Treatment of Diagrams in Document Image Analysis

Dorothea Blostein, Edward Lank, Richard Zanibbi
Computing and Information Science
Queen's University, Kingston Ontario, Canada, K7L 3N6
{blostein, lank, zanibbi}@cs.queensu.ca

Document image analysis is the study of converting documents from paper form to an electronic form that captures the information content of the document. Necessary processing includes recognition of document layout (to determine reading order, and to distinguish text from diagrams), recognition of text (called Optical Character Recognition, OCR), and processing of diagrams and photographs. The processing of diagrams has been an active research area for several decades. A selection of existing diagram recognition techniques are presented in this paper. Challenging problems in diagram recognition include (1) the great diversity of diagram types, (2) the difficulty of adequately describing the syntax and semantics of diagram notations, and (3) the need to handle imaging noise. Recognition techniques that are discussed include blackboard systems, stochastic grammars, Hidden Markov Models, and graph grammars.

1. Introduction

This paper concerns itself with diagram recognition, which is one component of document image analysis. Document image analysis involves conversion of a scanned document to an electronic form that captures the information content of the document [49]. The scope of diagram recognition problems is characterized by the following dimensions.

- The diagram notation A great diversity of diagram notations are in use. Recognition techniques have been developed for a variety of notations, including engineering drawings [21] [22] [33] [35] [72], mathematics notation [1] [10] [17] [25] [28] [71], music notation [4] [9] [23] [36], chemical structure diagrams [46], circuit diagrams [14] [37] [51], and line drawings [13] [52] [62]. There is need for a classification or categorization of diagrams, to provide a vocabulary for discussing the input domain of a diagram recognition system. Existing work in this area is surveyed in [8].
- The level of interpretation The following levels of interpretation occur in diagram recognition [42]: Image, Primitive, Symbol, Syntax, Structure, Entity, and Semantics. A diagram recognition technique can be characterized by the level of interpretation it takes as input, and the level that it produces as output. Symbol recognition converts from Image to Symbol level [16]. Techniques to convert from Symbol to Semantics level include [1] [14] [23], among many others. Techniques to convert from Image level directly to Structure level

This research is supported by Canada's Natural Sciences and Engineering Research Council.

(without symbol recognition) include [50] [74]. Integrated systems for converting from Image to Semantics level are described in [17] [33] [39].

- The presence of imaging noise Scanners introduce imaging noise. This causes uncertainty in primitive recognition and symbol recognition processes. Many document image analysis systems are applied to scanned, noisy images. There is also increasing interest to analyze noise-free diagrams in computer-generated documents. For example, clean postscript files are processed by [27] [54]. Such processing is needed because postscript files often become separated from the parent documents that generated them.
- Hand-drawn versus typeset diagrams Recognition techniques have been developed both for hand-drawn and typeset diagrams. Hand-drawn diagrams are challenging to process because symbol formation and symbol placement are much more variable than in typeset diagrams. In offline data, this makes symbol recognition more challenging and more error prone, and also increases the difficulty of recognizing the relevant spatial relationships among symbols.
- Online or offline data A hand-drawn diagram can be entered as online information (with timing information, as arises from use of a data tablet) or as offline information (without timing information, as arises from a scanner).

A great variety of diagram recognition techniques have been developed in response to the great variety of settings in which diagram recognition is used. Most of these are research systems, or systems that are custom-built to address the needs of one particular company (e.g. [2] [3]).

Few commercial diagram recognition systems are on the market. For example, one system for music notation is available (Musitek's SmartScore [68], preceded by MIDIScan), but none for math notation. This is in marked contrast to diagram generation systems, which are readily available (e.g. music notation [7] and math notation [41]).

High accuracy is required to make a document recognition system viable. The accuracy required of OCR (Optical Character Recognition) systems was studied by Cushman et al. [20]. They compare two methods of data entry for text that is available in hardcopy: (1) a person types the text and proofreads the result, (2) the text is recognized by an OCR system, followed by manual editing to correct recognition errors. They find that 98% correct recognition is needed to make the OCR system competitive. At this recognition rate, it takes half as much time to correct the OCR result as to type and proofread the text, and the corrected OCR result has as few residual errors as the proofread hand-typed result. There are OCR systems on the market that achieve such recognition rates. Unfortunately, most current diagram recognition systems do not meet comparable performance standards.

2. Challenges in Implementing a Diagram Recognizer

The designer of a diagram recognition system must find solutions to the following challenges. How should information about notational conventions be gathered, and how should it be represented within the software? How will noise, dialects, and error correction be handled? How will the prototype software be structured, so that it can scale up to handle the complexity of the complete notation?

2.1 Representing Notational Conventions

A diagram notation is a two-dimensional language characterized by notational conventions which define a mapping between symbols on a page and the information conveyed by a diagram. There is a many-to-one correspondence between images and information. This is because diagram layout can be varied in ways that do not change the information conveyed by the diagram. The various layouts differ in readability and aesthetic appeal. Ideally, diagram generation systems allow the user to choose a preferred layout, whereas diagram recognition systems accept all possible layouts.

Notational conventions are used in all parts of a diagram recognition system. Examples include the following. A symbol-segmentation module needs information about how symbols overlap, as in the separation of notes from staff-lines in music. A symbol recognition module needs information about symbol appearance (fixed symbols might be defined in a font, and parameterized symbols might be defined using a structural description). All aspects of semantic interpretation rely heavily on knowledge of notational conventions.

Software models of notational conventions are difficult to construct. There is no direct answer to the question "What is math notation?" or "What is music notation?". Diagram notations are not formally defined; rather, they are informally established, through common usage. The same is true of natural languages, such as English. In building a software model of notational conventions, the following sources of information can be drawn upon.

- Human experts can provide information about a diagram notation. Introspection is commonly used: the designers of a diagram recognition system rely on their own knowledge of the diagram notation.
- Sample documents can be studied.
- Informal written descriptions exist for a variety of notations, oriented toward manual typesetting (e.g. [15] [63]) or computer typesetting (e.g. [38] [64]). This information is not in a form that can easily be incorporated into diagram recognition software.
- Code in existing diagram generators and recognizers can be studied. The experience and knowledge in these systems is often significant. For example, Belkin notes that music-notation editors are typically under development for a decade or more [7].

A variety of methods can be used to store information about notational conventions within the recognition system. According to [53], these can be divided into three different approaches. The programming approach hides the knowledge about notational conventions in the source code of a program, lacking an explicit knowledge representation. The resulting systems are often monolithic, and therefore difficult to maintain and extend. The partially declarative approach describes the overall interpretation strategy in a declarative manner (as a set of rules), but program code contains the geometric reasoning used to identify primitives and symbols. The pure declarative approach uses a declarative specification language to formalize the entire set of notational conventions that are used. Pasternak recommends that the pure declarative approach be implemented via a partonomic structure and two taxonomic structures [53]. A partonomic structure describes aggregation of graphical primitives into domain-specific entities. One of the taxonomic structures describes geometric objects as specializations of other geometric objects. The other describes semantic objects as specializations of other semantic objects.

2.2 Handling Noise

Imaging noise may lead to errors or uncertainty in symbol recognition. Even when symbols are recognized perfectly, their syntactic role may be uncertain. (For example, a dot in mathematics notation may be a multiplication symbol, a decimal point, or noise.) The spatial relationships between symbols are often unclear; for example, an interesting study of in-line versus subscript versus superscript relations in mathematics is provided in [74]. A computational strategy is needed to manage this uncertainty. Two strategies that have been used include:

- Sequential processing with lists of alternatives. All possible interpretations are carried along from one stage to the next. For example, the symbol recognizer produces a list of alternatives (perhaps including "noise") for each symbol it recognizes. During syntax analysis, constraints are applied to eliminate alternative interpretations. As partial semantic analysis is achieved, further constraints may be applied (e.g. [24]).
- Concurrent execution with contextual feedback. If recognition processes execute concurrently (interleaved or in parallel), then later stages of interpretation can provide contextual feedback to earlier stages. The blackboard systems discussed in Section 5 provide such a control mechanism.

An interesting model for sequential processing is presented in [5]. The authors discuss some of limitations imposed by the need to carry forward all surviving alternatives, including the need to carry forward multiple segmentations. (*Segmentation* identifies all the pixels that constitute each symbol. If segmentation is incorrect, subsequent symbol recognition is certain to fail. Segmentation is difficult because overlapping symbols are common in diagram notations; in addition insufficient ink causes gaps in strokes, and excess ink makes two symbols look connected.)

Sequential processing does offer the advantage of isolating portions of the computation for intensive study. For example, researchers can focus on syntactic or semantic analysis without the need to study symbol recognition, as in [1] [23] et al. Concurrent execution with contextual feedback, on the other hand, offers increased flexibility in handling uncertainty, and may be necessary when the space of possible interpretations is extremely large. Further research is needed to develop general methods of providing contextual feedback.

2.3 Handling Dialects

Diagram notations have dialects. Therefore, it is impossible to construct one fixed software representation of a diagram notation. A better alternative is to define a core diagram notation and allowable variants. In some cases, the software may be able to automatically infer which variant to use. However, in many cases, the user will need to provide this information. Designing a suitable user interface for this purpose is a challenge.

2.4 Providing an Effective Interface for Error Correction

Diagram recognition systems are bound to make errors. Effective user interfaces are needed to allow users to correct errors in a convenient and natural way. A good user interface is critical to the practical operation of a recognition system. This aspect of diagram recognition systems has received relatively little attention; more research is needed. Here are two examples of systems in which error-correction facilities are provided.

Smithies et al. describe a system for recognizing mathematical expressions that are hand-written on a data tablet [69]. The user interface provides commands for correcting symbol recognition errors (e.g. a 'D' recognized as a '0'). for correcting over-segmented regions (where one symbol is interpreted as two symbols because of gaps between strokes), and for correcting under-segmented regions (where several symbols are mistakenly interpreted as a single symbol). Erroneous segmentation is corrected by using the mouse to draw a line to group a set of strokes together into a character.

Semi-automatic delineation of regions in floor plans is discussed in [65]. The goal is to identify subjective contours. These are not always explicit in a floor plan. An example is the division between kitchen and living room in an open-concept home. A set of subjective contours is automatically inferred. Manual correction of errors allows missing subjective contours to be added, and erroneous ones to be removed.

2.5 Handling Complexity

Many diagram notations are complex. Descriptions of their notational conventions are often lengthy (e.g. [63] for music notation). Of necessity, researchers in diagram recognition frequently work with small prototype systems, which handle only a small part of a diagram notation. Successful demonstration of a prototype system is certainly an important first step in demonstrating the viability of a recognition approach. However, methods that are effective in small prototype systems don't always scale up to a larger system. Complex, challenging issues must be addressed to achieve recognition systems that scale up gracefully. Much of the current research into software engineering is relevant to this topic.

The remainder of this paper summarizes a selection of existing methods used in diagram recognition. This is not a comprehensive survey, but serves to illustrate the variety of methods that have been used. These methods are not orthogonal; some of them can be used in combination.

3. Grammars and Production Rules

Grammars provide a systematic way to represent the syntax of a diagram notation, and provide a framework for attaching semantic routines as well. A grammar consists of a start symbol and a set of production rules. In a string grammar (the type used in compilers, for example), a production rule is used to replace one substring by another. Parsing involves a search for a correct sequence of production rules that can transform the start string into the target string. The string grammar defines a language of strings. Any string that is in the language can be parsed by the grammar. Other strings, which cannot be parsed, are syntactically incorrect according to this grammar.

The use of grammars in diagram recognition, and other pattern recognition applications, is discussed in [26]. Strings are one-dimensional, whereas diagrams are two-dimensional. String grammars can be adapted and extended in various ways, to allow them to describe two-dimensional patterns. For example, in PDL (Picture Description Language), a variety of spatial relationships are designated by the special characters * x + - ~. Combining this with a spatial interpretation of primitives (e.g. A is a horizontal line, B is a diagonal line, C is a circle), allows string expressions such as "A+BxC" to describe a picture [26]. Even with such tools, forming a linear description of a diagram can be difficult or inconvenient. As an alternative, a richer data structure than a sting can be used as the basis for a grammar. For example, a graph grammar consists of a start graph and a set of graph productions. Each production rule is used to replace one subgraph by another (Section 3.2). Imaging noise is a challenge in any grammar-based system. Techniques such as error-correcting parsing [26] or stochastic grammars (Section 3.3) can be used.

3.1 Anderson

Anderson's grammar for the recognition of math notation provides an excellent case study [1]. The recursive nature of math notation is particularly well-suited for syntactic analysis. Anderson uses a set-based grammar (called a coordinate grammar). During top-down parsing, a syntactic goal is assigned to a set of symbols. Initially, the syntactic goal EXPRESSION is assigned to the entire set of symbols. This symbol set must be parsed using a production rule that has an EXPRESSION non-terminal as its left-hand side. The production rule applies spatial constraints to partition the symbol set into subsets, and assigns a syntactic goal to each subset. This process repeats recursively, until the parse succeeds, or all possibilities have been exhausted. As an example, there is a production with SUMTERM on the left-hand side, and with four symbol subsets on the right hand side. One of these subsets is the symbol Σ; this symbol must be present if this production is to be applied. The other three subsets consist of symbols that are above, below, and to the right of the Σ. If there are symbols that do not fit into these categories (e.g. a symbol that is to the left of the Σ, or to the bottom-right of it), then the production cannot be applied. This means that an input expression is rejected as unparsable if it contains a Σ whose limits are wider than the Σ symbol itself. Similar symbol-layout restrictions occur elsewhere in the grammar. Nevertheless, Anderson's made a great contribution with this work, which is particularly impressive considering that it was carried out in the 1960s.

3.2 Graph grammars and Graph transformation

Graphs are an appropriate data structure to use in diagram recognition. Vertices can represent pixels, primitives, symbols, syntactic entities, structural entities, diagram interpretations, etc. Edges represent relationships among vertices. Attributes are associated with vertices and edges, to record non-structural information. Attributes in an image-level graph might record (x, y) image locations, whereas attributes in a semantic-level graph record the meaning that has been extracted from the document image.

Graphs can be processed using graph productions [12]. These are rules for transforming one graph into another, by removing a subgraph and replacing it with another. This is analogous to the role played by string productions, where one substring is replaced by another. Graph productions can be organized into a graph grammar (by designating a Start graph, as well as terminal and non-terminal labels). Alternatively graph productions can be used to transform an input graph directly into an output graph (by adding a means of controlling when productions are to be applied). Both methods of using graph productions have been applied to diagram recognition.

The transformation approach has been applied to recognition of circuit diagrams [14], music notation [6] [23] [24], and math notation [28]. The graph grammar approach has been applied to recognition of engineering drawings [22], tables [61], and math notation [43]. Most of these are research prototypes. However, the table recognition work by Rahgozar and Cooperman became part of a commercial product.

3.3 Stochastic Grammars

Images of diagrams are noisy. They contain imaging noise, coffee stains, areas with excessive amounts of ink (causing symbols to bleed together), and areas with insufficient ink (causing breaks in symbols). Various approaches can be taken when a grammar is to be used to analyze a noisy image [26]. These include error-correcting parsing, which finds the nearest legal interpretation of the input, as well as stochastic grammars, which add probabilistic information to the language described by a grammar.

A stochastic grammar is used by Chou to analyze typeset mathematics expressions [17]. The system is able to interpret low-resolution images that contain significant noise. Recognition of isolated symbols is very difficult in these images, because of the poor image quality. The images are interpreted successfully since the stochastic grammar finds a good interpretation of the input as a whole. Each production rule in the stochastic grammar has a probability associated with it. These probabilities are used to calculate an overall probability for each parse. The system assumes that the input conforms to a certain style of mathematics formatting (as produced by the troff system). Also, the entire character set must be known, with a sample of each character at each possible point size.

3.4 Visual Language Methods

The work surveyed in Sections 3.1 to 3.3 comes from the document recognition community. This research community publishes in journals [30] [31] and conferences [55] [57] [58] [60], among others. There is a separate community of visual language researchers, who publish in journal [34] and conferences [56] [59], among others. There is overlap in the problems being addressed by these two research communities, meaning that the two communities could likely learn from one another. The document recognition community is building systems to recognize diagram languages that are in use in society. The visual language community is developing new diagram languages (such as visual programming languages), and is developing technology for recognizing, editing, and generating diagrams expressed in these languages. In document recognition, much research effort centers on image processing (noise reduction, skew removal, etc.), vectorization, segmentation, and symbol recognition [49]. Consequently, relatively less effort has been placed on later interpretation stages (syntactic and semantic processing). In contrast, visual language researchers have intensively investigated methods of defining the syntax and semantics of visual languages, and methods of constructing parsers and interpreters for visual languages.

Some visual language approaches can be extended to make them suitable for document image analysis. Visual language approaches are summarized in the extensive survey by Marriott et al. [45]. These include grammatical approaches (string grammars with generalized relations, graph grammars, and multiset grammars), logic-based approaches (definite clauses, constraint logic programming, formalization of topology, logic formalisms containing visual expressions), and algebraic approaches (algebraic formalisms defining picture domains, application domains, and their correspondence). These approaches have been most intensively tested on newly-developed visual languages, particularly visual programming languages. These new languages are defined in a way that makes them amenable to formal treatment (unambiguous, easy to parse, clear semantics attached to spatial relationships). Consequently, these approaches need to be extended and adapted, before they can be used in document recognition systems. As an example, Costagliola et al. have developed *positional grammars*, and associated parsing methods [19]. This formalism adapts traditional, efficient LR parsing methods (as used in compilers) to process a significant class of visual languages. We were able to use some of the ideas in this approach to construct a document recognition system that recognizes handwritten mathematics expressions [75]. Significant extensions to the formalism were required, since sophisticated spatial reasoning is needed to interpret the messy symbol placement that is common in handwritten math expressions. Our extensions include a scheme for partitioning the image space around each symbol, the definition of searching functions to locate symbols in a baseline, and the use of tree transformations to refine the parse tree.

4. Recognition using a Model of Diagram Generation

Diagram recognition converts an image to a semantic representation. This is the inverse of diagram generation, where a semantic representation is converted to a corresponding image. Implicitly or explicitly, a diagram recognizer must be aware of the diagram generation process: the diagram recognizer must know the notational conventions that were used to create the diagram. Some approaches to diagram recognition use an explicit model of the diagram generation process. These are reviewed here.

The influential paper by Kopec and Chou [39] proposes the following structure for a document recognition system. The system contains a model of the diagram recognition process, which consists of a *Message Source*, an *Imager*, and a *Channel*. The Message Source defines the space of possible messages (semantics) that might be encoded in a diagram. As well, it defines the probability of each particular message. The Imager defines the mapping from a message to an noise free image. The Channel maps a noise free image to an observed image, by introducing distortions such as skew, blur, and additive noise. Given these components (Message Source, Imager, and Channel), it is now possible to formulate a precise statement of the optimization problem that must be solved by the *Decoder* (the diagram recognizer): the decoder receives an observed (noisy) image, and must produce a maximum-likelihood estimate of the original message. Kopec and Chou implement a solution to this optimization problem using a finite-state machine (a Hidden Markov Model) to model the Message Source and Imager. Their Channel is a simple bit-flip model, where each pixel has a certain probability of being flipped (changed from white to black or vice versa). They obtain excellent results in interpreting noisy images. Their results are particularly impressive considering the simplified model of diagram generation that is used: since the Hidden Markov Model is based on a finite state machine, it can only capture context free aspects of syntax.

In effect, the Kopec and Chou algorithm performs an exhaustive search of the entire space of messages (all possible image-interpretations). They invoke a generator for each possible message, and compare all the generated images to the given image to find the best match. A naive implementation of this computation would have very long run time. Reasonable run time can be achieved by appropriately restricting the form of the generator, and using a dynamic programming algorithm. In [39], a Hidden Markov Model is used for the generator (i.e., for the Message Source and Imager). The generator model is extended to a stochastic context free grammar in [18]. The HMM method is applied to music notation in [40]. In order to achieve a context-free description of the generator, the music notation is restricted to a single voice, with no beams or slurs between notes. Context-sensitive layout issues, such as the effect of note length on note spacing, cannot be modeled. Nevertheless, encouraging results are achieved.

Another study related to the interaction between diagram generation and diagram recognition is [11]. Here, the focus is on reusing the expertise in existing diagram generators, in order to improve diagram recognition software. As a case study, an existing music-notation editor (Lime) is used to proofread and correct the raw output of MIDIScan (a third-party commercial recognizer for music notation).

5. Blackboard Systems

The blackboard architecture provides a general and flexible framework for combining diverse knowledge sources [32]. The name derives from an analogy to human experts communicating via a blackboard. The three main parts of a blackboard system are a blackboard data structure, knowledge sources, and a control component [32]. The blackboard contains recognition hypotheses. The information on the blackboard is updated by the knowledge sources. The control component oversees the triggering of the knowledge sources. The blackboard can represent several competing recognition hypotheses, with associated confidence values. The knowledge sources read and update the confidence values as they read and update the recognition hypotheses.

Blackboards used in recognition systems are commonly divided into levels of abstraction. Kato and Inokuchi use five levels (image, primitive, symbol, meaning, goal) for music recognition [36], and four levels (input diagram, symbol hypotheses, diagram hypotheses, recognition result) for circuit-diagram analysis [37]. Vaxivière and Tombre use four levels (lines and blocks, shafts, symmetric entities, functional setups) for analysis of engineering drawings [72]. Novak and Bulko use five levels (picture, text, picture-model, text-model, problem-model) to interpret the text and diagram defining a physics problem [48]. The hierarchy of levels, models and algorithms used in text recognition are discussed in [70]. Four main levels are used: the image level (binary and grayscale), the character level, the linguistic level (divided into word, phrase, and sentence levels), and the textual structure level (divided into paragraph and document levels).

Knowledge sources can be of different types. For example, Novak and Bulko suggest using five types of knowledge-sources for semantic analysis [48]. These knowledge sources (1) represent expectations about related objects, (2) group related diagram elements, (3) make inferences from common-sense rules and domain rules, (4) relate diagram elements to *a priori* knowledge, and (5) infer missing items.

Blackboard architectures provide a flexible control structure, which can be used to handle noise and uncertainty. For example, in a music recognition system [36], pixels are erased from the blackboard as they are interpreted. If a contradiction occurs at a higher level of interpretation, the pixel-level data is restored and re-interpreted. This results in fast processing for clean input, with slower processing and more backtracking for noisy input. The cleaner parts of the input image provide contextual information for interpreting noisier parts of the image [36] [37].

Authors in document image analysis mention various advantages of blackboard systems. Sennhauser praises the ability to integrate new knowledge sources into the system, the support for conflicting hypotheses, the contextual feedback from high-level processing steps to low-level processing steps, and the ability to reprocess with new premises in cases where earlier processing was hampered by inappropriate hypotheses [67]. Wang and Srihari find that blackboard systems are able to handle both structure and randomness in the input: evidence is acquired from diverse knowledge sources, without putting undo emphasis on any single source. Also, the control structure can invoke knowledge sources such that the effort expended in gathering evidence is in accordance with input complexity [73]. Novak and Bulko like the flexible and adaptive order of processing available in a blackboard system. In their application, clues can be opportunistically exploited to relate and interpret the text and diagram that describe a physics problem [48].

6. Schema-based systems

Schema representations have been used in several diagram recognition systems. A schema class defines a prototypical drawing construct. During recognition a new schema instance is created to represent each drawing construct that is found in the image. Two hierarchies are important in schema representations: the *class hierarchy* and the *instance hierarchy*.

The class hierarchy represents commonalities among related drawing constructs. For example, Joseph and Pridmore define *broken line*, *chain*, and *witness* as subclasses of class *line* [33]. (A *witness* line forms part of the dimensioning information in an engineering drawing.)

The instance hierarchy represents composition of objects or relationships into more complex objects or relationships. A schema instance has a “subpart” relationship to each of the smaller geometric entities that constitute it. Schemata provide a uniformity of representation: they are used to represent objects as well as relationships between objects.

The Mapsee systems provide one example of schema-based diagram recognition [29] [47]. These systems are applied to interpretation of sketch maps. A sketch map is a hand-drawn map consisting of lines (which represent shorelines, roads, or rivers) and symbols for bridges and cities. Mapsee uses schemata and constraint satisfaction to interpret these maps. Schemata combine to form a *scene constraint graph*. Constraints are propagated by a network consistency algorithm, to specialize the labels associated with schemata. Schema label sets have a specialization hierarchy, stating, for example, that both islands and mainland are landmasses, and both landmasses and waterbodies are geosystems. A constraint provides information such as “if a geosystem has a component river-system, road-system or mountain-range, then the geosystem-label is constrained to be a landmass”.

The Anon system provides another example of schema-based diagram recognition [33]. The authors assert that CAD structures generated from mechanical engineering drawings embody composition and specialization hierarchies, which are naturally represented by schema-based systems. In the Anon system, engineering drawings are recognized using three techniques: a strategy grammar, schemata, and an image analysis library. At any given time, one schema is designated as the *controlling schema*. The controlling schema invokes the image analysis library, providing predictive information to guide the application of image analysis routines, and interpreting the results of the image analysis. The controlling schema encodes the image-analysis results as a token stream, which is passed up to the strategy grammar. The strategy grammar, an LR(1) grammar parsed by yacc, performs control actions (update the controlling schema, create a new schema, or designate a different schema to be the controlling one). Thus, spatial processing (where to look in the image, what to look for) is directed by the controlling schema, whereas symbolic processing (how to construct recognition hypotheses) is managed by the strategy grammar. The grammatical rules in Anon's strategy grammar are not intended to define a legal engineering drawing; rather, they specify strategies by which the various components of a drawing might be recognized.

The Anon system is aimed at low-level and intermediate-level diagram recognition. Anon analyzes an image to extract schemata representing drawing constructs. More global processes would be needed to integrate pieces of a drawing into a coherent whole. In contrast, the aforementioned Mapsee system does perform global processing, via constraint propagation. Mapsee does not perform segmentation and symbol recognition; symbols and line-segments are provided as input.

7. Kernel of a recognition system

We have discussed a variety of approaches to organizing the domain knowledge needed in a diagram recognition system: grammars and production rules, models of diagram generation, knowledge sources operating on a blackboard, and schema-based systems. It is wasteful to build every diagram recognition system from scratch. A recognition *kernel*, which makes common recognition algorithms available for general use, reduces the work involved in creating recognizers for a great variety of diagram notations. A recognition kernel does not solve the entire diagram recognition problem, but acts as a tool to be used in the construction of a complete system.

A variety of methods can be used to adapt and extend a recognition kernel to suit particular applications. Three publications on this topic are summarized here. So far, no recognition kernels are in widespread use. The development of widely-used recognition kernels would certainly help the field of diagram recognition to mature.

Pasternak describes an adaptable drawing-interpretation kernel that is implemented as a blackboard system [52]. Declarative geometrical constraints are applied to iteratively combine graphical objects into higher-level objects. The kernel supplies generally-applicable operations such as thresholding, line finding, and vectorization. Domain-specific knowledge bases must be added to configure the system for recognition of particular diagram notations

Graph-like diagrams (including schematics, flowcharts, and piping diagrams) are given general treatment in the recognition system of [44]. The system uses bottom-up processing to find lines, symbols, and text, with little reliance on domain-specific knowledge.

Satoh et al. describe a drawing-interpretation kernel that is adaptable to various drawing types through the addition of drawing-understanding rules [66]. Support for automatic rule inference is provided. Map recognition is used as an example. This system is directed at geometry-level understanding of a diagram. An example is finding dashed boundaries in a map.

8. Conclusion

Diagram recognition is a difficult problem, due to the need to represent notational conventions, handle noise and uncertainty, handle dialects, and cope with the complexity and variety of diagram notations that are in use. We have described several approaches to diagram recognition: grammars and production rules, recognition using a model of diagram generation, blackboard systems, schema-based systems, and recognition kernels. It is difficult to compare these approaches, to determine which one is most appropriate for a given recognition problem. One measure is expressive power: what computations can be expressed when using this approach? For an interesting discussion of the tension between expressiveness and efficiency in grammatical specification formalisms, see [45]. This does not help in choosing between blackboard systems and schema-based systems (for example), since both are Turing equivalent. The real need is for a convenient, readable, extensible representation of notational conventions. The literature provides us with experiences authors have had in using blackboards, schemata, production rules etc. It is difficult to compare this anecdotal evidence, to judge which approach is best. In the long run, we hope to see the emergence of a diagram recognition technology, which might be comparable to today's compiler technology, allowing rapid construction of recognition software, using well-understood and well-established methods.

References

[1] R. Anderson, "Two Dimensional Mathematical Notation," in *Syntactic Pattern Recognition, Applications*, K. S. Fu editor, Springer, 1977, 147-177.

[2] J. Arias, C. Lai, S. Chandran, R. Kasturi, A. Chhabra, "Interpretation of Telephone System Manhole Drawings," *Proc. Second Intl. Conf. Document Analysis and Recognition*, Tsukuba, Japan, Oct. 1993, 365–368.

[3] J. Arias, A. Chhabra, V. Misra, "A Practical Application of Graphics Recognition: Helping with the Extraction of Information from Telephone Company Drawings," in *Graphics Recognition — Algorithms and Systems*, Eds. K. Tombre and A. Chhabra, LNCS Vol. 1389, Springer, 1998, 314-321.

[4] D. Bainbridge, N. Carter, "Automatic Reading of Music Notation," in *Handbook of Character Recognition and Document Image Analysis*, Eds. H. Bunke and P. Wang, World Scientific, 1997, 583–603.

[5] H. Baird, D. Ittner, "Data Structures for Page Readers" *Proc. IAPR Workshop on Document Analysis Systems*, Kaiserslautern, Germany, Oct. 1994, 323-334.

[6] S. Baumann, "A Simplified Attributed Graph Grammar for High-Level Music Recognition," *Proc. Third Intl. Conf. on Document Analysis and Recognition*, Montreal, Canada, Aug. 1995, 1080–1083.

[7] A. Belkin, "Macintosh Notation Software: Present and Future," *Computer Music Journal*, **18** (1):53-69, 1994.

[8] A. Blackwell, Y. Engelhardt, "A Taxonomy of Diagram Taxonomies," *Proc. Thinking with Diagrams 98*, U. Wales, Aberystwyth, United Kingdom, August 1998, 60-70. Available at http://www.mrc-cbu.cam.ac.uk/projects/twd/mypapers/TwD98.html

[9] D. Blostein, H. Baird, "A Critical Survey of Music Image Analysis," in *Structured Document Image Analysis*, Eds. H. Baird, H. Bunke, and K. Yamamoto, Springer, 1992, 405-434.

[10] D. Blostein, A. Grbavec, "Recognition of Mathematical Notation," in *Handbook of Character Recognition and Document Image Analysis*, Eds. H. Bunke and P. Wang, World Scientific, 1997, 557–582.

[11] D. Blostein, L. Haken, "Using Diagram Generation Software to Improve Diagram Recognition: A Case Study of Music Notation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* **21** (11):1121-1136, Nov. 1999.

[12] D. Blostein, A. Schürr, "Computing with Graphs and Graph Transformation," *Software – Practice and Experience*, **29**(3), 1999, 197-217.

[13] L. Boatto, V. Consorti et al, "Detection and Separation of Symbols Connected to Graphics in Line Drawings," *11th Intl. Conf. on Pattern Recognition*, Delft, Netherlands, Sept. 1992, Vol. 2, 545–548.

[14] H. Bunke, "Attributed Programmed Graph Grammars and Their Application to Schematic Diagram Interpretation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, **4**(6):574-582, Nov. 1982.

[15] T. Chaundy, P. Barrett, C. Batey, *The Printing of Mathematics*, Oxford University Press, 1957.

[16] A. Chhabra, "Graphic Symbol Recognition: An Overview," in *Graphics Recognition — Algorithms and Systems*, Eds. K. Tombre and A. Chhabra, LNCS Vol. 1389, Springer, 1998, 68–79.

[17] P. Chou, "Recognition of Equations Using a Two-Dimensional Stochastic Context-Free Grammar," *Proc. SPIE Conf. on Visual Communications and Image Processing IV*, Philadelphia PA, 852-863, Nov. 1989.

[18] P. Chou and G. Kopec, "A Stochastic Attribute Grammar Model of Document Production and its use in Document Image Decoding," *Document Recognition II*, *SPIE Proceedings Series, Vol. 2422*, 1995, 66-73.

[19] G. Costagliola, A. De Lucia, S. Orefice, G. Tortora, "A Framework of Syntactic Models for the Implementation of Visual Languages," *Proc. 1997 IEEE Intl. Symposium on Visual Languages (VL'97)*, Capri, Italy, Sept. 1997, 58–65.

[20] W. Cushman, P. Ojha, and C. Daniels, "Usable OCR: What are the Minimum Performance Requirements?", *Proc. ACM SIGCHI 1990 Conference on Human Factors in Computing Systems*, Seattle, Washington, April 1990, 145-151.

[21] D. Dori, Y. Liang, J. Dowell and I. Chai, "Sparse-pixel recognition of Primitives in Engineering Drawings," *Machine Vision and Applications*, 6:69–82, 1993.

[22] D. Dori, A. Pnueli, "The Grammar of Dimensions in Machine Drawings," *Computer Vision, Graphics and Image Processing,* **42**:1–18, 1988.

[23] H. Fahmy, D. Blostein, "A Graph Grammar Programming Style for Recognition of Music Notation," *Machine Vision and Applications*, **6**(2):83-99, 1993.

[24] H. Fahmy, D. Blostein, "A Graph-Rewriting Paradigm for Discrete Relaxation: Application to Sheet-Music Recognition," *International Journal of Pattern Recognition and Artificial Intelligence*, **12**(6):763-799, Sept. 1998.

[25] C. Faure and Z. Wang, "Automatic Perception of the Structure of Handwritten Mathematical Expressions," in *Computer Processing of Handwriting*, Eds. R. Plamondon and C. Leedham, World Scientific, 1990, 337–361.

[26] K. S. Fu, *Syntactic Pattern Recognition and Applications*, Prentice Hall 1982.

[27] R. Futrelle, "Strategies for Diagram Understanding: Generalized Equivalence, Spatial/Object Pyramids and Animate Vision," *10th Intl. Conf. on Pattern Recognition*, Atlantic City, New Jersey, June 1990, 403–408.

[28] A. Grbavec, D. Blostein, "Mathematics Recognition Using Graph Rewriting," *Third Intl. Conf. on Document Analysis and Recognition*, Montreal, Aug. 1995, 417–421.

[29] W. Havens, A. Mackworth, “Representing Knowledge of the Visual World,” *IEEE Computer*, Oct. 1983, 90-96.

[30] IEEE Transactions on Pattern Analysis and Machine Intelligence, published by the IEEE Computer Society.

[31] International Journal on Document Analysis and Recognition, Springer.

[32] V. Jagannathan, R. Dodhiawala, L. Baum, Editors, *Blackboard Architectures and Applications*, Academic Press, 1989.

[33] S. Joseph, T. Pridmore, “Knowledge-Directed Interpretation of Mechanical Engineering Drawings,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, **14**(9):928-940, Sept. 1992.

[34] Journal of Visual Languages and Computing, Academic Press.

[35] T. Kanungo, R. Haralick, D. Dori, “Understanding Engineering Drawings: A Survey,” *Proc. Intl. Workshop on Graphics Recognition*, University Park, Pennsylvania, Aug. 1995, 119–130.

[36] H. Kato, S. Inokuchi, “The Recognition System of Printed Piano Music using Musical Knowledge and Constraints,” *Proc. IAPR Workshop on Syntactic and Structural Pattern Recognition,* Murray Hill NJ, June 1990, 231-248.

[37] H. Kato, S. Inokuchi, “The Recognition Method for Roughly Hand-Drawn Logical Diagrams Based on Utilization of Multi-Layered Knowledge,” *Proc. 10th Intl. Conf. on Pattern Recognition*, Atlantic City NJ, June 1990, 443-473.

[38] D. Knuth, “Mathematical Typography,” *Bulletin of the American Mathematical Society*, **1**(2), March 1979, 337-372.

[39] G. Kopec, P. Chou, “Document Image Decoding Using Markov Source Models,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, **16**(6):602-617, June 1994.

[40] G. Kopec, P. Chou, and D. Maltz, “Markov Source Model for Printed Music Decoding,” *Journal of Electronic Imaging*, **5**(1):7-14, Jan. 1996.

[41] L. Lamport, *LaTeX User's Guide & Reference Manual*, Addison-Wesley, 1986.

[42] E. Lank, draft paper, http://www.cs.queensu.ca/home/lank/dr.process.ps.gz.

[43] S. Lavirotte, L. Pottier, “Optical Formula Recognition,” *Fourth Intl. Conf. on Document Analysis and Recognition*, Ulm, Germany, Aug. 1997, 357–361.

[44] X. Lin, S. Shimotsuji, M. Minoh, T. Saki, “Efficient Diagram Understanding with Characteristic Pattern Detection,” *Computer Vision, Graphics, and Image Processing*, **30**:84-106, 1985.

[45] K. Marriott, B. Meyer, K. Wittenburg, “A Survey of Visual Language Specification and Recognition,” in *Visual Language Theory*, Eds. K. Marriott, B. Meyer, Springer, 1998, 5-85.

[46] J. McDaniel, J. Balmuth, “Automatic Interpretation of Chemical Structure Diagrams,” in *Graphics Recognition — Methods and Applications*, Eds. R. Kasturi and K. Tombre, LNCS Vol. 1072, Springer, 1996, 148–158.

[47] I. Mulder, A. Mackworth, W. Havens, “Knowledge Structuring and Constraint Satisfaction: The Mapsee Approach,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, **10**(6):866-879, Nov. 1988.

[48] G. Novak, W. Bulko, “Diagrams and Text as Computer Input,” *Journal of Visual Languages and Computing*, **4**, 1993, 161-175.

[49] L. O’Gorman, R. Kasturi, *Document Image Analysis*, IEEE Computer Society Press, 1995.

[50] M. Okamoto, B. Miao, “Recognition of Mathematical Expressions by Using the Layout Structure of Symbols,” in *Proc. First Intl. Conf. on Document Analysis and Recognition*, Saint Malo, France, Sept. 1991, 242-250.

[51] A. Okazaki, T. Kondo, K. Mori, S. Tsunekawa, and E. Kawamoto, “An Automatic Circuit Diagram Reader with Loop-Structure-Based Symbol Recognition,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, **10**(3):331–340, May 1988.

[52] B. Pasternak, “Processing Imprecise and Structural Distorted Line Drawings by an Adaptable Drawing Interpretation Kernel,” *Proc. IAPR Workshop on Document Analysis Systems*, Kaiserslautern, Germany, Oct. 1994, 349-363.

[53] B. Pasternak, “The Role of Taxonomy in Drawing Interpretation,” *Proc. Third Intl. Conf. Document Analysis and Recognition*, Montreal, Canada, Aug. 1995, 799–802.

[54] B. Poirier, M. Dagenais, “An Interactive System to Extract Structured Text from a Geometrical Representation,” *Proc. Fourth Intl. Conf. on Document Analysis and Recognition*, Ulm, Germany, Aug. 1997, 342–346.

[55] Proc. Annual Symposia on Document Analysis and Information Retrieval, Las Vegas, 1992, 1993, 1994, 1995, 1996, 1997, sponsored by the University of Nevada.

[56] Proc. CHI Conferences, Human Factors in Computing Systems, ACM press.

[57] Proc. IAPR Workshops on Document Analysis Systems.

[58] Proc. IAPR Workshops on Graphics Recognition.

[59] Proc. IEEE Symposium on Visual Languages, IEEE Computer Society Press.

[60] Proc. Intl. Confs. on Document Analysis and Recognition, sponsored by IAPR, IEEE.

[61] M. A. Rahgozar, R. Cooperman, “A Graph-based Table Recognition System,” *Document Recognition III*, SPIE Proceedings Series, Vol. 2660, 1996, 192-203.

[62] J. Ramel, N. Vincent, H. Emptoz, “A Coarse Vectorization as an Initial Representation for the Understanding of Line Drawing Images,” in *Graphics Recognition — Algorithms and Systems*, Eds. K. Tombre and A. Chhabra, LNCS Vol. 1389, Springer, 1998, 48–57.

[63] G. Read, *Music Notation: A Manual of Modern Practice (2nd Edition)*, Taplinger Publishing, New York, NY, 1979.

[64] D. Roush, “Music Formatting Guidelines,” Technical Report OSU-CISRC-3/88-TR10, Department of Computer and Information Science, The Ohio State University, 1988.

[65] K. Ryall, S. Shieber, J. Marks, M. Mazer, “Semi-Automatic Delineation of Regions in Floor Plans,” *Proc. Third Intl. Conf. on Document Analysis and Recognition*, Montreal, Canada, Aug. 1995, 964–969.

[66] S. Satoh, H. Mo and M. Sakauchi, “Drawing Image Understanding System with Capability of Rule Learning,” *Proc. Second Intl. Conf. on Document Analysis and Recognition*, Tsukuba, Japan, Oct. 1993, 119–124.

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

[68] SmartScore music-notation recognition software by Musitek, http://www.harmony-central.com/Newp/1999/SmartScore.html

[69] A. Smithies, K. Novins, J. Arvo, “A Handwriting-Based Equation Editor,” *Proc. Graphics Interface '99*, sponsored by the Canadian Human-Computer Communications Society, Kingston, Ontario, Canada, June, 1999, 84-91.

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

[71] H. Twaakyondo and M. Okamoto, “Structure Analysis and Recognition of Mathematical Expressions,” *Proc. Third Intl. Conf. on Document Analysis and Recognition*, Montreal, Canada, August 1995, 430–437.

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

[73] C. Wang, S. Srihari, “A Framework for Object Recognition in a Visually Complex Environment and its Application to Locating Address Blocks on Mail Pieces,” *Intl. Journal of Computer Vision*, **2**:125-151, 1989.

[74] Z. Wang, C. Faure, “Structural Analysis of Handwritten Mathematical Expressions,” *Proc. Ninth Intl. Conf. on Pattern Recognition*, 32–34, Rome, Italy, Nov. 1988.

[75] R. Zanibbi, “Recognition of Mathematics Notation via Computer Using Baseline Structure”, M.Sc. Thesis, Dept. Computing and Information Science, Queen’s University, Kingston, Ontario, Canada, Jan. 2000.