPanther	Unix/X Windows system	ASCII	Document image groundtruth
Illuminator	Unix/X Windows system	DAFS	Document image groundtruth
Oulu Database Browser	Multi-Platform/Java	ASCII	Document image groundtruth
TrueViz	Multi-Platform/Java	XML format	Document image groundtruth
Transcriber	Unix/Windows NT	XML format	Speech annotation
ATLAS	Unix/Windows NT	XML format	Linguistic annotation
Alembic Workbench	Unix system	SGML/PTF format	Linguistic/named entities annotation
ViPER	Multi-Platform/Java	ASCII	Video sequence groundtruth
XGobi	Unix/X Windows system	S data format/ASCII	Statistical data
S-PLUS	Windows 95/98	Customized data	Statistical data
CLASP	Unix/Macintosh	Commonly used formats	Statistical data
Mondrian	Multi-Platform/Java	ASCII/databases	Categorical/geographical data
PolyPaint+	SunOS/Solaris	netCDF	Geographical data
Spotfire	MS Windows	Database/spreadsheet/ASCII	Decision making by data analysis
Slicer Dicer	MS Windows	Binary/ASCII/Commonly used formats	Medical/scientific data defined on grids

Illuminator is implemented on the Unix and X Windows platforms and is written in C.

The MediaTeam Oulu Document Database [6] is a collection of scanned documents with corresponding groundtruth for the physical and logical structure of the documents. It was developed by the University of Oulu MediaTeam. The document database browser is a visualization tool for exploring the contents of the database. The browser is written in the Java programming language and allows visualization of document images and corresponding metadata simultaneously. The browser can explore the database and select particular documents for visualization. The browser also provides a window to list attributes of the document. Document images which were originally stored in TIFF image format are stored in JPEG image format and metadata is stored in an ASCII file format.

Pink Panther and Illuminator work only on the Unix platform. Because there are many tools that are executable only on the Windows platform, this is a limitation. The Oulu document database browser is written in the Java programming language, and can be run on various platforms.

However, the Oulu document database supports JPEG image format only, while TIFF is the most popular image format for document images. Furthermore, the file representation of the groundtruth is non-standard. In fact, all the above tools store document metadata in their own file formats. To provide data compatibility, a standard file format, or a file format to which other file formats can be easily converted, is needed.

A prototype system for visualizing and editing groundtruth is currently being built at the University of Fribourg, Switzerland [7]. This system allows users to edit the hierarchical structure of the document. However, the system does not provide a compatible OCR evaluation package to visualize OCR segmentation results.

2.2. *Other visualization tools*

We surveyed visualization tools in other data domains to find out the best way to provide multi-platform and data compatibility. In this section we summarize features of visualization tools in various domains such as statistical, categorical, geographical, and medical data as well as linguistic data and speech signals.

Transcriber [8–10] is a tool for segmenting, labeling, and transcribing speech signals. It supports most common audio formats and stores the transcription in XML format. It was developed in the Tcl/Tk and C programming languages, and works on Unix and Windows NT platforms.

ATLAS [11] is an architecture and tool for linguistic analysis based on a formal model for annotating linguistic artifacts. It uses an XML-based ATLAS interchange format (AIF) for storing annotated corpora, and was developed in the C++, Perl, Tcl/Tk, and Java programming languages.

Alembic Workbench [12] is a new set of integrated tools that uses a mixed-initiative approach to bootstrapping the manual tagging process with the goal of reducing the overhead associated with corpus development. The Alembic Workbench is developed using the Tcl/Tk, Perl, C, and Lisp programming languages, and works on the Unix platform. Alembic uses the SGML and parallel tag file (PTF) formats for source text and annotations.

Video processing evaluation resource (ViPER) [13] consists of three main components: ViPER-GT, ViPER-PE, and ViPER-Viz. ViPER-GT contains modules for configuring and producing groundtruth information which describes a video sequence. The ViPER-PE module provides performance evaluation capabilities for comparing computed results with appropriate groundtruth information. ViPER-Viz enables a user to visualize groundtruth, analysis results, performance evaluation results, or an entire video clip. ViPER was developed in the Java programming language, and groundtruth and results are stored in ASCII file format.

XGobi [14–16] is an X Window application for interactively exploring statistical data. Its current functionalities include brushing, identification, and editing of connected lines, as well as rotation and the grand tour, with several interactive projection pursuit indices. Several functions can be linked so that actions in one window are promptly reflected in another.

S-PLUS [17] is a desktop data analysis tool that provides data analysis and visualization capabilities to identify trends in data. It allows data import and export from spreadsheets such as Excel, as well as from a wide range of relational and other data sources.

The common lisp analytical statistics package (CLASP) [18] is a tool for visualizing and statistically analyzing data. CLASP provides an interactive environment for data manipulation and statistical analysis and a variety of descriptive and hypothesis-testing statistics. It includes many features that facilitate exploratory data analysis.

Mondrian [19] is a data-visualization system written in Java. Its main emphasis is on visualization techniques for categorical data and geographical data. Mondrian provides various plots such as mosaic plots, maps, barcharts, and parallel coordinates, which are fully linked and allow various interrogations.

PolyPaint+[20] is an interactive scientific visualization tool that displays complex structures within three-dimensional data fields. It provides color shaded-surface display, as well as simple volumetric rendering in either index or true color. PolyPaint+ routines first compute the polygon set that describes a desired surface within the 3D data volume, and these polygons are then rendered as continuously shaded surfaces. Objects rendered volumetrically may be viewed along with shaded surfaces. Additional data sets can be overlaid on shaded surfaces by color coding the data according to a specified color map.

Spotfire [21] is a decision analysis workspace that uses the connectivity of the Web to provide a workspace in which to access large amounts of complex data from wherever it resides, to visually explore and analyze the data, and to share results.

Slicer Dicer [22] provides tools for analysis, interpretation, and documentation of complex data defined in three or more dimensions. It helps in exploring the data visually by "slicing and dicing" to create arbitrary orthogonal and oblique slices, rectilinear blocks and cutouts, isosurfaces, and projected volumes. It also provides animation sequences featuring continuous rotation, moving slices, blocks, parametric variation (time animation), oblique slice rotation, and varying transparency.

A more detailed review and taxonomy of visualization tools can be found in an article by Shneiderman [23], and a good general reference for user interfaces is Shneiderman's book [24].

3. Desired GUI functionalities

Since TrueViz will be used by different researchers for different tasks, we first summarize the functionalities that are desired of such a tool. The simplest task that the tool could be used for is to visualize and input multilingual text. Next, it could be used to mark regions of a scanned document image as text or graphics, and assign labels to regions. A researcher wanting to look at the results obtained by a DIA system might want to search for all the incorrectly recognized characters and then zoom into the image at those locations. A researcher interested in extracting the logical structure of a document might want to label the reading order of the text areas, or the hierarchy of the text regions corresponding to sections and subsections.

After studying the various tasks for which a user might want to use the to-be-designed tool, we formulated the following set of requirements for the graphical user interface:

Entities: Users should be able to visualize and edit zone-, line-, word-, and character-level geometric groundtruth. Furthermore, they should be able to establish their own entity structure. For each entity, they should be able to define attributes (e.g. bounding boxes) and specify their values.

Scale: Users should be able to zoom in and out of the image and overlaid groundtruth so that they can study the image and OCR error results at the page, paragraph, line, word, or character level.

Color: It should be possible to display entities that have different attributes in different colors. For example, image zones could be shown in one color and table or text zones in another. Thus if a DIA system incorrectly recognizes a table zone as an image zone, the error would be easily identifiable from the color coding.

Logical information: The visualization tool should allow users to visualize and edit the logical reading order of text zones, and also to specify the hierarchy of the text zones. For example, it should be possible to visually specify that a subsection is contained in a section.

Multilingual visualization: Since DIA systems are being developed for various languages and scripts, users should be able to visualize groundtruth text in these languages and scripts. The use of a standard encoding such as Unicode is highly desirable.

Multilingual data entry: While regular English text can be entered by regular keyboards, keyboard mappings that allow other languages and scripts to be entered should also be available.

XML-based representation: The XML markup language would be ideal for representing page layout groundtruth since it is the current industry standard and various parsers, syntax checkers and editors are publicly available for it.

Converters: Converters to convert standard datasets such as the University of Washington dataset (in DAFS format) into the XML representation would help bootstrap research by providing seed datasets.

Search: Users should be able to search for strings in the groundtruth and find the locations where they appear in the image. The search module should work in any language and users should be able to specify edit distances for approximate searching, which is essential when searching for strings in noisy OCR text.

Evaluation: The tool should have a built-in OCR evaluation module or should be compatible with one, so that users are able to visualize OCR evaluation results easily.

Multiplatform: Since researchers and data entry persons work on various platforms such as UNIX, PC, and Mac, the tool should be platform-independent so that users need not spend time learning how to use it on a platform that they are not familiar with.

Public domain: In order for the community to take full advantage of it, the tool should be freely available.

4. Design and implementation

4.1. Overview

The TrueViz display is vertically split into two panels (see Fig. 1). The left panel is an image panel for displaying a document image and corresponding geometric metadata, and the right panel is a tree view for displaying textual metadata structure.

The image panel displays a document image and overlays geometric metadata on the image. Currently, three kinds of geometric metadata can be visualized: bounding boxes, logical relationships, and an Infopanel. The bounding box of an entity is visualized as a polygon whose color represents the type of the entity. "Logical relationship" refers to logical reading order, and is visualized using an arrow from one entity to the next. The Infopanel is a small window for displaying a few important attributes of the entity. The image and metadata visualization can be scaled to various resolutions.

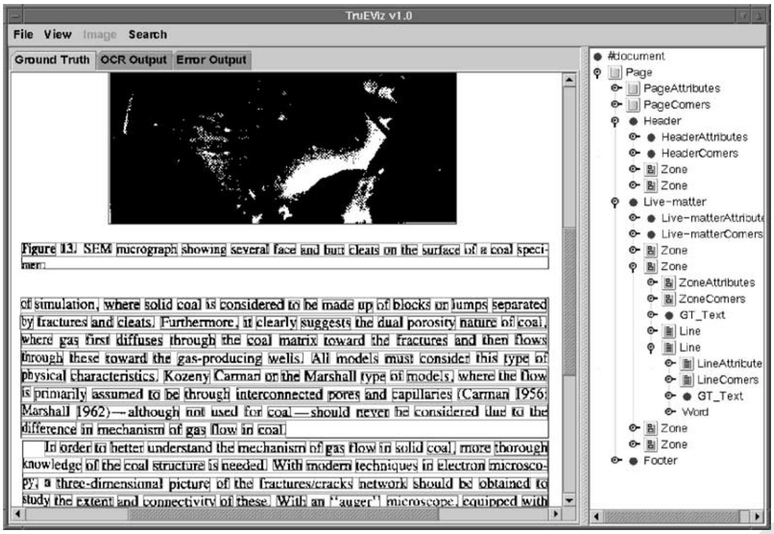


Fig. 1. TrueViz consists of an image panel (left) and a tree view (right).

The tree view displays the XML-based groundtruth metadata in a tree structure of expandable and collapsible nodes. The attribute values can be edited in the tree nodes and the groundtruth text can be edited in the separate multilingual text editor.

4.1.1. Metadata visualization

Entities can be classified into four categories: Zones, Lines, Words, and Characters. Entities are hierarchical in nature, so a Zone is contained within a Page, a Line is contained within a Zone, a Word is contained within a Line, and a Character is contained within a Word. Because of the hierarchical nature of the entities, it is necessary to change views in order to view specific portions of the structure. There are five views: Image Only, Page, Zone, Line, Word, and Character. The Image Only view shows only the image without any groundtruth visualization. The Page view shows metadata for all entities, from the highest level to the lowest level. This view is not editable or selectable. The Zone view shows only Zone metadata. A Zone's data can be accessed by clicking on the Zone. This causes the Zone to be active (selected) and highlighted, and the Infopanel to pop up. The Infopanel is a small window for displaying important metadata for the active entity (see Fig. 10). The corresponding node in the tree view will also be selected. Similarly, the Line view shows all Line metadata (see Fig. 2(a)), the Word view shows all Word metadata (see Fig. 2(b)), and the Character view shows all Character metadata. As in the Zone view, metadata can be selected, and the Infopanel for the active entity is popped up.

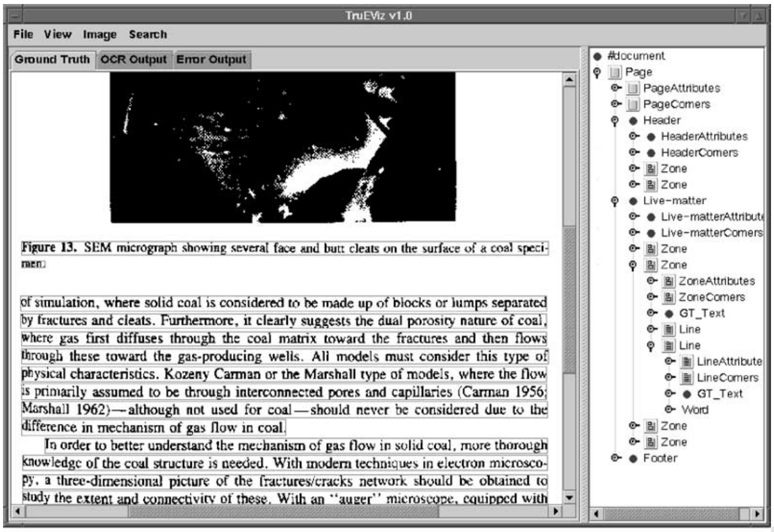
There are two options for views: 'Fill Bounding Boxes' and 'Logical Relations'. If the 'Fill Bounding Boxes' option is checked, all entities are painted in colors corresponding to their types (see Fig. 3(a)). Otherwise, entities are displayed using polygonal outlines whose colors also represent their types. This option is useful when the document is displayed at a large scale, because a user can see the type of an entity from its color even if the bounding box is too large to fit on the screen. If the 'Show Logical Relations' option is selected, the logical reading order relations are visualized using arrows from each entity to the next logical entity (see Fig. 3(b)).

4.1.2. Metadata editing

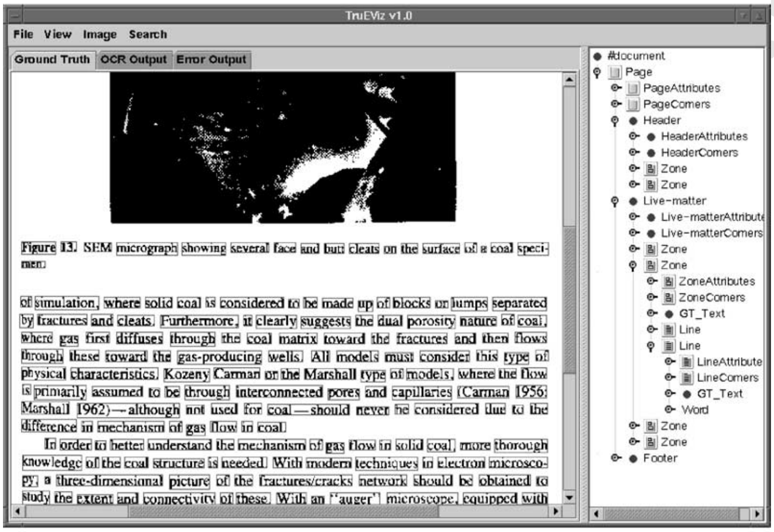
Groundtruth metadata can be edited in two ways: graphical editing and text editing. All metadata can be edited within the attribute value node in the tree view. Because the groundtruth text may contain multilingual text, it is edited in the separate multilingual text editor. The metadata visualized in the image panel can also be edited graphically. It is very difficult to correct bounding boxes of entities by editing their coordinates. Therefore, TrueViz enables users to change the coordinates of bounding boxes graphically. In addition to the bounding boxes, the logical relationships can be changed graphically. The image panel can also be used to create and delete entities.

4.2. Search

TrueViz provides a multilingual approximate search functionality. A search string and edit distance can be specified in the search window. TrueViz provides multilingual input for a search string. The edit distance is the minimum number of substitutions, insertions, and deletions required to transform one string into another. The maximum edit distance allowed during the search can be specified [25]. After the search is finished, all entities containing the search



(a)



(b)

Fig. 2. Hierarchical display. (a) Line view displays all Line entities. (b) Word view displays all Word entities.

string within the specified edit distance are highlighted (see Fig. 4).

5. The data format

5.1. Overview

Groundtruth metadata is stored in XML file format [26–29] (see Fig. 5), and document images are stored in TIFF image file format. The tree view reflects the XML data file, and an internal data structure is created to visualize the groundtruth metadata. The internal data structure consists of region of interest (ROI) nodes. An ROI is a generic term used to describe any area of the image that the user deems of interest. The internal data forms a directed acyclic graph with ROIs as nodes and hierarchical or logical links as edges.

5.2. XML data format

The groundtruth data is organized in a hierarchical structure. The highest-level and therefore most inclusive entity is the Document. A Document is, in its simplest form, a

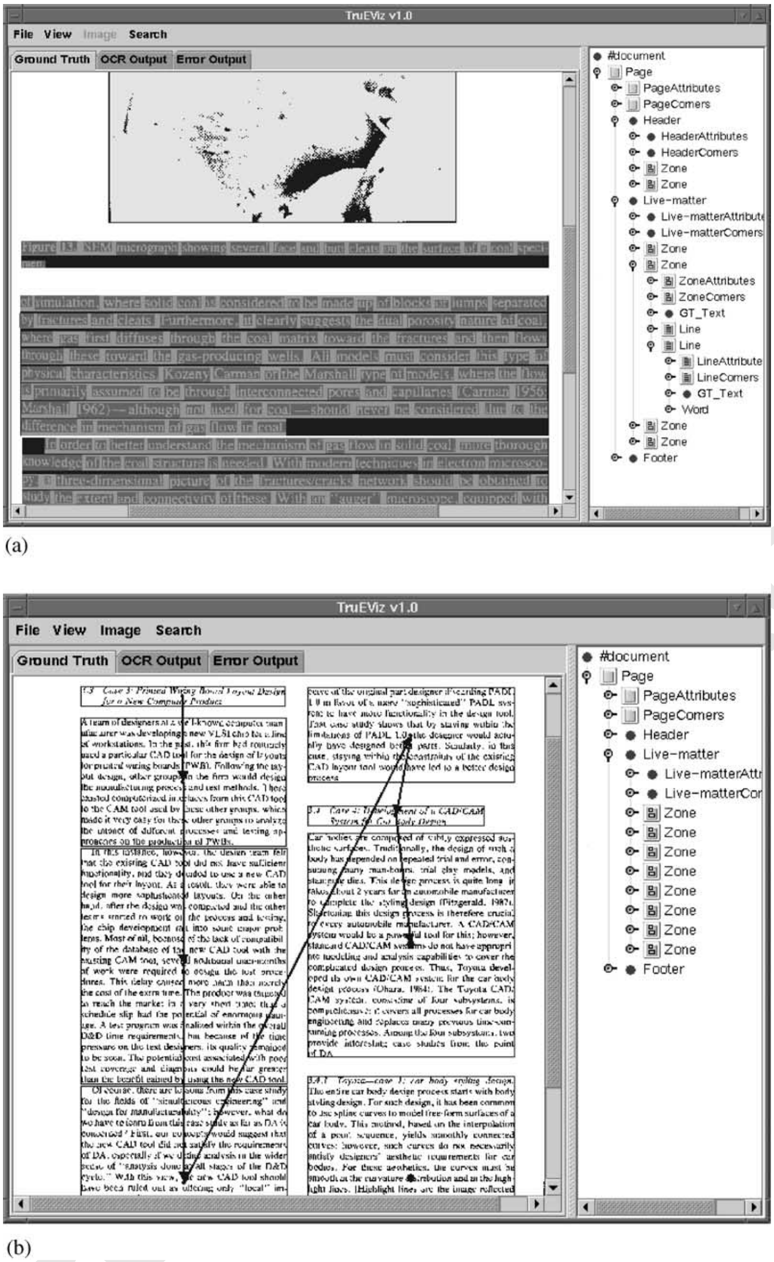


Fig. 3. View options. (a) Fill bounding boxes. (b) Logical relations.

collection of individual units, known as Pages, which are related to or support a specific topic or purpose (e.g. a report or manual). A Page is the next level down in the hierarchy and represents individual units of a Document. Each Page has an associated image that represents the original hard copy. A Page contains one or more Zones. A Zone is usually a rectangular area definable by its horizontal and vertical coordinates within a page. The purpose of a Zone is to identify a key area of the page such as title, heading, graphic, page number, etc. Each Zone may contain one or more Lines. A Line is an individual line of text. A Line can be broken down into one or more Words, each of which may contain one or more Characters. Each tag in the XML file represents an entity or attribute. An entity name can be any alphanumeric word, but the only entities that can be graphically edited in TrueViz are Zone, Line, Word, and Character.

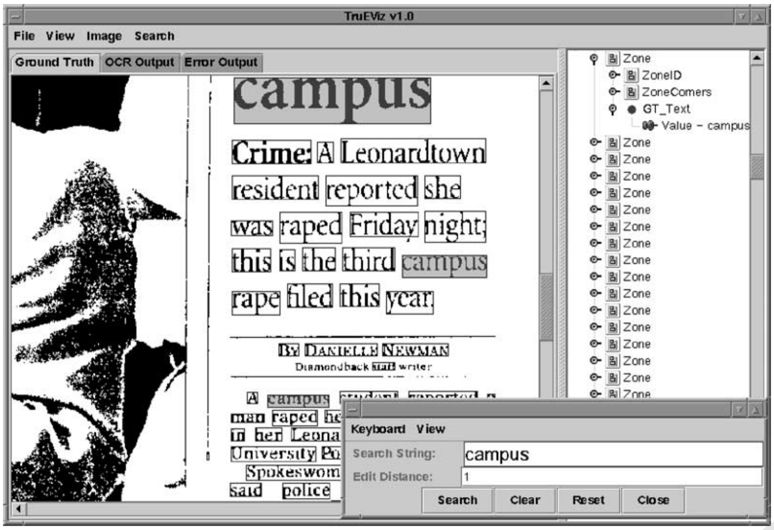


Fig. 4. Search string "campus" and edit distance 1 are specified in the search window (in the lower right corner), and the matching entities are highlighted.

An entity's attributes can be listed under the entity's tag in the XML file. While any attribute name can be listed, some built-in attributes are crucial for the visualization of groundtruth data.

ID: ID is the identification of the entity. The attribute name for ID is combined with the entity name. For example, the ID of a Zone entity is represented as ZoneID, and similarly we use LineID for Line, WordID for Word, and CharacterID for Character.

Corners: Corners represent the bounding box of the entity. The upper left, upper right, lower right, and lower left vertices are listed inside a Corners tag in order. Like the ID, the attribute name is combined with the entity name.

Next: Next stores the ID of the logically following entity. As with the ID, the attribute name is combined with the entity name.

GT_Text: GT_Text stores the groundtruth text of the entity.

An example of a simple entity is shown in Fig. 6.

5.3. Internal data structure

The groundtruth metadata is stored in XML file format, which is essentially a tree. The entities, on the other hand, form a directed acyclic graph structure. Each entity contains child entities and has a next logical entity. The graph representing the entity structure can be expressed by (see Fig. 7)

$$G = (V, E), \tag{1}$$

where $V = \{Zone, Word, Line, Character\}$, $E = \{contains, next\}$. Because of the difference between the entity structure and the XML structure, TrueViz has an internal data structure that is a little different from the XML structure. The internal data structure consists of ROI nodes, and the ROIs form a directed acyclic graph as described in Eq. (1). A next logical entity is stored as an attribute of an entity in the XML file, and is converted into a link from an ROI to the next ROI in the internal data structure.

For parsing XML files and converting XML structures into internal structures, Java application program interfaces (APIs) were used. Two kinds of Java APIs can be used for XML parsing: simple API for XML (SAX) and document object model (DOM) [29]. SAX is an event-based framework for parsing XML data. It reads through the XML document, breaks down the data into usable parts, and defines the events that occur at each step of the process. DOM provides a data representation of an XML document as a tree, which can be traversed and manipulated. A DOM parser was used in TrueViz because TrueViz has an internal data structure that needs to be kept in memory.

5.4. Flexible entity structure

Various entity hierarchies are used [30–32], depending on the type of document. Users may want to build document metadata using their own structures. TrueViz provides flexible entity structures so that users can build their own entity structures and document type definition (DTD) files for defining and verifying the structures of their XML files. The entity structure is extracted from the XML file, and the DTD file can be used to verify that the XML file conforms to the corresponding entity structure. The DTD file can be created and edited using any existing public domain editor.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Page SYSTEM "Trueviz.dtd">
<Page>
        <PageID Value="P000"> </PageID>
        <PageType Value="Journal"> </PageType>
        <PageNumber Value="1"> </PageNumber>
        <PageColumns Value="1"> </PageColumns>
        <Font Size="9-12" Spacing="Undefined" Style="Normal" Type="Serif"> </Font>
        <Zone>
                <ZoneID Value="Z000"/>
                <ZoneNext Value="Z001"/>
                <CharacterOrientation Type="String" Value="up-right"/>
                <DominantFontSize Type="String" Value="9-12"/>
                <DominantFontSpacing Type="String" Value="proportional"/>
                <DominantFontStyle Type="String" Value="plain"/>
                <DominantFontType Type="String" Value="serif"/>
                <Language Type="String" Value="English"/>
                <TextAlignment Type="String" Value="justified"/>
                <TextReadingDirection Type="String" Value="left-right"/>
                <ZoneCorners>
                        <Vertex x="1281" y="3136"></Vertex>
                        <Vertex x="1296" y="3136"></Vertex>
                        <Vertex x="1296" y="3169"></Vertex>
                        <Vertex x="1281" y="3169"></Vertex>
                </ZoneCorners>
                <GT_Text Value="a"></GT_Text>
                <Line>
                        <LineID Value="Z000L000"/>
                        <LineCorners>
                                <Vertex x="1281" y="3136"></Vertex>
                                <Vertex x="1296" y="3136"></Vertex>
                                <Vertex x="1296" y="3169"></Vertex>
                                <Vertex x="1281" y="3169"></Vertex>
                        </LineCorners>
                        <GT_Text Value="a"></GT_Text>
                        <Word>
                                <WordID Value="Z000L000W000"/>
                                <WordCorners>
                                        <Vertex x="1281" y="3136"></Vertex>
                                        <Vertex x="1296" y="3136"></Vertex>
                                        <Vertex x="1296" y="3169"></Vertex>
                                        <Vertex x="1281" y="3169"></Vertex>
                                </WordCorners>
                                <GT_Text Value="a"></GT_Text>
                                <Character>
                                        <CharacterID Value="Z000L000W000C000"/>
                                        <CharacterCorners>
                                                <Vertex x="1281" y="3136"></Vertex>
                                                <Vertex x="1296" y="3136"></Vertex>
                                                <Vertex x="1296" y="3169"></Vertex>
                                                <Vertex x="1281" y="3169"></Vertex>
                                        </CharacterCorners>
                                        <GT_Text Value="a"></GT_Text>
                                </Character>
                        </Word>
                </Line>
        </Zone>
        <Zone>
                <ZoneID Value="Z001"/>
                <ZoneNext Value=""/>
                <CharacterOrientation Type="String" Value="up-right"/>
                <DominantFontSize Type="String" Value="9-12"/>
                <DominantFontSpacing Type="String" Value="proportional"/>
                <DominantFontStyle Type="String" Value="italic"/>
                <DominantFontType Type="String" Value="serif"/>
                <Language Type="String" Value="English"/>
                <TextAlignment Type="String" Value="justified"/>
                <TextReadingDirection Type="String" Value="left-right"/>
                <ZoneCorners>
                        <Vertex x="2281" y="3136"></Vertex>
                        <Vertex x="2296" y="3136"></Vertex>
                        <Vertex x="2296" y="3169"></Vertex>
                        <Vertex x="2281" y="3169"></Vertex>
                </ZoneCorners>
                <GT_Text Value="b"></GT_Text>
        </Zone>
</Page>
```

Fig. 5. An example XML file.

If an element has an “Entity” attribute and its value is “True”, the element is recognized as an entity when the XML file is parsed. An entity structure for an XML file is automatically built by TrueViz from the recognized entities and their level information. An example XML file with a user-defined entity structure is shown in Fig. 8, and Fig. 9 is the entity structure extracted from the XML file. If there are no elements with the attribute “Entity”, the default entity structure (see Section 5.2) is used.

```
<Zone>
  <ZoneID Value="Z001"/>
  <ZoneNext Value="Z002"/>
  <GT_Text Value="Hello, world">
  <ZoneCorners>
    <Vertex x="10" y="10"/>
    <Vertex x="100" y="10"/>
    <Vertex x="100" y="30"/>
    <Vertex x="10" y="30"/>
  </ZoneCorners>
<Zone>
```

Fig. 6. An example of a simple entity structure.

6. Multilingual features

6.1. Multilingual text data

Java programs running on JDK1.1 or JDK1.2 can display any Unicode [33] character which can be rendered with

```
<Page>
  <Paragraph Entity="True">
    <Line Entity="True">
      <Word Entity="True">
        <Character Entity="True">
        </Character>
      </Word>
    </Line>
  </Paragraph>
  <Figure Entity="True">
  </Figure>
  <Table Entity="True">
  </Table>
</Page>
```

Fig. 7. Entity structure.

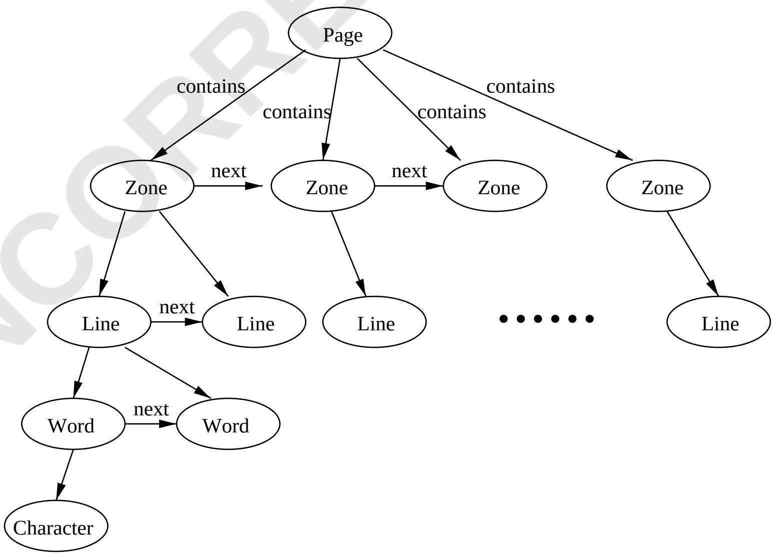


Fig. 8. An XML file with user-defined entity structure.

a host font. TrueViz displays multilingual text using Java Unicode facilities (see Fig. 10). TrueViz can read Unicode characters from the XML file, and saves the XML file in the Unicode UTF8 format [33]. However Java does not provide a multilingual input method. We therefore developed such a method, which is described in Section 6.2.

6.2. Multilingual input system

TrueViz provides a multilingual input system. Some languages like Chinese, Japanese, or Korean use more characters than can be input by a regular keyboard. To handle such languages, a sequence of several characters needs to be typed to construct a single character. While this composition process is going on, the input system accepts the sequence of characters, and produces composed text and committed text. The composed text is the intermediate text which is being processed to produce the intended text. The final text is called committed text (see Fig. 12). Input capabilities for various languages can be easily added using this common interface. In addition to the default English language input, Russian input is also currently implemented. The input system can be used anywhere multilingual text input is needed (see Fig. 11). For example, TrueViz supports multilingual text input in the search window for multilingual search. For people who are not familiar with the keyboard mapping, TrueViz provides a keyboard mapping display. In addition to keyboard input, TrueViz provides Unicode character input using a code table, so that any Unicode character can be selected and inserted into a text.

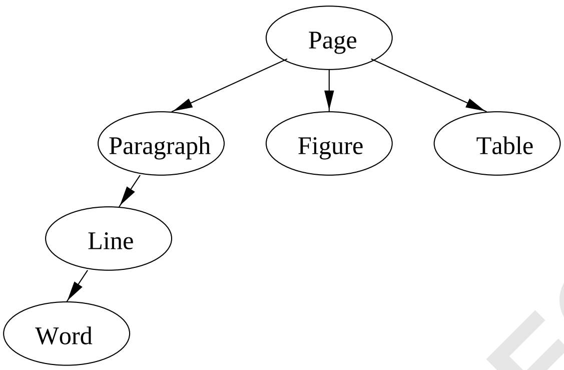


Fig. 9. User-defined entity structure.

6.3. Adding new input capabilities

Input capabilities for various languages can be easily added using the common interface *did.gui.DIDInputMethod*. A new input capability can be added by implementing the following member functions of the interface.

public DIDKeyBDisplay getKeyBDisplay( ): The function for getting the keyboard mapping display.

public String getComposingText( ): The function for getting the current composed text.

public String getCommittedText( ): The function for getting the current committed text.

public void keyTyped(*char ch*): The function for sending a typed character to the input.

public void showKeyboard( ): The function for showing keyboard mappings.

7. Future directions

TrueViz provides basic OCR groundtruthing functionalities. We hope that researchers in the international community will volunteer to add other features.

Currently TrueViz provides only English and Russian input. Other languages such as Korean, Japanese, and

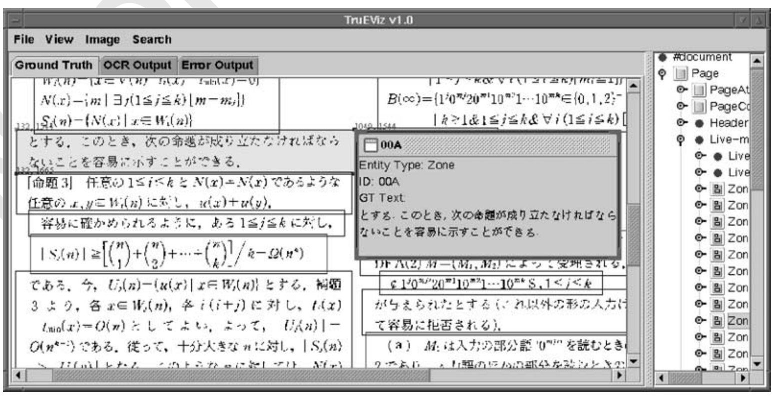
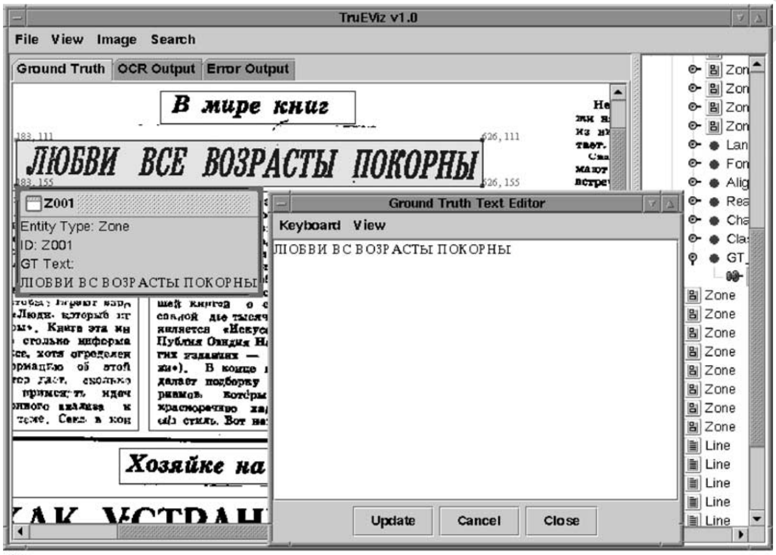


Fig. 10. Infopanel and multilingual display.

(a)



(b)

Fig. 11. Multilingual input. (a) Russian input in search. For users who are not familiar with the keyboard mapping, a keyboard mapping display window is provided. (b) Russian input in groundtruth editor.

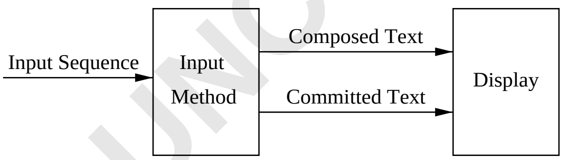


Fig. 12. Input system.

Chinese would be useful in multilingual OCR. A public-domain Java package with keyboards for various scripts/languages [34,35] that could be incorporated into TrueViz would be of great benefit to researchers.

Since tables are not trees, existing XML validation programs cannot verify table groundtruth data. Thus convenient ways for representing, annotating, and validating tables is needed.

Converters for DAFS to XML and XML to DAFS are currently implemented. This makes the XML representation compatible with the public domain performance evaluation toolkit PSET [36–39], and allows researchers to visualize segmentation evaluation results using TrueViz. Converters from SGT format (produced by the Pink Panther groundtruthing tool), XDOC (the Xerox representation for groundtruth), and the Caere representation would be helpful.

Document images contain huge amounts of data, and XML files can require more disk space than binary-formatted files. If compressed XML files could be saved and read, the file size of the XML files would not be a concern.

Zooming is a very integral part of any document image groundtruth visualization tool. A more “zoom-centric” design using the zoomable user interface package Jazz [40] could be explored.

TrueViz was tested by several members of our research group. A more thorough quantitative user evaluation using questionnaires would be desirable [41].

Since the OCR community currently does not have annotation standards similar to the Corpus Encoding Standard [42], it would be beneficial to start a working group to build such a standard and also ensure that TrueViz is compatible with this new encoding standard.

8. Summary

We have described TrueViz, a software system for multilingual groundtruth data entry for OCR. The system was designed to satisfy the requirements specified by the OCR community at the MLOCR conference [3]. TrueViz allows user to visualize and enter text in various languages. It also allows a data entry person to delineate physical zones on the document image. The text and geometric groundtruth is saved in an XML file. The decision to use open standards such as Java, XML, and Unicode allowed us to use open-source packages during the development. The TrueViz system is in the public domain and we hope that the international community will contribute components to the basic system.

Acknowledgements

The authors would like to thank the participants of MLOCR99 for valuable discussions; Jeff Czorapinski and Ivan Bella for their help in the initial phases of this project; Song Mao for discussion and user testing; Ben Bederson for comments on the user interface; Thomas Baby for suggesting the method of handling flexible entity structures; and Azriel Rosenfeld for editorial comments.

This research was funded in part by the Department of Defense under Contract MDA0949-6C-1250, Lockheed Martin under Contract 9802167270, the Defense Advanced Research Projects Agency under Contract N660010028910, and the National Science Foundation under Grant IIS9987944.

References

[1] C.H. Lee, T. Kanungo, TRUEVIZ User’s Manual, August 2000.

[2] T. Kanungo, C.H. Lee, J. Czorapinski, I. Bella, TRUEVIZ: a groundtruth/metadata editing and visualizing toolkit for OCR, in: Proceedings of the SPIE Conference on Document Recognition and Retrieval, San Jose, CA, 2001, pp. 1–12.

[3] Working group notes, international workshop on performance evaluation issues in multilingual OCR, http://www.cfar.umd.edu/∼kanungo/workshop/reco.html, September 1999.

[4] B.A. Yanikoglu, L. Vincent, Pink Panther: a complete environment for ground-truthing and benchmarking document page segmentation Pattern Recognition 31 (1998) 1191–1204.

[5] T. Fruchterman, DAFS: a standard for document and image understanding, in: Proceedings of the Symposium on Document Image Understanding Technology, Bowie, MD, 1995, pp. 94–100.

[6] J. Sauvola, H. Kauniskangas, MediaTeam Oulu Document Database, MediaTeam, University of Oulu, Finland, http://www.mediateam.oulu.fi/MTDB/, 1998.

[7] O. Hitz, L. Robadey, R. Ingold, An architecture for editing document recognition results using XML technology, in: Proceedings of the Fourth IAPR International Workshop on Document Analysis Systems, Rio de Janeiro, Brazil, 2000, pp. 385–396.

[8] C. Barras, E. Geoffrois, Z. Wu, M. Liberman, Transcriber: development and use of a tool for assisting speech corpora production, Speech Commun. Special Issue on Speech Annot. Corpus Tools 33 (1–2).

[9] E. Geoffrois, C. Barras, S. Bird, Z. Wu, Transcribing with annotation graphs, in: Proceedings of the Second International Conference on Language Resources and Evaluation, Athens, Greece, 2000, pp. 1517–1521.

[10] C. Barras, E. Geoffrois, Z. Wu, M. Liberman, Transcriber: a free tool for segmenting, labeling and transcribing speech, in: Proceedings of the First International Conference on Language Resources and Evaluation, Granada, Spain, 1998, pp. 1373–1376.

[11] S. Bird, D. Day, J.G.J. Henderson, C. Laptun, M. Liberman, ATLAS: a flexible and extensible architecture for linguistic annotation, in: Proceedings of the Second International Language Resources and Evaluation Conference, Athens, Greece, 2000, pp. 1699–1706.

[12] D. Day, J. Aberdeen, L. Hirschman, R. Kozierok, P. Robinson, M. Vilain, Mixed-initiative development of language processing systems, in: Proceedings of the Fifth Conference on Applied Natural Language Processing, Washington, DC, 1997.

[13] D. Doermann, D. Mihalcik, Tools and techniques for video performance evaluation, in: Proceedings of the 15th International Conference on Pattern Recognition, Barcelona, Spain, 2000, pp. 167–170, http://documents.cfar.umd.edu/LAMP/Media/Projects/ViPER/.

[14] D.F. Swayne, D. Cook, A. Buja, Xgobi: interactive dynamic data visualization in the X window system, J. Comput. Graph. Stat. 7, http://www.research.att.com/areas/stat/xgobi/.

[15] D.F. Swayne, N. Hubbell, A. Buja, XGobi meets S: integrating software for data analysis, in: Proceedings of the Symposium on the Interface, 1991, pp. 430–434, http://www.research.att.com/areas/stat/xgobi/.

[16] D.F. Swayne, D. Cook, A. Buja, XGobi: interactive dynamic graphics in the X window system with a link to S, in: Proceedings of the American Statistical Association Meetings, 1992, http://www.research.att.com/areas/stat/xgobi/.
[17] W.N. Venables, B.D. Ripley, Modern Applied Statistics with S-Plus, Springer, Berlin, 1999, http://www.splus.mathsoft.com/.
[18] S.D. Anderson, D.L. Westbrook, A. Carlson, D.M. Hart, P.R. Cohen, Common Lisp Analytical Statistics Package: User Manual, University of Massachusetts, 1995, http://eksl-www.cs.umass.edu/clasp.html.
[19] University of Augsburg, Mondrian, http://jetta.math.uni-augsburg.de/Mondrian/.
[20] National Center for Atmospheric Research/Meso-scale and Microscale Meteorology, PolyPaint User Manual, Version 3.0, http://lasp.colorado.edu/polypaint/home.html.
[21] C. Ahlberg, B. Shneiderman, Visual information seeking: tight coupling of dynamic query filters with starfield displays, in: Proceedings of the ACM CHI94 Conference, Boston, MA, 1994, pp. 313–317, http://www.spotfire.com/products/spotfire_net.asp.
[22] PIXOTEC, LLC., Slicer Dicer, http://www.slicerdicer.com/.
[23] B. Shneiderman, The eyes have it: a task by data type taxonomy of information visualizations, in: Proceedings of the IEEE Symposium on Visual Languages, 1996, pp. 336–343, http://otal.umd.edu/Olive/.
[24] B. Shneiderman, Designing the User Interface, Addison-Wesley, Reading, MA, 1998.
[25] D. Gusfield, Algorithms on strings, trees, and sequences: computer science and computational biology, Cambridge University Press, Cambridge, 1997.
[26] E.R. Harold, XML Bible, IDG Books, Foster City, CA, 1999.
[27] T. Bray, J. Paoli, C.M. Sperberg-McQueen, Extensible Markup Language (XML), W3C, http://www.w3.org/TR/REC-xml, 1998.
[28] L. Wood, A.L. Hors, V. Apparao, L. Cable, M. Champion, J. Kesselman, P.L. Hegaret, T. Pixley, J. Robie, P. Sharpe, C. Wilson, Document Object Model (DOM), W3C, http://www.w3.org/TR/DOM-Level-2/, 1999.
[29] B. McLaughlin, Java and XML, O'Reilly, Sebastopol, CA, 2000.
[30] R.B. Allen, J. Schalow, Metadata and data structures for the historical newspaper digital library, in: Proceedings of the Eighth International Conference on Information Knowledge Management, Kansas City, MO, 1999, pp. 147–153.
[31] T. Kanungo, R.B. Allen, Full-text access to historical newspapers, Technical Report CS-TR-4014, Laboratory for Language and Media Processing, University of Maryland, College Park, MD, April 1999.
[32] Proceedings of the IAPR Workshop on Document Layout Interpretation and its Applications.
[33] T.U. Consortium, The Unicode Standard, Version 2.0, Addison-Wesley Developers Press, 1997.
[34] K.Y. Leong, H. Liu, O.P. Wu, Java input method engine, in: Proceedings of the Seventh International World Wide Web Conference, Brisbane, Australia, 1998, http://www7.scu.edu.au/programme/fullpapers/1915/com1915.htm.
[35] K.Y. Leong, H. Liu, O.P. Wu, Web internationalization and Java keyboard input methods, in: Proceedings of INET 98, Geneva, Switzerland, 1998, pp. 21–24.
[36] S. Mao, T. Kanungo, Software architecture of PSET: a page segmentation evaluation toolkit, Technical Report CAR-TR-955, University of Maryland, College Park, MD, http://www.cfar.umd.edu/∼kanungo/pubs/trpset.ps., Software is available at http://www.cfar.umd.edu/∼kanungo/software/software.html, September 2000.
[37] S. Mao, T. Kanungo, PSET: a page segmentation evaluation toolkit, in: Fourth IAPR International Workshop on Document Analysis Systems, Rio de Janeiro, Brazil, 2000, pp. 451–462.
[38] S. Mao, T. Kanungo, Empirical performance evaluation methodology and its application to page segmentation algorithms, IEEE Trans. Pattern Anal. Mach. Intell. 23 (3) (2001) 242–256.
[39] S. Mao, T. Kanungo, Empirical performance evaluation of page segmentation algorithms, in: Proceedings of the SPIE Conference on Document Recognition and Retrieval, 2000, pp. 303–314.
[40] B. Bederson, J. Meyer, L. Good, Jazz: an extensible zoomable user interface graphics toolkit in Java, Technical Report CS-TR-4137, UMIACS-TR-2000-30, University of Maryland, College Park, MD, http://www.cs.umd.edu/hcil/jazz/, May 2000.
[41] J.P. Chin, V.A. Diehl, K.L. Norman, Development of an instrument measuring user satisfaction of the human–computer interface, in: Proceedings of SIGCHI '88, New York, NY, 1988, pp. 213–218, http://www.lap.umd.edu/QUIS/index.html.
[42] Expert Advisory Group on Language Engineering Standards, Corpus Encoding Standard—Document CES 1, Version 1.5, http://www.cs.vassar.edu/CES/.

About the Author—CHANG HA LEE received a BS in Computer Science from Seoul National University, Seoul, Korea, in 1995 and an MS in Computer Science from University of Maryland, College Park in 2001. He is currently doctoral student at the University of Maryland Computer Science department. His research interests include 3D computer graphics, molecular graphics, bioinformatics, and document image analysis.

About the Author—TAPAS KANUNGO is a Research Staff Member at the IBM Almaden Research Center, San Jose, CA. Prior to joining IBM he served as a Co-Director of the Language and Media Processing Lab at the University of Maryland, College Park, where he conducted research in the areas of document image analysis, OCR-based cross-language information retrieval, pattern recognition and computer vision. He received his Ph.D. and M.S. in Electrical Engineering from the University of Washington, Seattle, in 1996 and 1990 respectively. From March 1996 to October 1997 he worked at Caere Corporation, Los Gatos, CA, on their OmniPage OCR product. During the summer of 1994 he worked at Bell Labs, Murray Hill, NJ, and during the summer of 1993 worked at the IBM Almaden Research Center, San Jose,

CA. Prior to that, from 1986 to 1988, he worked on speech coding and online handwriting analysis in the Computer Science group at Tata Institute for Fundamental Research, Bombay, India.

Tapas Kanungo co-chaired the 2001 SPIE Conference on Document Recognition and Retrieval and the 1999 IAPR Workshop on Multilingual OCR. He was a Co-Guest Editor of the International Journal of Document Analysis and Recognition Special Issue on Performance Evaluation, and has been program committee member of several conferences. He is a senior member of the IEEE.