Document Representations

Dorothea Blostein, Richard Zanibbi
School of Computing
Queen's University
Kingston, Ontario, Canada
{blostein, zanibbi}@cs.queensu.ca

George Nagy
Electrical, Computer, Systems Eng.
Rensselaer Polytechnic Institute
Troy, New York, USA
nagy@ecse.rpi.edu

Rob Harrap
Department of Geology
Queen's University
Kingston, Ontario, Canada
harrap@geol.queensu.ca

Abstract

Many document representations are in use. Each representation explicitly encodes different aspects of a document. External document representations, using standard file formats (such as JPEG, postscript, HTML, LaTeX), are used to communicate document-data between programs. Internal document representations are used within document analysis or document production software, to store intermediate results in the transformation from the input to output document representation. These document representations are central to defining and solving document analysis problems. Issues that can be investigated include defining equivalence of documents and distance between documents, mathematically characterizing the mapping between document representations, characterizing the external information needed to carry out these mappings, and characterizing the differences between the forward and inverse mappings that occur during document analysis and document production. From our ongoing investigation of these issues, we present a summary of internal document representations used in the table-recognition literature, and case studies of external document representations in the domains of circuit diagrams and text documents.

1. Introduction

Document processing systems use a variety of document representations. This is true both for systems that analyze documents and systems that produce documents. The input to a document analysis system is an appearance-oriented representation (typically a bitmap), and the output is a more explicit representation of the information that is supposed to be conveyed by the document (using a domain-specific file format such as LaTeX for mathematics or PSpice schematics for circuits). Document production systems perform a translation in the other direction. In addition to these *external document representations*, which are stored in files, document-processing software commonly uses additional, intermediate document representations to bridge the gap between the input and output representations. We call these *internal document representations.*

1.1 Issues

An understanding of document representations is fundamental to document image analysis. Issues that can be investigated include the following.

- Define *equivalence* of documents, and *distance* between documents, for a variety of document representations. This allows a formal problem statement for document analysis and document production: the input is a document in one representation, and the output is an equivalent document in a different representation. An error metric is provided by the distance between the ideal and actual output files. As discussed below, similarity measures are difficult to define. The worth of a similarity metric can be assessed based on its utility for a given task, or by measuring its fidelity to human behavior [Resn99].

- Mathematically characterize the mappings between document representations. Are the mappings one-to-one or one-to-many? Are they invertible? If the mappings are not invertible, characterize the information that is lost. We are considering two approaches to mathematically characterize representations: (1) a communications perspective, using an information-theoretic characterization, and (2) a compiler perspective, using grammatical techniques.
- Characterize the amount and type of external information that is needed in order to transform from one document representation to another. This provides one measure of the complexity of the mapping.
- Characterize the differences between the forward and inverse mappings that occur during document analysis and document production. Document analysis must address issues of noise and uncertainty, but these issues are of little concern during document production. On the other hand, document production must address issues of readability and aesthetics, and these issues are of little concern during document analysis. Thus, there are significant differences between the external information (the domain model) that is used in document analysis software compared to document production software. As a result, there can be significant differences in the internal document representations that are used.
- Characterize the generality versus domain-specificity of a document representation. A bitmap is very general: the same file type can represent images from any document domain. Representations such as LaTeX or Spice are more domain-specific.

A variety of factors make these issues difficult to investigate. One problem is that document analysis always involves interpretation. The human reader (or computer software) interprets the document in light of his/her/its goals, beliefs, and judgment. Variations in terminology also cause difficulty. Terms such as *layout*, *context*, *abstraction*, *syntax* and *semantics* are widely used, but the meaning of these terms is not standardized, so their use easily leads to disagreements and confusion. The great variety of documents and document notations further complicates the investigation.

Complex problems arise in attempting to define document equivalence. Incompatibilities can result from subtle differences in assumptions that underlie the data representation. For example, the Geographic Information Systems ARC/INFO and MapInfo differ in their definition of a region (polygon) object [Gahe99]. A polygon in ARC/INFO is defined as part of a coverage (assuring topological closure, boundary-coincidence between neighboring objects, and absence of overlap of object interiors), whereas a polygon in MapInfo has no such constraints (objects may overlap, and common boundaries are not recognized). Gahegan concludes that an exchange format based around geometry and associated attributes is not sufficiently rich to support informed use of data. The geographic model must be included because of subtle differences in the meaning of data that are not apparent when considering their geometry alone. In moving data from one GIS to another, mismatches in the underlying data models could be reported as warnings to the user.

Distance between documents is at least as difficult to define as equivalence of documents. Distance measures used in document image analysis include edit distance [KNRN95, PhCh99], Hamming distance, and Hausdorff metric. These distance measures are primarily aimed at characterizing the amount of noise or recognition error. For evaluation of document production systems, distance measures must reflect aesthetic issues as well. For example, a circuit diagram can be transformed to two document images, one with a readable layout and the other with a spaghetti-like layout. These two document images both correctly depict the structure of the circuit diagram, but they differ markedly in their "aesthetic appeal". Aesthetic criteria used for graph layout could provide a starting point. These include minimizing edge crossings, minimizing drawing area or aspect ratio of the drawing, minimizing total edge length or maximum edge length or variance of edge length, minimizing the number of edge bends, maximizing the smallest angle between two edges incident on the same vertex, and displaying symmetry [DETT99].

1.2 Levels of Representation

Many publications about Document Image Analysis describe levels of document representation. Here are a few examples. Srihari represents postal addresses at the image level, feature level, character level, word level, phrase level, sentence level, paragraph level, and document level [Srih93]. Sennhauser's blackboard system for text analysis contains hypotheses at the page, block, chunk, and symbol level [Senn94]. Vaxivière and Tombre use one level for each phase in engineering drawing analysis: lines and blocks, shafts, symmetric entities, functional setups [VaTo94]. Maderlechner and Mayer define a four-level model for maps, consisting of levels for image, image graph, graphics and text, and semantic objects [MaMa94]. Du et al. discuss a contextual architecture, for handling contextual constraints in a uniform way while performing pattern recognition tasks that require intermediate levels of abstraction [DDLA97].

Many authors (including ourselves) have used the phrase *levels of abstraction* to refer to these internal document representations. However, it is difficult to define what *abstraction* means in the context of document representations. One of the traditional definitions of *abstraction* is as follows.

Abstraction (from *ab*: away from, and *trahere*: to pull) withdrawing or removing some aspect of an object or exemplar, to focus on the rest.

Using this definition, a bitmap image of mathematical notation is not “less abstract” than the corresponding LaTeX file. Each of these representations contains information that is not present in the other. Both can be thought of as abstractions of a *Parent Document*, which contains complete information about the document, including appearance, structure, and interpretation. (Unambiguously defining the *Parent Document* is difficult because the mapping between bitmap and LaTeX is not one to one. A given bitmap corresponds to many LaTeX files, and a given LaTeX file corresponds to many bitmap files.) We avoid the term *abstraction* in the rest of this document, since we are not able to define it satisfactorily. For the future, examination of the well-established term *abstract data type* may help establish a definition. The definition should provide a clear basis for testing the relative abstraction levels of two document representations, with the answer that both are at the same level, or that one is at a lower level than the other, or that the ranking is undefined.

The use of intermediate levels of representation is widespread in all types of image analysis, not just document image analysis. For example, satellite images are treated at the raw pixel level, two feature levels (segmentation, and clustering/classification), and semantic level, in [BeCL97]. Truvé describes an approach to computational vision which is based on multiple levels of interpretation. Transitions from one level to another use three stages: parsing (which assigns labels to features and groups of features), interpreting, and pruning [Truv90]. Levels of representation are also used in document production systems. For example, eleven passes for producing music notation are described in [BlHa94]; each of these passes produces an intermediate representation.

In studying document representations, we can focus on internal representations (Section 2) or external representations (Section 3). Internal document representations are described in publications, but it is difficult to collect sample documents that use these representations. In the case of external document representations, sample documents are readily available. However, data must be interpreted cautiously, because a given file format may allow a variety of data to be stored. For example, a postscript file typically contains symbol information such as "character 'A' at (x, y)" and "line of thickness B with endpoints (x1, y1) and (x2, y2)". However, a postscript file can also directly include bitmap images. Section 3 has further discussion of the study of files in an effort to characterize external document representations.

2. Internal Document Representations

As described in the literature, existing systems for document recognition and document production use a great variety of data structures and computational techniques. In many cases, the control structure of a document analysis system does not provide a clear reflection of the level-oriented language definition that is guiding analysis. Rather, the control structure reflects the fact that document analysis involves many shifts of attention from one level of representation to another. For example, contextual information at one level may guide analysis decisions at another level. Or, the presence of certain configurations at one level may cause the formation of analysis hypotheses at another level. Parts of the document may be fully recognized at a time when other parts have been only partially recognized. Our goal is to characterize the levels of representation used in a variety of document domains, with minimal dependence on the details of particular systems for recognizing or producing documents. We begin with tables and table-recognition systems.

Table 1 introduces a model of document structure [ZaBC03]. This model has been useful in analyzing the table-recognition literature, and can perhaps be adapted to other document-recognition or document-production domains. In Table 1, each level in the document structure is characterized by a set of objects, where each object has four inter-related types of content, which describe the logical structure and physical structure of a document.

Logical Structure

- Object Type: type of data represented by the object, references between objects
- Object Syntax: composition of objects to form this object; spatial relations among objects.

Physical Structure

- Object Geometry: location of objects
- Object Formatting: formatting attributes and spacing of objects.

For a related discussion, see [Hand99]. In using Table 1 for describing the table-recognition literature, the Primitive Region level consists of four types of objects: Table, Block, Cell and Cell Content. Columns and rows are considered to be different types of Block object [ZaBC03].

A goal of the model in Table 1 is to characterize levels of representation, independently of the control flow of a particular document processing system. If levels of representation can be characterized independently of control flow, then levels can be used to precisely define the task that should be accomplished by document processing software. There already are various models of levels which integrate control flow into the model. For example, Maderlechner and Mayer uses four levels in a model of large-scale maps [MaMa94]. The four levels are image, image graph, graphics and text, and semantic objects. Each level consists of objects, operations to be performed on the objects, and relations between the objects. In addition, some operations and relations cross between levels. The control strategy contains a mixture of top-down and bottom-up operations.

Table 1 Levels of Representation in Documents

Object type	Object Syntax	Object Geometry	Object Formatting
Data Array level			
Pixel map or Character Map (e.g. Character Maps are used for email documents)	Matrix of pixel or character values. A pixel-map object is composed of pixel sub-objects.	Polygon describing shape of the pixel-map or character-map object.	None.
Primitive level			
Connected component, labeled with character or symbol class	Set of adjacent data array cells. E.g. a connected component of pixels, or a connected set of delimiter characters in an email document.	Polygon (possibly with holes) describing shape of object.	Font and symbol attributes (e.g. font family, font size, style, colour; line thickness)
Lexical level			
Lexical object types include number, right-arrow, dotted line	The lexical object is a composition of one or more primitives (.e.g ≤, '->', dotted lines)	Polygon or parametric shape.	Spacing of primitives. Font and symbol attributes (e.g. family, size, style, colour)
Primitive Region level			
Primitive Region types include line, paragraph, block of text, table, math expression, image, chart, vector drawing. References within, outside region.	Composition of lexical objects and primitive regions into a primitive region (such as a paragraph of text, or a table). Spatial relations on lexical objects and primitive regions.	Polygon or parametric shape.	Spacing of lexical objects and primitive regions.
Functional Region level			
Functional Region types include figure, table and associated text, section heading, section, offset image. References between primitive regions.	Composition of functional and primitive regions into a functional region (defining the reading order of these regions). Spatial relations on primitive and functional regions.	Polygon or parametric shape.	Spacing of primitive and functional regions.
Page level			
Page types include title page, body page. References between functional regions.	Composition of functional regions into a page (defining the reading order of functional regions). Spatial relations on functional regions.	Polygon or parametric shape.	Spacing of functional regions.
Document level			
Document types include technical article, book. References between pages	Set of pages. Page ordering.	None	None
Corpus level			
Corpus type (e.g. table recognition literature). References between documents.	Set of documents. Document ordering (e.g. alphabetical by title)	None	None

Criteria are needed for evaluating a proposed characterization of levels. The model in Table 1 has proven useful in summarizing the table-recognition literature, but it is difficult to formally justify the correctness or effectiveness of a model such as this. An open question is whether the levels defined in Table 1 (or some other set of levels) could be used to describe both document analysis and document production. For example, a system for producing music notation [BlHa94] uses internal document representations that do not correspond well to the levels in Table 1. Perhaps new, better software for producing music notation could be written, using levels such as those in Table 1.

However, production of music notation involves complex decisions (choose stem directions, choose beaming boundaries, determine note spacing) that do not arise in analysis of music notation. Conversely, analysis of music notation involves complex issues (dealing with noise, segmenting overlapping symbols) which do not arise in production of music notation. Therefore, it is an open question whether the same levels of representation can or should be used for both document analysis and document production.

3. External Document Representations

External document representations use standardized file formats such as JPEG, postscript, HTML, LaTeX, and PSpice schematic. Files are readily-available artifacts, which can be used to study document representations. The existence of different kinds of files to represent essentially the same information in different forms is a visible manifestation of "levels of representation". One of our goals is to characterize or define the differences between these levels. *Meaning* is not an intrinsic property of a document, but something that depends on the program or human reading the document. Thus, a LaTeX file does not intrinsically have more meaning than the postscript file derived from it, but it is obvious that different tools are required to extract that meaning from the two files.

We propose to study the use of external document representations in practical situations, characterizing the quantity and type of data that is stored. Care must be taken to allow for the variety of data that can be stored using a given file format. For example, a postscript file can include bitmap images within it. Thus, a program that translates from JPEG to postscript could be merely repackaging the same pixel data, or it could be performing character and line recognition.

3.1 Units of Information

We propose to gather statistics about files used to store sample documents. For a math document, these files might include a bitmap file, a postscript file, a LaTeX file and a Maple file. Each file contains *units* of information, where a unit can be an object, a relationship between other units, or a parameter (with a scope indicating which other units are affected). Each unit is explicitly represented by some bits in the file. Implicit relationships do not count as units. For example, a bitmap file could be characterized as containing units that are objects (pixels) and units that are parameters ("number of rows", "number of columns", "number of bits of colour"). Implicit relations between pixels ("this pixel is a neighbor of that pixel") do not count as units. Clearly, a lot of variability will occur in these measurements. For example, the number of pixels in a bitmap depends greatly on the spatial resolution. Much work will be required to define and measure the units of information for a real document. Spice Schematics, for instance, generate about half a dozen files, some ASCII and some binary, for even the simplest circuit. Preliminary measurements are discussed in the case studies below.

3.2 Obtaining Files Representing a Document

The first step in studying a sample document is to collect files that contain different representations of the document. At least two methods can be used. The first method begins with a scanned document image, and applies document image analysis software to produce files that more explicitly encode the information content of the document. The second method begins with manual entry of the information content of a document; document production software is used to produce files that explicitly encode document appearance (e.g. postscript and bitmap files). We plan to use sample documents from a variety of domains, including formatted text, math notation, music notation, maps, and engineering drawings. For example, a sample document in the music domain is Beethoven's Harp String Quartet. Method one generates a set of files for this sample document, by scanning a published edition of the string quartet. These scanned images are processed by music-notation analysis software (e.g. SmartScore [SS]), to produce other document representations, such as MIDI or NIFF (Notation Interchange File Format) or other formats discussed in [Self97]. Method two generates a set of files for this sample document by using software for producing music notation. Many software packages are on the market. If Lime is used [BlHa94], the user enters the notes for the String Quartet by playing on a MIDI keyboard. This information (sequences of notes, with pitch and duration) is stored in a Tilia [HaBl93] file. Other document representations (NIFF, postscript, bitmap) are produced automatically. Method two could be applied again, this time using software such as MusicTeX. In this case, the user types the MusicTeX source file for the string quartet. Then other document representations (postscript, bitmap) are produced automatically. In this manner, a variety of files, all representing the Harp String Quartet, are produced. We have not yet undertaken this study, but we believe that it will be interesting to compare the characteristics of files produced by document-analysis software to files produced by document-production software.

3.3 Case Study: Unformatted and Formatted Text Files

We study the representation of natural language text documents, such as novels or business letters. Although some of the representations for text can also accommodate tables, mathematical notation, line drawings, and even photographic images, here we restrict ourselves to plain text. We have not yet attempted to measure units of information, but begin with file sizes.

We consider specifically the representation of a 45-line page of text, with each line containing 60 characters (including blanks). A compact representation (which, however, could be compressed further due to the repetitive nature of the text) is a plain ASCII (.txt, .asc, .ans) file of 2,462 eight-bit bytes. At the other end, a direct bilevel 300 dpi representation, with eight bits packed to a byte, requires 1,051,875 bytes. The “300 mono” bmp file produced by GhostView is 4,224,062 bytes. Bitmaps can, of course, be compressed: for example, a CCITT G4 (digital fax) Tagged Image Format (tif) file is 130,00 bytes, and a “300 mono” GhostView PNG file is only 51,775 bytes.

PostScript can make use of standard font description files, but it can also encode the bitmaps directly. For the page above, the former requires 30,000 bytes, the latter 1.2 million bytes. The character-coded version, using standard font description files, provides better support for searchable text.

Adobe PDF and PostScript are essentially equivalent representations, and are inter-convertible. The major difference is that PDF files are encoded with a lossless compression algorithm. The size of the PDF file depends on the algorithm: here PDFWriter in MS-Word produced a 1600 byte file, while the Adobe PDF-Writer for Windows yielded 3200 bytes. Adobe Distiller, which is generally believed to preserve typeface fidelity, produced 9000 bytes. Because compression increases with the length of the text, the size of PDF files increases only sublinearly with the number of characters. There is, however, a fixed overhead of a few thousand bytes. Differences between the 600dpi printed versions of the various PDF files are small but noticeable. Character-encoded PDF files are searchable and annotatable. We did not consider annotations.

PostScript and PDF preserve format, but not explicitly. It is not, for instance, possible to search them for a specific format, or to copy a format to other files. MS-Word files maintain formats explicitly, are backward compatible with earlier versions of MS-Word, and also preserve some author preferences. They are therefore much larger. The .doc version of the above file was 24,000 bytes. Most of the essential information was also preserved in a 6700 byte Rich Text Format (.rtf) file, which was originally designed to be the *lingua franca* of word processors.

Tex and DVI depend, like PS and PDF, on externally stored typeface files. The file sizes are comparable to PDF. The portability engendered by ASCII encoding does not affect their level of representation, which is the same as that of other searchable and modifiable text representations.

The Parent Document, a complete, ideal representation of such a text document might consist of:

1. sentence and higher-level relations
2. interword syntax
3. lexicon of valid words
4. string of words
5. layout (formatting)
6. font libraries

The document representations described above contain various subsets of the information in the Parent Document. For example, postscript and PDF files contain items 4, 5, and 6, whereas a plain ASCII representation contains only item 4. The plain ASCII representation efficiently serves the needs of programs that do not make use of formatting information. These include most programs for automated information retrieval, text categorization, summarization and statistical text analysis.

3.4 Case Study: Circuit Files

We show several different representations of a simple, two-resistor circuit. Figure 1 shows the textual description of the connectivity, called a *netlist*. A circuit analysis program can check this file and, if no errors are found, produce a file that describes the behavior of the circuit, i.e., current and voltages at various points. The circuit file can include additional directives, such as calls for transient analysis, thermal analysis, or parametric plots. But the basic units of the netlist document are clearly *element type, element connectivity, and element value*. The netlist provides only circuit topology. The geometric properties of a circuit diagram are preserved in a textual schematics file, parts of which are shown in Figure 2. This schematics file can be used to generate the diagram shown in Figure 3. The schematics file contains the coordinates of every vertex of the circuit graph and has provisions for the footprints of component packages, pointers to other files with detailed behavioral component models, both lateral and hierarchical relationships (ports, buses) with other functional circuits-blocks described in other files, as well as numerous presentation layer details. Our example targets discrete components on a printed circuit board only because we are less familiar with integrated circuit specifications.

The schematics file (Figure 2) can be used to generate a netlist. The netlist generated from the schematics (Figure 1b) is equivalent, but not identical, to one created from scratch (Figure 1a), and either suffices for approximate electrical analysis of the circuit. Combined with component packaging information from other files, the netlist also allows downstream programs to configure a circuit board for physical realization. The output of the board layout program can then be analyzed for stray capacitances and inductive couplings, which depend on the actual geometry rather than the layout used in the circuit diagram of Figure 3. In actual practice, the analysis of even a simple *physical* circuit requires a dozen different types of files.

Example of a PSpice circuit file

```
V              0 2 dc 12
R1             0 1 10
R2             1 2 20
.end
```

(a)

Schematics Netlist

```
R_R1        $N_0002 $N_0001  10
R_R2        $N_0001 0  20
V_V1        $N_0002 0 12V
```

(b)

Figure 1 A PSpice circuit (.cir) file with three elements: a 12V voltage source, a 10ohm resistor, and a 20ohm resistor. (a) Manually entered circuit file. (b) Netlist created by the Schematics program.

```
*version 9.1 671166557
u 8
R? 3
V? 2
@libraries
@analysis
@targets
@attributes
@translators
a 0 u 13 0 0 0 hln 100 PCBOARDS=PCB
a 0 u 13 0 0 0 hln 100 PSPICE=PSPICE
a 0 u 13 0 0 0 hln 100 XILINX=XILINX
@setup
unconnectedPins 0
connectViaLabel 0
connectViaLocalLabels 0
NoStim4ExtIFPortsWarnings 1
AutoGenStim4ExtIFPorts 1
@index
pageloc 1 0 1184
@status
c 103:04:28:19:20:11;1054164011
*page 1 0 970 720 iA
@ports
port 7 GND_EARTH 320 215 h

@parts
part 2 r 320 175 h
a 0 sp 0 0 0 10 hlb 100 PART=r
a 0 s 0:13 0 0 0 hln 100 PKGTYPE=RC05
a 0 s 0:13 0 0 0 hln 100 GATE=
a 0 a 0:13 0 0 0 hln 100 PKGREF=R1
a 0 ap 9 0 9 4 hln 100 REFDES=R1
a 0 u 13 0 21 3 hln 100 VALUE=10ÉÉÉÉÉÉ.

part 3 r 360 175 d

part 1 titleblk 970 720 h
a 1 s 13 0 350 10 hcn 100 PAGESIZE=A
a 1 s 13 0 180 60 hcn 100 PAGETITLE=
a 1 s 13 0 300 95 hrn 100 PAGENO=1
a 1 s 13 0 340 95 hrn 100 PAGECOUNT=1

@conn
w 6
s 360 215 320 215 5
@junction
j 360 175
+ p 3 1
+ p 2 2 ÉÉÉÉÉÉ..
+ s 7
+ w 6

@attributes
a 0 s 0:13 0 0 0 hln 100 PAGETITLE=
a 0 s 0:13 0 0 0 hln 100 PAGENO=1
a 0 s 0:13 0 0 0 hln 100 PAGESIZE=A
a 0 s 0:13 0 0 0 hln 100 PAGECOUNT=1

@graphics
```

Figure 2 A portion of the PSpice schematics file. A circuit drawing created from this file is shown in Figure 3.

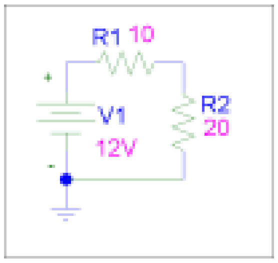

Figure 3 Display created from the PSpice schematics (.sch) file in Figure 2.

The Parent Document, a complete, ideal representation of a circuit, might consist of:

1. circuit topology
2. models of discrete components (voltage sources, resistors, transistors)
3. models of distributed components (physical and electrical properties of "wiring")
4. manipulable (vector-graphics) representation of the circuit diagram
5. symbol library

A netlist file contains items 1 and 2. A postscript or PDF file contains items 4 and 5. The transformation from bitmap to netlist is considered Document Image Analysis. The (interactive) transformation from netlist to bitmap is Document Production. Both transformations can use internal document representations, and the transformations can be carried out without constructing the entire Parent Document.

4. Summary and Conclusion

Document representations are central to defining and solving document analysis problems. Issues that can be investigated include defining equivalence of documents and distance between documents, mathematically characterizing the mapping between document representations, characterizing the external information needed to carry out these mappings, and characterizing the differences between the forward and inverse mappings that occur during document analysis and document production.

We have presented a summary of internal representations, useful for describing the literature in table recognition, as well as case studies of external document representations used for circuit-diagram documents and text documents. This is ongoing work.

References

[BeCL97] L. Bergman, V. Castelli, C.-S. Li, "Progressive Content-Based Retrieval from Satellite Image Archives," *D-Lib Magazine*, October 1997, www.dlib.org/dlib/october97/ibm/10li.html

[BlHa94] D. Blostein, L. Haken, "The Lime Music Editor: A Diagram Editor Involving Complex Translations," *Software – Practice and Experience*, Vol. 24, No. 3, March 1994, pp. 289–306.

[DETT99] G. Di Battista, P. Eades, R. Tamassia, I. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice Hall, 1999.

[DDLA97] L. Du, A. Downton, S. Luca, B. Al-Badr, "Generalized Contextual Recognition of Hand-Printed Documents Using Semantic Trees with Lazy Evaluation," *Fourth International Conf. on Document Analysis and Recognition*, Ulm, Germany, August, 1997, pp. 238–242.

[Gahe99] M. Gahegan, "Characterizing the Semantic Content of Geographic Data, Models, and Systems," Chapter 6 in *Interoperating Geographic Information Systems*, Goodchild, Egenhofer, Fegeas, Kottman, Eds., Kluwer, 1999, pp. 71-83.

[HaBl93] L. Haken, D. Blostein, "The Tilia Music Representation: Extensibility, Abstraction, and Notation Contexts for the Lime Music Editor," *Computer Music Journal*, Vol. 17, No. 3, 1993, pp. 43–58.

[Hand99] J. Handley, *Electronic Imaging Technology*, Chapter 8, SPIE Optical engineering Press, Bellingham Washington, 1999, pp. 289-316.

[KNRN95] J. Kanai, G. Nagy, S.V. Rice, T.A. Nartker, Automated Evaluation of OCR Zoning, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.17, No. 1, Jan. 1995, pp. 86-90.

[MaMa94] G. Maderlechner, H. Mayer, "Automated Acquisition of Geographic Information from Scanned Maps for GIS using Frames and Semantic Networks," *12th Int'l Conf. on Pattern Recognition*, Vol. 2, Jerusalem, Oct. 1994, pp. 361–363.

[PhCh99] I. Phillips, A. Chhabra, "Empirical Performance Evaluation of Graphics Recognition Systems," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 21, No. 9, Sept. 1999, pp. 849-870.

[Resn99] P. Resnik, "Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language," *J. Artificial Intelligence Research*, Vol. 1, July 1999, pp. 95–130.

[Self97] E. Selfridge-Field, *Beyond MIDI – The Handbook of Musical Codes*, MIT Press, 1997.

[Senn94] R. Sennhauser, "Integration of Contextual Knowledge Sources Into a Blackboard-based Text Recognition System," *IAPR Workshop on Document Analysis Systems*, Kaiserslautern, Germany, Oct. 1994, pp. 211-228.

[SS] SmartScore software, at http://news.harmony-central.com/Newp/1999/SmartScore.html

[Srih93] S. Srihari, "From Pixels to Paragraphs: the Use of Contextual Models in Text Recognition," *Proc. Second Intl. Conf. Document Analysis and Recognition*, Tsukuba, Japan, Oct. 1993, pp. 416-423.

[Truv90] S. Truvé, "Image Interpretation Using Multi-Relational Grammars," *Proc. Third International Conference on Computer Vision*, Dec. 1990, pp. 146–155.

[VaTo94] P. Vaxivière, K. Tombre, "Knowledge Organization and Interpretation Process in Engineering Drawing Interpretation," *Proc. IAPR Workshop on Document Analysis Systems*, Kaiserslautern, Germany, Oct. 1994, pp. 313-321.

[ZaBC03] R. Zanibbi, D. Blostein, J. R. Cordy, "Recognizing Tables in Documents," submitted for publication, May 2003.