

Symbolic Pruning in a Structural Approach to Engineering Drawing Analysis

Tom Henderson and Lavanya Swaminathan
School of Computing
University of Utah
Salt lake City, Utah 84112, USA
tch@cs.utah.edu

Abstract

Interpretation of paper drawings has received a good deal of attention over the last decade. Progress has also been made in related areas such as direct interpretation of human drawings (HCI), search and indexing of graphics databases, and knowledge representation in the domain of graphics and drawing understanding. One of the most interesting applications in this domain is the analysis of semantics in engineering drawings.

We propose a structural modeling approach combined with a nondeterministic agent system to produce interpretations of scanned images of CAD drawings. We allow a broad set of thresholds during the image analysis and show how this large search space can be efficiently pruned by taking advantage of constraints from the model or specific application.

1. Introduction

The semantic interpretation of industrial drawings is a very difficult problem. Tompore summarizes the state of the field for CAD drawings [11] and as a document image analysis problem [12, 13]. An earlier overview by Haralick [7] gives insight into the scale of the technical drawing problem (e.g., about 250 million drawings are generated annually), as well as a discussion of performance measurement. A major problem pointed out by these reviewers is that there is a dearth of work relating high-level models of documents, drawings or graphics to the various levels of analysis. The goal of technical drawing analysis is to interpret the contents of an image, such as that shown in in Figure 1.

Several systems have been developed along these lines, although none works well in an automated fashion. Various problems arise; e.g., when text touches graphics or when there is noise in the scanned image or when the thresholds of the image-analysis codes support only a few classes of drawings. One notable system is CELESSTIN [1, 9]; how-

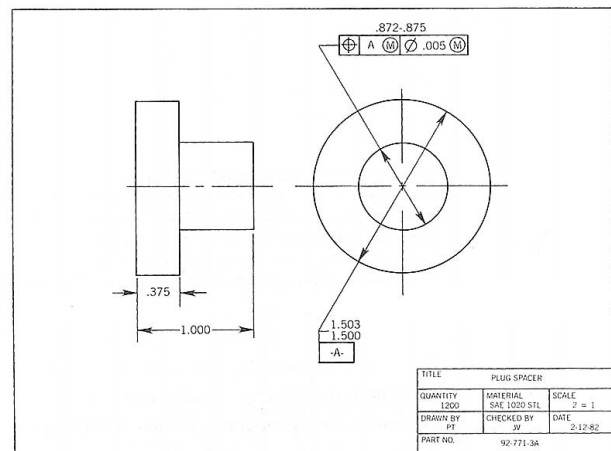


Figure 1. Part of a Scanned Technical Drawing

ever, as pointed out by Haralick, CELESSTIN suffers from the fact that it is based on a hierarchical rule system and has many rules; furthermore, whatever interpretive models the rules instantiate exist only in the thresholds and logic of the hand-coded rules.

In the standard form of analysis, the scanned drawing is digitized, noise is removed, text and graphics are recognized, graphics is vectorized, dimensions are extracted and the whole drawing is analyzed by applying knowledge rules. Each of these steps normally uses just a single set of thresholds. Our goal is to allow a wide range of thresholds to generate a potentially large search space (hopefully including the correct components), and then to apply domain constraints to prune the space.

1.1. A High Level Model for Technical Drawings

We define the layout of the technical drawings in terms of a structural grammar (see [6] for an overview of the field). The components of a technical drawing, such as *text*,

pointer arrows, dimensions, etc., are defined as either terminal symbols in the grammar (e.g., *line segments*), or nonterminal symbols (e.g., *dimension description*) defined in terms of rewrite rules that describe their sub-components and the relations that must be satisfied between the sub-components.

The combination of the structural approach and nondeterministic agent analysis is a natural fit and provides a means to control the exploration of the exponentially large search space.

Annotation models have been presented in the literature (e.g., see [2, 3, 4, 5]), and we have extended these ideas and developed a novel approach to high-level modeling. We have also implemented the basic image analysis tools to extract text, graphics, graphical primitives. These form the basis actions for the agent architecture approach.

1.2. Agent Architecture

Agents are independent software processes with the following properties: autonomous (react to environment), have state (beliefs, commitments, etc.), persistent (process never terminates), can communicate (send and receive messages related to effort), and perform some action (have abilities to analyze and create data). For more complete accounts, see [8, 14].

An agent architecture is a software architecture for decision making with intelligent (flexible) processes embedded within it. The agents may be proactive or reactive, and should cooperate (including communicate) to achieve a goal.

We explore the use of nondeterministic agent systems (NDAS) to achieve a more flexible system for technical drawing analysis. They are called nondeterministic because the agents explore alternative parts of the solution space simultaneously, and every agent works to produce some result which may or may not contribute to the final result. The final result derives from only a subset of the work put in by all the agents. We explore nondeterminism in this problem domain since deterministic systems usually make irrevocable decisions (e.g., threshold selection) that eliminate possible solutions. The technical drawing problem domain contains many factors that vary with the drawing: thresholds, text fonts and size, noise levels, etc., and this variation makes it interesting to explore the possible solution space dynamically and in a breadth-first way.

We demonstrate this through the design and analysis of the NDAS system and provide experimental results to support the claims. (See [10] for more detail.)

2. A Structural Model for Engineering Drawings

Higher-level models describe the semantics of the drawings, and as such involve determining the relations between the primitives of the drawing and their meanings. We use graph models (semantic nets), which are explored through a grammatical paradigm. After the document is digitized, vectorized and the connected components extracted, the result is a set of image primitives such as segments, arcs, arrows and text blocks. Based upon the relationships that exist between these primitives, a structural model is used to recognize important features like dimensions, annotations, and legends in the document.

Figure 2 shows an instance of dimensioning taken from the scanned image of an actual technical drawing. A structural rewrite rule for this dimension set is:

$dimension_set := ptr_ray1 + ptr_ray2 + text$

where

$collinear(ptr_ray1, ptr_ray2)$
 $collinear(ptr_ray1, text)$
 $collinear(ptr_ray2, text)$
 $[between(ptr_ray1, text, ptr_ray2)]$

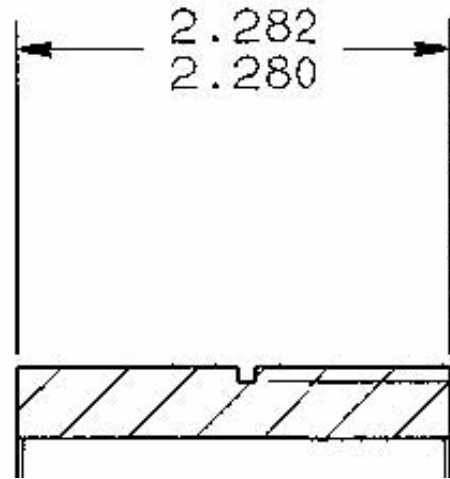


Figure 2. Example of dimension set

In order to organize the activity of the analysis agents, we have developed an engineering drawing model comprised of structures typically found in such drawings, and relations between those structures. This approach is based on structural and syntactic shape methods (e.g., see [6]); however, our method is novel in that it allows for the natural application of the NDAS agent system to recover the desired structures from an image.

Terminal structures correspond to the terminal symbols of a shape grammar, and include: **text**, **box**, **pointer_ray**,

pointer_line, **pointer_arc**, **circle**, **line_segment**, and **graphic**.

Higher-level structures correspond to the nonterminal symbols in a shape grammar, and can be described by rewrite rules which define sub-structures which comprise the new structure, and the relations that must exist between the sub-structures. Some examples of these include:

- *dimension*: **text** with a *symmetric_pointer_pair*.
- *dimension_set*: a *dimension* enclosed by a *symmetric_line_pair* or an *angle_dimension* enclosed by an *asymmetric_line_pair*.
- *dimension_description*: a *dimension_set* and corresponding **graphic**. Complete dimension information, including the graphic being described.

2.1. Analysis Complexity Reduction

Given a strategy of generating as much of the search space as possible, it is necessary to find ways to reduce the number of alternatives; however, this must be done in a systematic and correct way. We have developed **symbolic pruning** to exploit the formal aspects of the grammar and the algebraic relations to eliminate duplications and redundancies.

Our goal is to determine the complexity of the total number of possible symbols for the given grammar generated from a set of ground structures with respect to a specific image. Let G be a grammar, and for the current context, let it represent the rewrite rules. We define a *production sequence* as a correct application in some order of the rewrite rules of G . Thus:

$$ps = \prod_{j=1}^n i_j$$

defines a production sequence, ps , where i_j is the index of the j^{th} production (repeating an index is allowed).

Symbol redundancy of a vocabulary symbol, v , called $SR(v)$, is the count of the number of distinct production sequences that produce the symbol. This is the same as the number of ways the ground structures can be mapped onto the terminal symbols to produce the symbol v . For a terminal symbol, a , we have

$$SR(a) = |g|$$

where $|g|$ is the number of ground structures of this terminal symbol.

In order to calculate $SR(v)$ for a non-terminal symbol v , we introduce the following notion. An **0-form rewrite rule** is one with only terminal symbols on the right hand side. An **0-form grammar** is one with only 0-form rewrite rules. Algorithm 0-form produces an 0-form grammar from a general grammar.

Algorithm 0-form

On input: a general grammar, G
 On output: an 0-form grammar, $G2$
 $G1 \leftarrow G$
 $G2 \leftarrow$ empty set

whenever there exists a rewrite rule, R , in $G1$ such that $lhs(R)$ is a non-terminal and there are only terminals in the $rhs(R)$:

Add R to $G2$
 For every R' in $G1$ such that $lhs(R)$ is in the $rhs(R')$
 Replace each occurrence of $lhs(R)$ in R' with $rhs(R)$
 and call new rule R''
 Add R'' to $G1$

Note that this applies to non-recursive grammars, but if there is a recursive rewrite rule, then it can be easily tagged, and the user can set a maximum depth for it. For example, any number of section lines can occur in a technical drawing so long as they are parallel.

For a terminal symbol w and a rewrite rule R , define $count(w,R)$ to be the number of times that w appears on the right hand side of R . Then, for a rewrite rule in 0-form and a non-terminal symbol, v , we have:

$$SR(v) = \sum_R \prod_w [SR(w)(SR(w) - 1) \dots (SR(w) - count(w, R) + 1)]$$

where R is in the rewrite rules, and the sum is taken over all rules with v in the left hand side, and w is a distinct symbol in the right hand side of R . If any summand is negative or zero, then it is not added in; if all summands are negative or zero, then v cannot be produced.

Consider the following example grammar for a Parallel Pair of Segments (PPS , where $SLP1$ and $SLP2$ correspond to the individual segments in the pair); we assume that all are symmetric:

```
G = { PPS -> SLP1 + SLP2
      SLP1 -> SLP
      SLP2 -> SLP
      SLP -> line1 + line2
      line1 -> line_seg
      line2 -> line_seg
    }
```

The $G1$ set produced by Algorithm 0-form is:

```
G1 = { PPS -> SLP1 + SLP2
```

```

SLP1 -> SLP
SLP2 -> SLP
SLP -> line1 + line2
line1 -> line_seg
line2 -> line_seg
SLP -> line_seg + line2
SLP -> line_seg + line_seg
SLP1 -> line_seg + line_seg
SLP2 -> line_seg + line_seg
PPS -> line_seg + line_seg + SLP2
PPS -> SLP1 + line_seg + line_seg
PPS -> line_seg + line_seg
      + line_seg + line_seg
}

```

The G_2 set produced by Algorithm 0-form is:

```

G2 = { line1 -> line_seg
      line2 -> line_seg
      SLP -> line_seg + line_seg
      SLP1 -> line_seg + line_seg
      SLP2 -> line_seg + line_seg
      PPS -> line_seg + line_seg
            + line_seg + line_seg
}

```

Consequently, we have the following results:

```

Suppose that   SR(line_seg) = 4
then
                SR(line1) = 4
                SR(line2) = 4
                SR(SLP)   = 12
                SR(SLP1)  = 12
                SR(SLP2)  = 12
                SR(PPS)   = 24

```

Note that the symbolic redundancy depends on the number of ground structures (terminal symbols in image). For example:

```

Suppose that   SR(line_seg) = 2
then
                SR(line1) = 2
                SR(line2) = 2
                SR(SLP)   = 2
                SR(SLP1)  = 2
                SR(SLP2)  = 2
                SR(PPS)   = 0   fails!

```

There are not enough ground structures to satisfy the need for four distinct terminal symbols to produce nonterminal symbol PPS .

The symbol redundancy measure is a worst case estimate for the number of redundant symbols produced since for a

specific set of ground structures, the required relations between the terminal symbols may not hold; this would prevent the generation of the nonterminal symbol. However, this does give a useful measure of the complexity of the analysis to be performed (and, in fact, makes it possible to know when there are too few ground structures to produce a complete parse).

The symbol redundancy measure of the number of ways in which a given symbol can be produced allows an analysis to determine if any of the redundant production sequences can be eliminated. There are two approaches to actually eliminate redundancies:

- compile out one rewrite rule: start symbol, with relations between terminals; eliminate combinations where only relations are symmetric; synthesize the relations for this rewrite rule.
- allow only one production sequence from a symmetric relation equivalent set when they differ only in the assignment of the same symbol in right hand side.

We describe a solution using the second approach. To do this requires the following definition: two production sequences are *symmetric relation equivalent* if they differ only in assignment of same symbol in right hand side of rewrite rule. Likewise, a set of production sequences are symmetric relation equivalent if they are all pairwise symmetric relation equivalent. One approach to symbolic pruning then is to find sets of symmetric relation equivalent production sequences and to allow only one during the structural analysis. This is a parse time operation.

3. Experiments

To determine how well the structural analysis is performed, we applied NDAS to the image in Figure 1; all four dimension sets were found. Table 1 gives the symbol redundancy for the vocabulary symbols in our grammar for engineering drawings. Several examples were run with good success and are reported in [10].

4. Conclusions and Future Work

The results of these experiments are encouraging. The structural analysis proceeds correctly, and the agent system seems to be robust and explores much of the interesting part of the search space. The percentage of dimensions found ranges from 16.6 % for noisy images to 100 % for clean images, and the pruning methods lead to orders of magnitude reductions in the number of symbols considered during the analysis.

We have demonstrated that the nondeterministic agent system (NDAS) combined with a complementary structural

<i>Symbol</i>	<i>worst case</i>	<i>after pruning</i>
line_segment	21	21
pointer_ray	4	4
text	2	2
circle	1	1
box	1	1
pointer_ray1	4	4
pointer_ray2	4	4
line_segment1	21	21
line_segment2	21	21
line_segment3	21	21
text1	2	2
text2	2	2
text_comb	2	0
text_£nal	4	2
symmetric_pointer_pair_in	12	0
symmetric_pointer_pair_out	12	2
dimension_rays_in	48	0
dimension_rays_out	48	2
dimension	96	2
dimension_set	40320	2
pointer_ray_extn	84	0
check_sign	420	0
check_pair	1680	0
dimension_description	> 100000	2
text_in_box	4	0
text_in_box1	4	0
text_in_box2	4	0
text_in_box3	4	0
text_in_box4	4	0
one_datum_ref	0	0
datum_ref	0	0
datum_below_text	0	0
dashed_lines	7980	3
dash_lines1	7980	3
dash_lines2	7980	3
circle_center_dim	> 100000	1

Table 1. Symbol Redundancy for Grammar

modeling approach can achieve a coherent analysis of annotations in technical drawings. This process can be made more efficient by means of symbolic pruning as introduced here. The experimental data indicates that this is a feasible approach and gives a firm basis upon which to define error, precision and performance measures.

Acknowledgment: This work was supported in part by ARO grant number DAAD19-01-1-0013.

References

- [1] C. Ah-Soon and K. Tombre. A step towards reconstruction of 3-d cad models from engineering drawings. In *Proceedings 3rd International Conference on Document Analysis and Recognition*, pages 331–334, Montral (Canada), 1995.
- [2] S. Collin and D. Colnet. Syntactic analysis of technical drawing dimensions. *International Journal of Pattern Recognition and Artificial Intelligence*, 8(5):1131–1148, 1994.
- [3] D. Dori. A syntactic/geometric approach to recognition of dimensions in engineering drawings. *Computer Vision, Graphics and Image Processing*, 47:271–291, 1989.
- [4] D. Dori and A. Pnueli. The grammar of dimensions in machine drawings. *Computer Vision, Graphics and Image Processing*, 42:1–18, 1988.
- [5] A. Habed and B. Boufama. Dimension sets in technical drawings. In *Proceedings of Vision Interface*, pages 217–223, Trios-Rivieres, CA, 1999.
- [6] T. C. Henderson and A. Samal. Shape grammar compilers. *Pattern Recognition*, 19(4):279–288, 1985.
- [7] T. Kanungo, R. M. Haralick, and D. Dori. Understanding engineering drawings: A survey. In *Proceedings of First IARP Workshop on Graphics Recognition*, pages 217–228, University Park, P.A, 1995.
- [8] V. Subrahmanian. *Heterogeneous Agent Systems*. MIT Press, Cambridge, MA, 2000.
- [9] K. T. Suzanne Collin and P. vaxiviere. Don't tell mom i'm doing document analysis; she believes i'm in the computer vision £eld. In *Proceedings of 2nd International Conference on Document Analysis and Recognition*, pages 619–622, Tsukuba Science City (Japan), October 1993.
- [10] L. Swaminathan. Agent-based engineering drawing analysis. Master's thesis, University of Utah, Salt Lake City, Utah, December 2002.
- [11] K. Tombre. Analysis of engineering drawings: State of art and challenges. In *Graphics Recognition - Algorithms and Systems*, volume 1389 of *Lecture Notes in Computer Science*, pages 257–264, Springer Verlag, April 1998.
- [12] K. Tombre. Graphics documents: Achievements and open problems. In *Proceedings of the 10th Portuguese Conference on Pattern Recognition*, Lisbon (Portugal), 1998.
- [13] K. Tombre. Ten years of research in the analysis of graphics documents: Achievements and open problems. In *Proceedings of the 10th Portuguese Conference on Pattern Recognition*, Lisbon (Portugal), 1998.
- [14] G. Weiss. *Multi-Agent Systems*. MIT Press, Cambridge, MA, 1999.