Agent-Based Engineering Drawing Analysis

Thomas C. Henderson and Lavanya Swaminathan
School of Computing
University of Utah
4 February 2002

UUCS-02-008

Abstract

Interpretation of paper drawings has received a good deal of attention over the last decade. 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 have also seen advances.. One of the most interesting applications in this domain is the analysis of semantics in engineering drawings.

Although several sophisticated automatic systems have been developed, for example, the CELESSTIN system in France, there are still significant problems in their application: (1) they are linear and do not allow backtracking solutions, (2) they are typically flat rule-based systems with many rules, and (3) the knowledge models are embedded directly in the rules and are unavailable in a higher-level form. We still need a more automatic, self-contained, less complex and robust system involving minimal human intervention. The use of autonomous agents in the field of image analysis is just starting.

Our thesis is that a set of nondeterministic agents provides the necessary methodology to address the shortcomings of previous systems. We explore the organization, communication and high-level knowledge representation of a set of agents designed to perform engineering drawing analysis. This permits the exploration of interesting parts of the search space, especially when combined with good pruning and focus mechanisms.

1 Overview

The semantic interpretation of industrial drawings is a very difficult problem. Tombre summarizes the state of the field for CAD drawings [**?**] and as a document image analysis problem [**?**]. An earlier overview by Haralick [**?**] 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.

1.1 Technical Drawing Analysis

The goal of technical drawing analysis is to interpret the contents of an image. Applications are as diverse as: (1) CAD drawings (2D to 3D), (2) sketch interpretation (HCI), (3) indexing and searching (e.g., technical DB's), and (4) technical document analysis. Figure 1 shows part of a scanned technical drawing; the standard paradigm for technical drawing analysis is to threshold, segment, vectorize, interpret and reconstruct 3D.

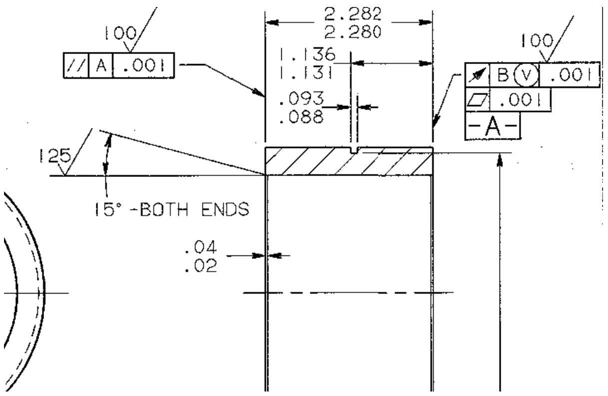

Figure 1: Part of a Scanned Technical Drawing

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. One notable system is CELESSTIN [**?**]; however, 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 model the rules instantiate exists only in the thresholds and logic of the hand-coded rules.

1.2 Agent Architecture for Analysis

Agents are independent software processes with the following properties: (1) autonomous (react to environment though), (2) have state (beliefs, commitments, etc.), (3) persistent (process never terminates), (4) can communicate (send and receive messages related to effort), and (5) perform some action (have abilities to analyze and create data). (See [**?**].) 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 and communicate to achieve a goal.

We propose the use of nondeterministic agent systems (NDAS) to achieve a more flexible system for technical drawing analysis. They are called nondeterministic because every agent works independently and asynchronously to produce some result which may or may not contribute to the final result which is only a subset of the work produced by all the agents. Our approach exploits nondeterminism to avoid irrevokable decisions that eliminate possible solutions. The technical drawing problem domain contains many factors that vary with the drawing: necessary thresholds, text fonts and size, noise levels, etc., and this variation makes it necessary to explore the possible solution space dynamically.

The issues that we intend to explore using the NDSA approach include: (1) *the complete analysis of technical drawing annotations* (this includes the ability to recognize dimensioning, features and their annotations, tolerances, and references to nomenclature), (2) *the functional and geometric interpretation of technical drawings*, (3) *threshold sensitivity analysis* (determination of the relationship between a change in threshold and its impact on the analysis process and result), and (4) *error, precision, robustness and performance analysis* (an engineering account of these aspects of a system would be extremely useful in making cost performance tradeoff decisions. Most results in the literature do not define very carefully what it means for a system to work nor what error is).

Annotation models have been developed (e.g., see [**?**]); we use semantic nets for high-level models. We have implemented basic image analysis tools to extract text, graphics, etc. These form the basis actions for the agent architecture approach.

The thesis of this work is that such models may be realized through a set of software agents acting independently and in parallel to ultimately produce a coherent analysis.

2 Nondeterministic Agent Systems

We have begun to explore the: (1) organization, (2) communication, (3) constraint handling, and (4) higher-level modeling capabilities of NDAS, using the analysis of technical drawings as the application domain (see Figure 2).

The agents look for certain kinds of objects in their field of action, and when the appropriate conditions hold, the agent will act and produce another set of objects. These objects may be desired data objects (e.g., a segmented image) or new agents (e.g., to produce a better analysis based on genetic algorithms). The idea is that the agents work independently and in parallel and continue to work until conditions cause them to become quiescent. Such an approach allows for feedback loops as previously processed data may reappear after modification by a higher-level process. This also allows a coarse to fine improvement as more work of broader context is completed. Agents may also monitor the activity of other agents and measure performance or provide data for consumption by the (human) user. Of course, the user may participate as an agent as well! (Note that it is straightforward to define the agents

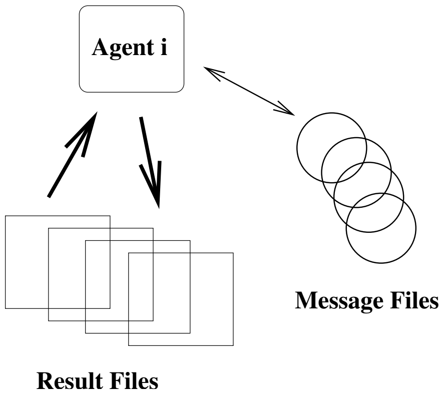

Figure 2: Nondeterministic Agent System for Technical Drawing Analysis.

so as to capture the standard processing paradigm.)

The nondeterministic agents are a collection of independent Unix processes. There are two aspects of organization: internal and external.

2.1 Internal Agent Organization

Each agent is written as a C++ program which watches its current executing environment (directory) for the existence of certain files or processes and then takes actions as specified by its internal code. This can be described by the following set of actions: (1) *Monitor*: watch for the existence of files, processes, etc., (2) *Action Program*: executable actions to take, and (3) *Wrapup*: cleanup files, check termination conditions, etc.

2.2 External Organization

The organization between agents includes how they announce their existence, if necessary, file protocols, etc. In addition, this includes the variety of agents that the user may wish to define; e.g., facilitation agents, performance analysis agents, process control agents, image analysis agents, feature analysis agents, higher-level model agents, etc.

Communication between agents is also an issue of their organization. and is achieved through files; for example, each agent creates a file which describes itself and its internal organization at a high level. A message protocol is defined. This includes the medium (e.g., files), the syntax (e.g., Knowledge Query and Manipulation Language – KQML), and semantics (e.g., ontology, etc.). Such communication includes: (1) sender, (2) receiver ID (may be broadcast or single recipient), (3) language (e.g., XML), (4) ontology (defines the meaning of the syntactic expressions), and (5) file name (where message can be found).

Ontology is similar to database schema and gives a specification of the objects, concepts and relationships that exist in the domain of interest. Concepts are represented by objects and relations between objects.

One of the aspects of engineering drawing analysis which requires an efficient mechanism is the determination of relations between the various entities in and extracted from an image. Pixels are related to text and graphics primitives, and primitives have relations like *near*, or *parallel*, etc.

2.3 Semantic Networks

Higher-level models involve descriptions of 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). After the document is digitized, vectorized and the connected components extracted, the result is a set of graphical primitives such as segments, arcs, arrows and text blocks. Based upon the relationships that exist between these primitives, a semantic net model can be used to recognize important features like dimensions, footnotes, legends in the document. E.g., as a higher-level structure, consider an *annotated graphic*:

$$Graphic(g) \wedge Pointer(p) \wedge Annotation(a) \wedge$$

$$PointsAt(p,g) \wedge (PointsFrom(p,g) \vee PointsAt(p,a))$$

3 Experiments

A prototype system has been developed which includes the following agents: (1) basic noise removal, (2) image inversion, (3) component labeling, (4) text and graphics separation, (5) text region extraction, (6) line segment detection, and (7) parallel line detection.

As a sample task, we are working on the determination of parallel line segments. For the baseline case, we have a Matlab code implemented in the standard way which produces a set of parallel line segments for a given image. We compare those results with the NDAS output in which multiple outputs, corresponding to various choices of parameters, are allowed for each processing step. Our preliminary results are that NDAS produces a wider range of results and appropriate agents can be devised to select a qualitatively better result based on knowledge of the particular domain.

The current system uses a process graph (see Figure 3) to evaluate the agents' work. The search space is pruned, both by removing poor quality results, as well as by comparing output files with each other and inhibiting agent sequences. In the case studied here, if agent operations are commutative, as thresholding and inverting the image happen to be, then the threshold agent learns not to work on files produced by the invert agent, since the two results are identical.

Acknowledgment: This work was supported in part by NSF grant STC-EIA-8920219 and ARO grant ARO-DAAD19-01-1-0013.

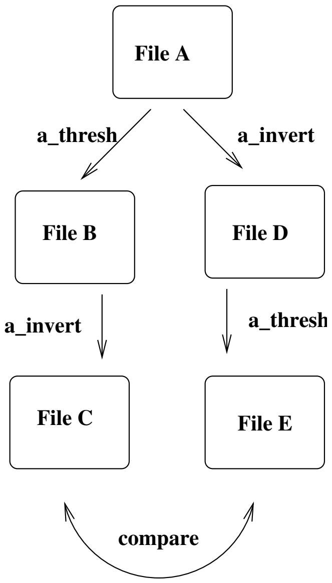

Figure 3: Process Graph.