1-1-1997

A System for Recognizing a Large Class of Engineering Drawings

Yuhong Yu
Lucent Technologies, Inc., Naperville, IL, yuhongyu@lucent.com

Ashok Samal
University of Nebraska - Lincoln, asamal1@unl.edu

Sharad C. Seth
University of Nebraska - Lincoln, seth@cse.unl.edu

A System for Recognizing a Large Class of Engineering Drawings

Yuhong Yu, Ashok Samal, *Member*, *IEEE*, and Sharad C. Seth, *Fellow*, *IEEE*

Abstract—We present a system for recognizing a large class of engineering drawings characterized by alternating instances of symbols and connection lines. The class includes domains such as flowcharts, logic and electrical circuits, and chemical plant diagrams. The output of the system, a netlist identifying the symbol types and interconnections, may be used for design simulation or as a compact portable representation of the drawing. The automatic recognition task is divided into two stages: 1) Domain-independent rules are used to segment symbols from connection lines in the drawing image that has been thinned, vectorized, and preprocessed in routine ways. 2) A drawing understanding subsystem works in concert with a set of domain-specific matchers to classify symbols and correct errors automatically. A graphical user interface is provided to correct residual errors interactively and to log data for reporting errors objectively. The system has been tested on a database of 64 printed images drawn from text books and handbooks in different domains and scanned at 150 and 300 dpi resolution.

Index Terms—Symbolic drawings, flow diagrams, segmentation and labeling, domain independence, automatic and interactive error correction.

1 Introduction

There is a well recognized need to convert the paper-based designs and diagrams to object-oriented format, particularly in engineering projects where both the quantity and archival period of drawings involved is large. Many of these drawings were originally drawn on paper, yet they are still actively used, and are periodically updated. Conversion of these types of drawings into a high-level description that can be easily manipulated by computers and humans would reduce the time for data entry and modifications. It would further provide a basis for retrieving and accessing documents based on high-level queries.

A system to convert a large subclass of engineering drawings to high-level description is presented here. Drawings in this subclass are characterized by dichotomy of symbols and connection lines (CLs), alternation of symbols and CLs, well-defined inputs and outputs, and flow of signals, chemical material, or program control that originates at input and terminates at output. A large number of traditional engineering drawings and diagrams fall into this class of drawings, called flow drawings. Examples include logic circuit diagrams, electrical circuit diagrams, chemical plant flow diagrams, program flowcharts, pipe and instrument diagrams, wiring diagrams, PERT charts, and entity-relationship charts.

The high-level description produced by the system includes the symbols and their location in the drawing and the interconnection between the symbols. This information can be expressed as a graph, where nodes and edges in the graph represent the symbols and connection lines of the flow drawing, respectively. Logic and electric-circuit designers use text versions of the graphical representation, called the netlist, to exchange design data among themselves and between programs. Once a netlist description is derived, it can be used in many ways: Inventories can be compiled automatically; logic design verification becomes possible once symbols are associated with logic functions; and automated tools can be used to analyze performance of the system represented by the drawing.

Described in this paper is a fully operational system with many unique features. A rule based approach is used first to separate the potential symbols from potential connection lines without knowledge of the type of drawing under consideration. Simple generic rules are found to be quite effective for this purpose.

Next, symbols are identified during a traversal of the drawing, in which potential symbols are matched against a domain-specific library. Some segmentation errors also get corrected during this step. The modular organization of the system and the symbol library allows conversion of a different type of drawing by just the replacement of the current library. The system has been extensively tested on drawings of four different types, derived from many sources and scanned at different resolutions. An interactive module allows correction of residual errors by human operators. As the errors are corrected, this module logs accurate data on errors that can be used in future to incorporate a form of learning.

The top-level architecture of the system is shown in Fig. 1.[1] The document is first scanned and vectorized using commercial software. Then, a series of operations, applied during the preprocessing step, attempt to remove artifacts

- *Y. Yu is with Lucent Technologies, Inc., 2000 Naperville Rd., Naperville, IL 60540-0000. E-mail: yuhongyu@lucent.com.*
- *A. Samal and S.C. Seth are with the Department of Computer Science and Engineering, 115 Ferguson Hall, University of Nebraska, Lincoln, NE 68588-0115. E-mail: {samal, seth}@cse.unl.edu.*

Manuscript received 22 Feb. 1996;. Recommended for acceptance by R. Kasturi. For information on obtaining reprints of this article, please send e-mail to: transpami@computer.org, and reference IEEECS Log Number 105308.

1. Diagrams in this paper are drawn as data flow diagrams. See [3] for a detailed discussion of this and other types of models.

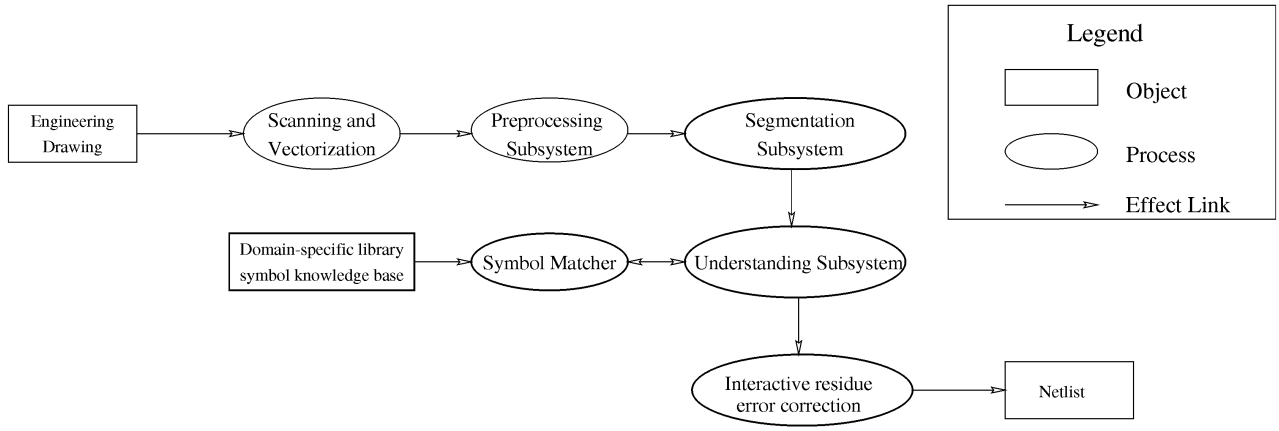


Fig. 1. Overview of the flow drawing understanding system.

introduced during scanning and vectorization. Various parameters used in the preprocessing steps have been carefully correlated to the scanning resolution.

Segmentation and understanding subsystems are the core modules of the system. Segmentation separates the symbols from the connection lines, relying on a set of generic rules, such as, a connection line is almost always composed of horizontal and vertical segments; and it always connects exactly two symbols.

The understanding subsystem produces a high-level interpretation of the drawing from the results of segmentation. It is here that the domain-dependent information is brought to bear on drawing interpretation. The shape and description of the symbols vary from one type of drawing to another. Such knowledge is isolated from the rest of the system and is placed in a library. The library organization allows a hierarchical matcher to identify symbols in the drawing with variable confidence. The automatic interpretation phase starts from symbols that have been identified with a high confidence value and recursively looks at other connected symbols. Symbols recognized with too low a confidence value cause a reexamination of a portion of the original segmentation through a modification of the classical split-and-merge algorithm. Some segmentation errors get corrected in this way by use of domain-dependent knowledge.

We believe, however, that no automated drawing understanding system is likely to be perfect, due to the presence of noise, inconsistent use of drawing conventions, and poor quality of the drawing. Therefore, we have integrated an interactive error-correction unit in our system that allows the user to correct residual errors.

The system has been extensively tested to verify the correctness of the algorithms and to validate the premise that it is possible to derive high-level description of a set of different classes of drawings, with similar properties, by encapsulating domain knowledge in a replaceable module. We experimented with four domains: logic circuits, electrical circuits, chemical-plant flow diagrams, and program flow charts. We created a test database of 64 drawings from the four domains and scanned each at two resolutions. In its automated mode, the system correctly recognized 74.2 percent and 88.5 percent of the symbols in electrical circuit diagrams and chemical plant flow diagrams, respectively. Higher recognition rates of 94.1 percent and 100 percent were achieved with logic circuit diagrams and flowcharts, respectively.

The rest of the paper is organized as follows. The segmentation system, which plays a crucial role in the success of the system, is described in Section 2. The generic rules are also discussed there. The hierarchical matcher used to identify symbols and the approach used to derive the high-level description of the drawing are presented in Section 3. In Section 4, the mechanism and features of the interactive error correction module are explained. The testing procedure, the results, and a discussion of the results are given in Section 5. A detailed comparison of our research with related work is presented in Section 6. Finally, Section 7 provides a summary of the work and some directions for future research.

2 Symbol Segmentation

Our first step in understanding flow drawings is to segment symbols from connection lines (CLs). Most drawing analysis systems do not segment symbols from CLs before the understanding phase. Recent systems have used the approach of expanding from identified symbol loops to predetermined surrounding region to search for symbols [5], [7], [9], [16]. Loop-free symbols have been identified with the help of surrounding text [7] or by feature points and line tracing [16]. In another approach, thinning is followed by line tracking to search for symbols [4].

In our approach, a few simple generic rules are used to define a complete partition of the drawing so that each line segment is either part of a potential symbol (PS) or a potential connection line (PCL). The results indicate that this distinction can be made with high degree of accuracy. Errors that remain after segmentation can still be detected when domain dependent knowledge is brought to bear on the symbol recognition task. The issue here is essentially of the degree to which interaction is allowed between the segmentation and the labeling processes. Because of the generic segmentation in our system, a weak interaction is sufficient, thus permitting a more modular design.

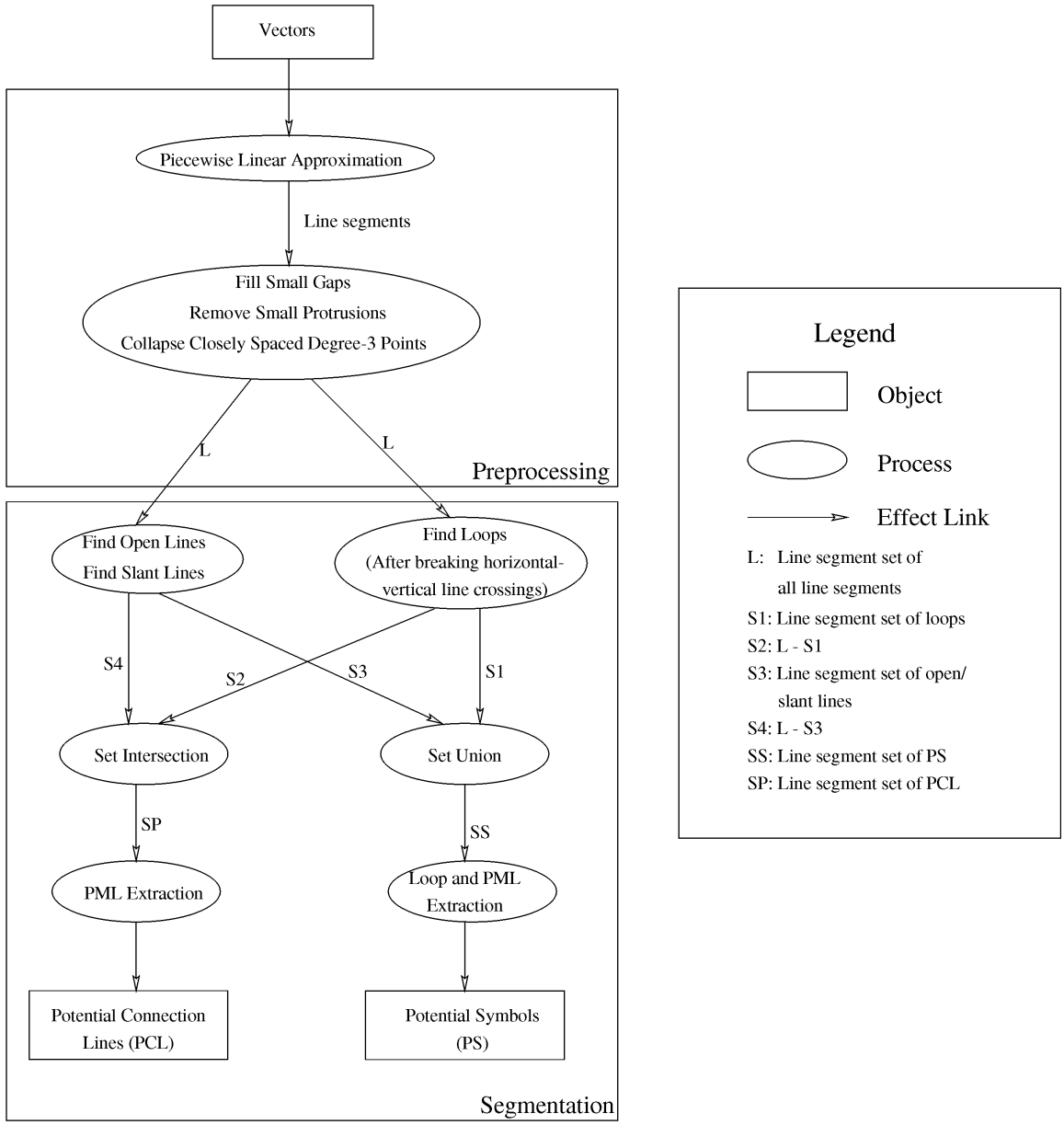


Fig. 2. Preprocessing and segmentation algorithm.

2.1 Generic Rules

Symbols in flow drawings possess characteristics different from those of the CLs, and these characteristics can be described by a few simple rules. It should be noted that these rules are fuzzy in nature. Some rules apply more frequently than others. However, all the rules hold for the vast majority of the drawings of this class.

1) Connection lines consist of horizontal and vertical lines.
2) Loops with simple geometric shapes, e.g., circles, rectangles, are part of symbols.
3) Slant lines and open lines (straight lines with both ends as degree-one points) are parts of symbols.
4) Symbolic loops do not contain crossing horizontal and vertical lines and, hence, are distinguishable from non-symbolic loops formed by crossing connection lines.
5) When a loop is part of a symbol, every thing inside the loop is assumed to belong to the symbol.
6) A connection line always terminates at a predefined symbol or a point symbol (input/output or a branch point).
7) Connection lines are longer than lines in symbols.

The first four rules form the basis of our segmentation algorithm. The rest are used in the symbol classification phase.

2.2 Segmentation Algorithm

We assume that the characters have been removed from the drawing by an existing technique (see, for example, [8]) or through manual editing. The main steps of our approach is shown in Fig. 2. In the preprocessing steps, line segments are produced from straight line vectors through piecewise linear approximation and further cleaned up by removal of artificial gaps, spurs, and closely spaced degree-three points.

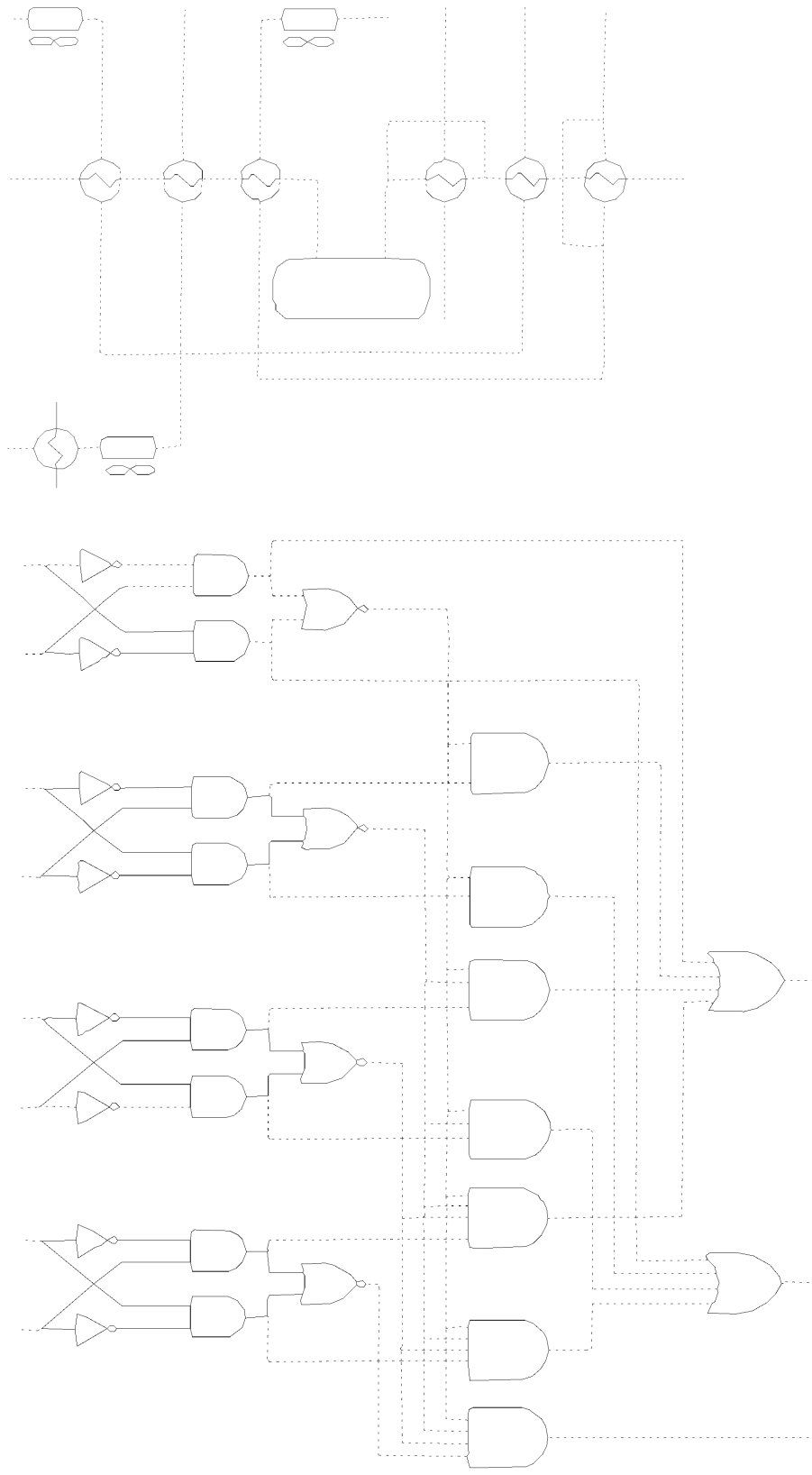


Fig. 3. Examples of the segmented drawings without error correction.

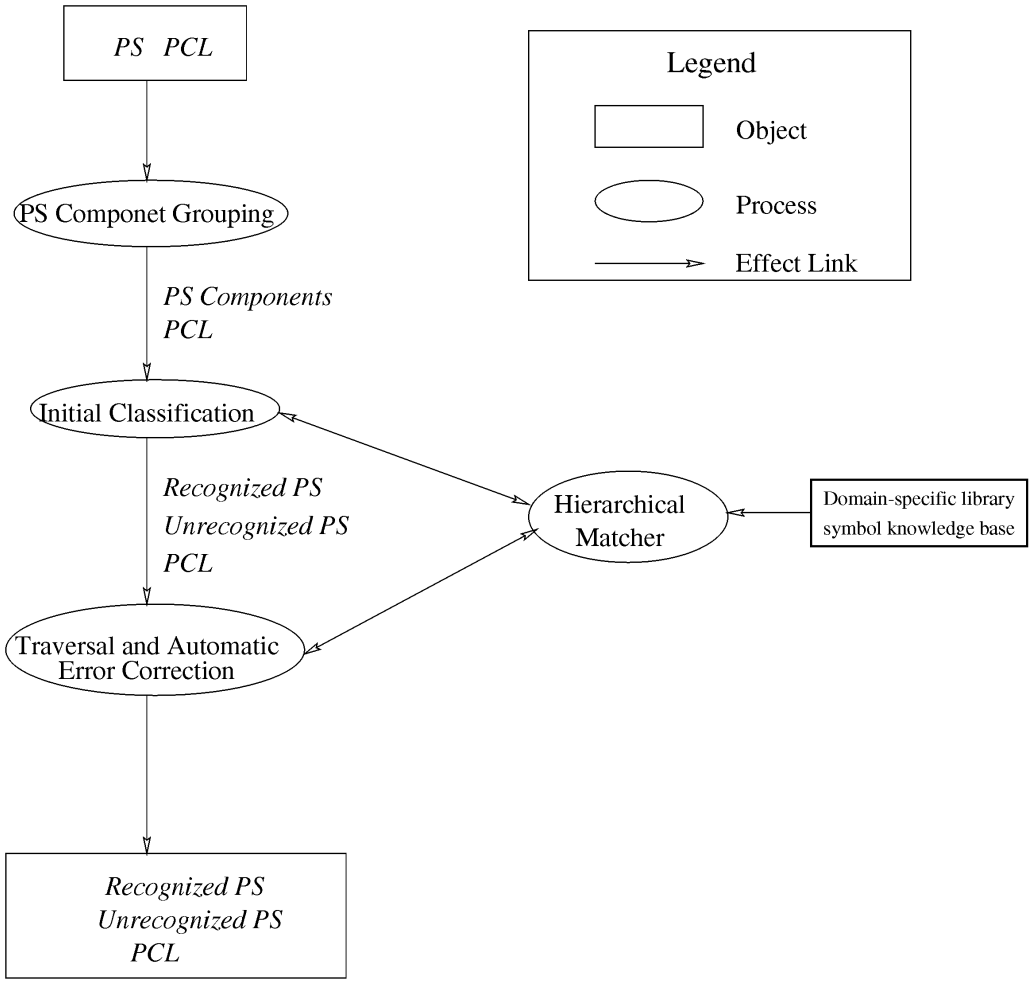


Fig. 4. Overview of the Understanding Subsystem.

Potential symbol line segments are extracted from the preprocessed line segment set using the generic rules described earlier. The loops formed by horizontal/vertical crossing CLs are opened up before a graph traversal algorithm is applied. For identifying loops, a drawing is treated as a digraph in which each line segment defines two edges of opposite directions, and the end points define two nodes. The line segments attached to the end points are ordered. The graph is traversed depth-first with the counterclockwise neighboring edge selected as the next edge. Only simple loops traversed from inside are selected as PS loops. Slant lines and open lines are collected with the PS loops to form the PS set. Line segments not classified as PS are grouped into polylines, which form the PCL set. Further details of the segmentation algorithm may be found in references [22], [23].

2.3 Results

Fig. 3 shows two test drawings after symbol segmentation. The PSs are shown as solid lines and the PCLs are shown as dashed lines. Notice that the segmentation is not perfect. Four of the six circular heat exchanger symbols in the chemical flow drawing are not completely segmented; parts of these symbols are classified as CLs, which are indicated as dotted lines. On the other hand, some CLs in the logic diagram are misclassified as PSs. Some of the misclassified CLs, along with several symbols, form a connected component and create a PS that is comprised of more than one symbol.

Misclassification of PSs or PCLs results in segmentation errors. The PSs and PCLs can be either completely misclassified or partially misclassified. However, surprisingly, with just a few domain-independent rules, the accuracy of segmentation is very high. In an experiment with a set of 24 drawings, drawn from the four application domains, 96 percent of the symbols were completely segmented and the rest were partially segmented. Only in 25 percent of the drawings were there instances of connection lines left completely as a symbol part [23]. These errors can be corrected during symbol classification, when more information about the symbols is known.

All four generic rules described earlier in this section have proven to be effective. However, there are instances when each rule breaks down. For example, in rare cases, parts of connection lines are slanted. Occasionally, a connection line connects to a symbol in such a way that the connecting point forms a junction that is confused as a crossing of two connection lines. Overall, the rules worked very well and the high accuracy of segmentation proves their effectiveness.

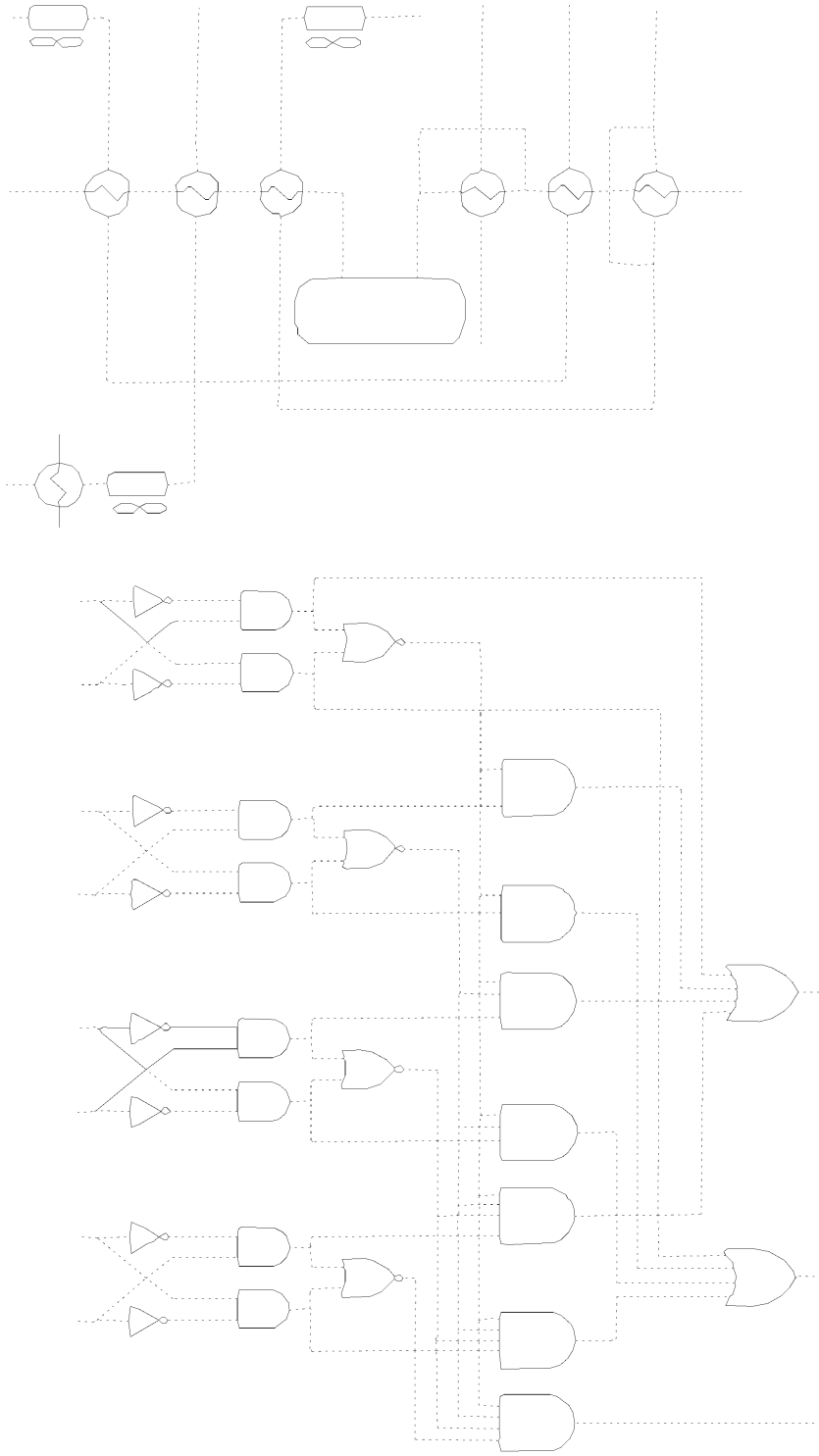


Fig. 5. Examples of the drawings after traversal and error correction.

It should be noted that the segmentation routine is not called after this point. The higher level routines may, however, reorganize the results produced by the segmentation procedure (see Section 3.4). The understanding subsystem and the hierarchical symbol matcher are unaffected by any change to this module.

3 Symbol Classification

A flow drawing can be viewed in two different ways. In a top-down view, it can be seen as a large complex symbol consisting of a group of symbols joined by connection lines. The degree-one points in connection lines are input or output in flow drawings and, as such, can be regarded implicitly as symbols. Similarly, degree-three points in connection

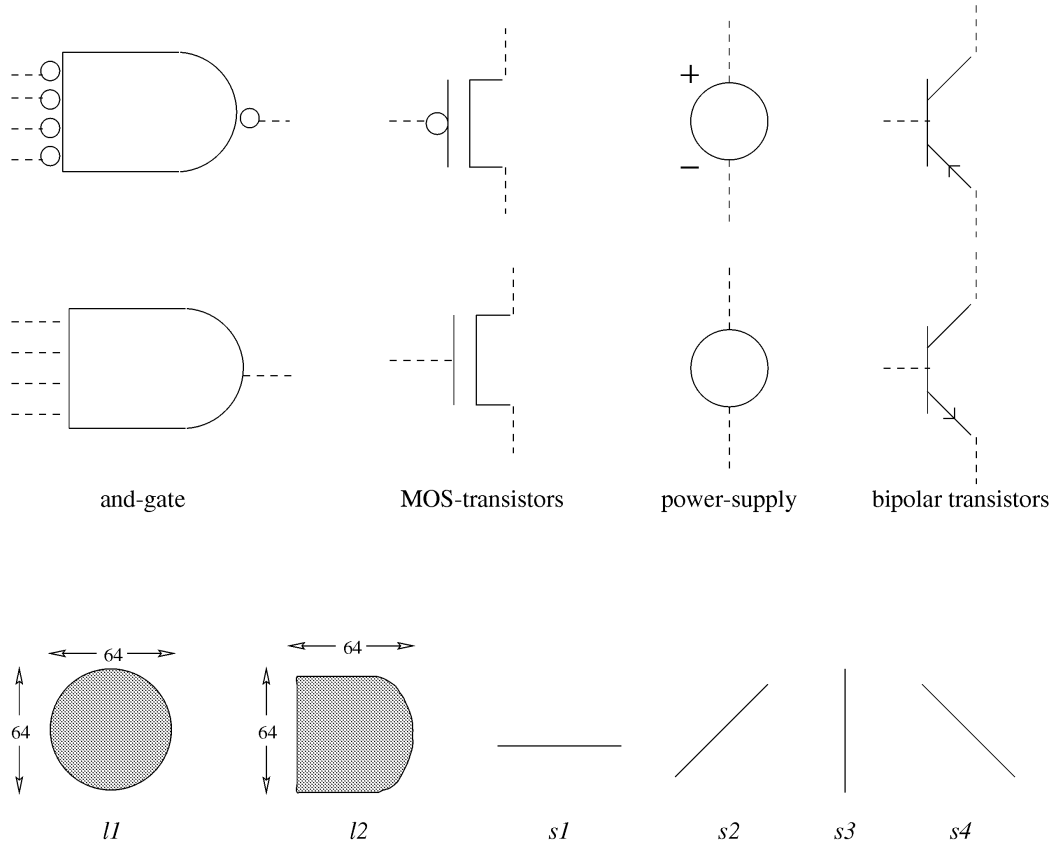


Fig. 6. A small sample symbol set and the library of loops and strokes.

lines can be considered to be branch-point symbols. Each complex symbol, in turn, can be decomposed down to loop and polyline entity level. In a bottom-up view, the drawing is a thinned image formed by a collection of line segments. Each line segment belongs to either a symbol or a connection line. The goal of the understanding system is to identify these line segments, group the ones that belong to symbols, recognize the symbols, and, finally, determine the connectivity of the symbols using segments that are identified as connection lines. This bottom-up view is particularly relevant to the approach followed here.

We assume that the symbols have a hierarchical organization as follows. A symbol is comprised of one or more symbol components, each of which, in turn, consists of a connected set of entities. An entity can be either a simple loop or a stroke. A loop is assumed to have finite number of domain-specific distinct shapes. A stroke is approximated as straight lines and grouped into four domain-independent categories, based on their orientation. The domain knowledge, necessary for symbol identification, is captured in the form of two libraries: a library of loops that the symbols may have and a hierarchically organized library of symbols. Further details about the libraries and the symbol matching process appear later.

3.1 Overview

Fig. 4 gives an overview of the symbol classification process. The input to the process comes from our segmentation algorithm, which produces a set of entities labeled as PS (potential symbol) or PCL (potential connection lines). The first step involves grouping the set of PS entities into connected components and classifying the entities in each component. The loop entities are classified by matching against the loop library using a scaled template matching approach. The polyline sections of PS entities are classified into four types of strokes: horizontal, right diagonal, vertical, and left diagonal.

The next step involves initial classification of symbols, using the domain-specific symbol library. Since a symbol may have multiple (disjoint) components, the symbol library is organized as a two-level hierarchy. At the top level, each symbol is defined as a collection of one or more symbol components. At the next level, each symbol component is characterized by data which capture its entity counts and some other properties (an example of symbol library appears in the next section). The symbol component matcher classifies a PS component by comparing it with the symbol components in the library. As a result, poorly matched PS components are placed in the unrecognized category. The remaining PS components are further submitted to the symbol matcher for classification. If the PS component can be part of a multicomponent symbol, such as a transistor in a CMOS circuit, other PS components in the neighborhood may also be submitted to the symbol matcher for recognition.

The final step in symbol classification involves a traversal of the drawing from the well recognized symbols. The PCLs between these symbols are confirmed as CLs. The recognition process continues by choosing, in turn, an unrecognized PS component that, according to the measures described later, has the highest potential to be recognized as a PS component.

TABLE 1
A SMALL SYMBOL LIBRARY EXAMPLE

Library = [and-gate, MOS-transistors, power-supply, bipolar-transistors]

and-gate = [C1]
MOS-transistors = [C2, C3]
power-supply = [C4, C5, C6]
bipolar-transistors = [C7]

C1	*l1*	*l2*	*s1*	*s2*	*s3*	*s4*
SC_{max}	9	1	–	–	–	–
SC_{min}	1	1	–	–	–	–

$mf = l1\ S_{mf} = 1\ C_{min} = 3\ C_{max} = 9$

C2	*l1*	*l2*	*s1*	*s2*	*s3*	*s4*
SC_{max}	1	–	–	–	1	–
SC_{min}	0	–	–	–	1	–

$mf = s3\ S_{mf} = 1\ C_{min} = 1\ C_{max} = 1$

C3	*l1*	*l2*	*s1*	*s2*	*s3*	*s4*
SC_{max}	–	–	2	–	1	–
SC_{min}	–	–	2	–	1	–

$mf = s3\ S_{mf} = 1\ C_{min} = 2\ C_{max} = 2$

C4	*l1*	*l2*	*s1*	*s2*	*s3*	*s4*
SC_{max}	1	–	–	–	–	–
SC_{min}	1	–	–	–	–	–

$mf = l1\ S_{mf} = 1\ C_{min} = 2\ C_{max} = 2$

C5	*l1*	*l2*	*s1*	*s2*	*s3*	*s4*
SC_{max}	–	–	1	–	1	–
SC_{min}	–	–	0	–	0	–

$mf = s3\ S_{mf} = -1\ C_{min} = 0\ C_{max} = 0$

C6	*l1*	*l2*	*s1*	*s2*	*s3*	*s4*
SC_{max}	–	–	1	–	–	–
SC_{min}	–	–	0	–	–	–

$mf = s1\ S_{mf} = 1\ C_{min} = 0\ C_{max} = 0$

C7	*l1*	*l2*	*s1*	*s2*	*s3*	*s4*
SC_{max}	–	–	1	1	2	1
SC_{min}	–	–	1	1	2	1

$mf = s2\ S_{mf} = 1\ C_{min} = 3\ C_{max} = 3$

Note : A – in the entry is the same as a 0, and is used only for clarity.

Some errors in segmentation and symbol classification can be corrected during the traversal. We identify two types of errors for the purpose of error correction. First, a PS component may be under-segmented, i.e., two or more symbol components are grouped together by some CLs misclassified as PSs. Second, one symbol component may be over-segmented and split into several PS components. If under-segmentation is suspected, the unrecognized PS component is split to see if it improves the recognition. Conversely, for over-segmentation, the system attempts to merge the PS component with one or more neighboring PS components for improved recognition. In either case, resulting new PS components are classified by the symbol component and symbol matchers.

Fig. 5 shows the results after the traversal for the two drawings whose segmentation is shown in Fig. 3. The chemical plant flow diagram in Fig. 3 only has four errors of over-segmentation in the exchanger symbols—the circular loop in the symbol is mistakenly broken in each case by the segmentation algorithm because of the crossing orthogonal lines. All of the four errors are corrected by the traversal. There are several instances of under-segmentation in the logic diagram in Fig. 3, of which all but two are corrected during traversal. The two exceptional cases are not fully split because our matcher calculated no improvement for the split.

3.2 Symbol Library Example

As mentioned above, the domain-specific information is captured in two libraries: a loop library and a symbol library. We will use the small symbol set in Fig. 6 to illustrate the organization of the two libraries. The set consists of four symbols and we show two instances of each symbol.

Many properties can be used to characterize symbols and symbol entities: shapes, aspect ratios, sizes, relative sizes, relative positions. For loops, we use their normalized shapes. Each loop is normalized to 64 × 64 size and stored in a bitmap format. Loops of symbols which may appear in different orientations in drawings are represented by multiple bitmaps, one for each possible orientation. The aspect ratio of the loop shapes is not used in our representation and analysis. The loops in the library are generated semi-automatically by visually inspecting them in drawings and selecting them with a mouse. The loop library for our example symbol set is shown in Fig. 6. Only two loops, *l*1 and *l*2, are needed for our set of symbols. In addition, the four strokes, *s*1, *s*2, *s*3, and *s*4, are also shown for completeness.

The symbol library consists of a high-level description of the symbols. Since a symbol may contain more than one component, a symbol is represented as a set of descriptions of its components. The and-gate and the bipolar transistor symbols have only one component each. The MOS transistor has two components and the power supply has three, the circle, the +, and the –. Thus, we need a total of seven components to describe our library. Each component is described by two feature vectors and four other parameter specifications. The features used are *strokes* and *loops*. As mentioned earlier, the strokes are approximated by four types of straight lines: horizontal, right diagonal, vertical, and left diagonal. Each type of loop in the library is given a label. A feature vector stores the number of strokes or loops of each type. Often, the count of features of a specific type, however, is not fixed for a given symbol, e.g., an and-gate may have between

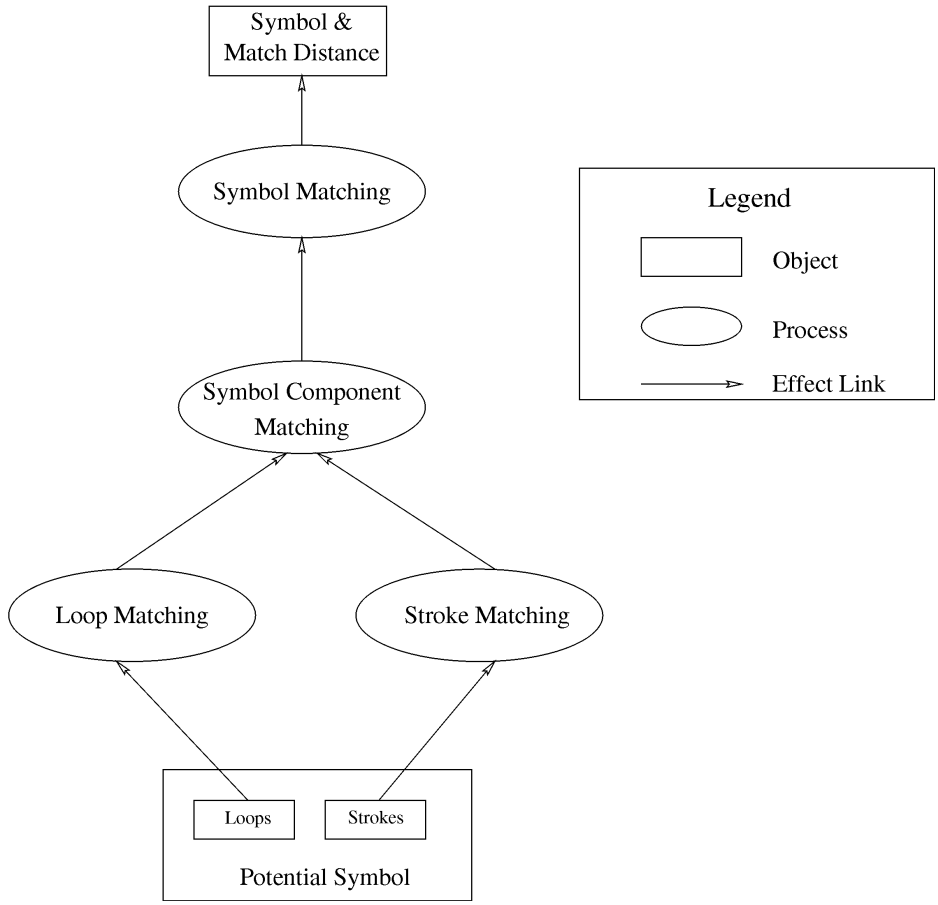


Fig. 7. Structure of the hierarchical matcher.

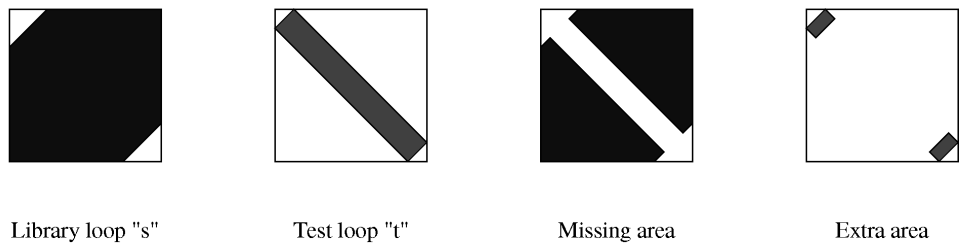


Fig. 8. Illustration of concepts in the loop matching equation.

one to nine "inversion bubbles" (small circular loops) in a logic diagram. Therefore, we provide for a range of value for each feature count. Table 1 shows the entries for symbols in Fig. 6.

The first (SC_{min}) and second (SC_{max}) feature vectors store the minimum and maximum feature counts for the symbols, respectively.[2] Four other properties of a symbol component are stored as values following the two feature vectors. The main feature ID (mf), is the index of the stroke or loop of the main feature in the symbol component, i.e., the largest or most unique feature that must exist in it. We also store the size of this main feature (s_{mf}), whether it is large, small, or *don't care* (coded as 1, 0, –1, respectively). The minimum and maximum numbers of CLs adjacent to the symbol component are stored in c_{min} and c_{max}, respectively.

The symbol library, as exemplified by Table 1, is created manually. It is built by counting the feature numbers and adjacent CLs. The main feature ID, the main feature size, and the size ratios are estimated by visual inspection of the symbols. The process can be automated if a drawing containing all the standard symbols is available. The feature counts or CL counts can certainly be obtained automatically, so can the size and size ratios. However, the current implementation does not have an automatic symbol builder.

2. A third vector (SC_{ratio}) that contains the minimum relative size ratio between the main feature and other features in a component is optional.

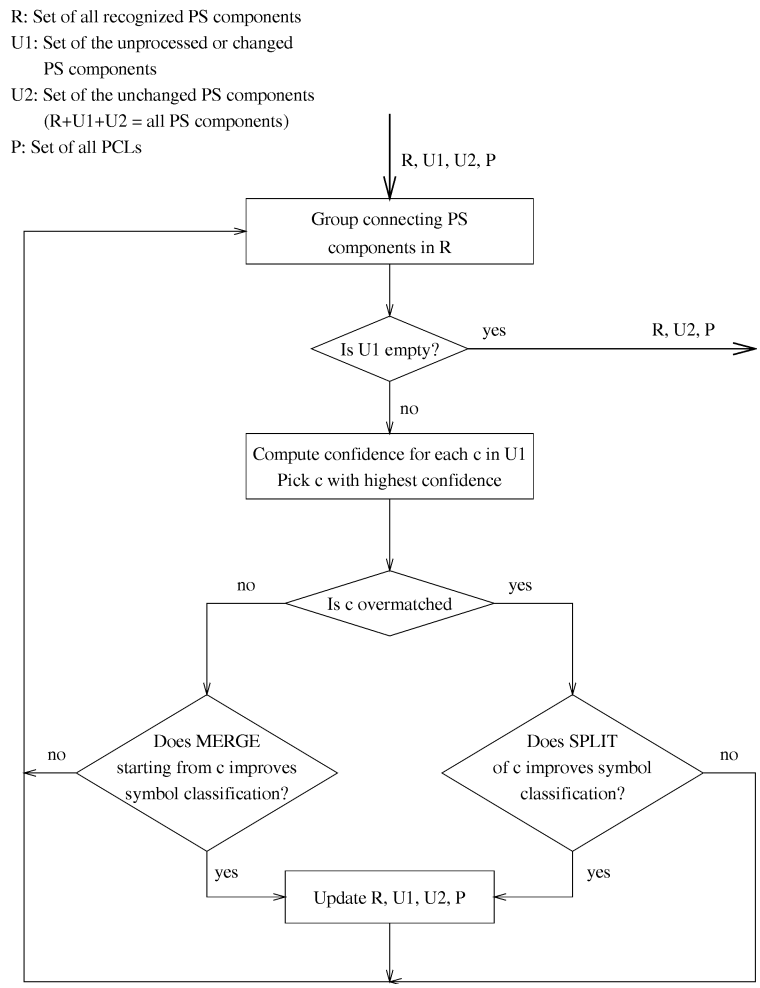


Fig. 9. Algorithms of drawing traversal and automatic error correction.

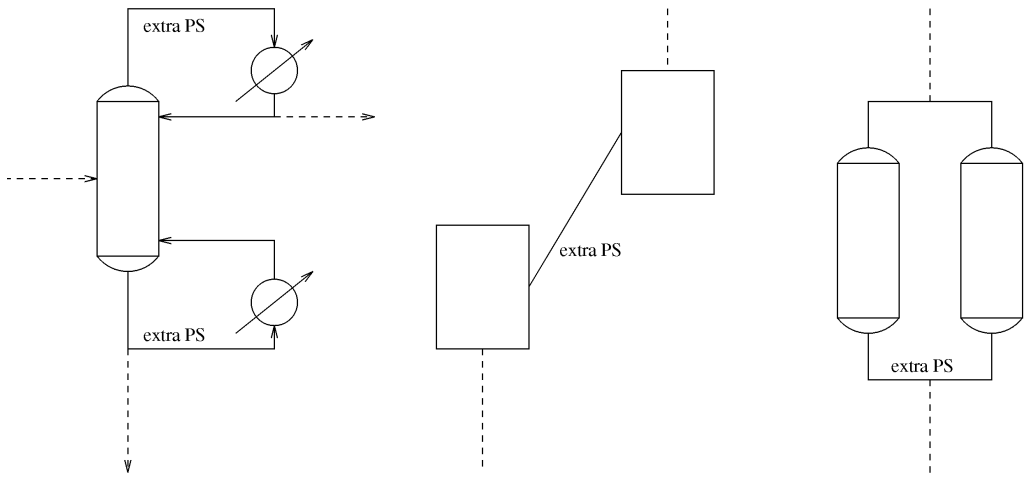


Fig. 10. A few examples of under-segmented PS component.

3.3 Hierarchical Matching

Since the symbols are represented hierarchically, it is natural to use a hierarchical matcher to recognize the symbols. At the lowest level, individual loops and strokes are matched. This provides the description for each PS component which in turn is matched with the library symbol components. Finally, the PS components are grouped and matched with the symbols. Each matcher produces a set of best matches along with associated confidence measures. The three levels of matching are illustrated in Fig. 7.

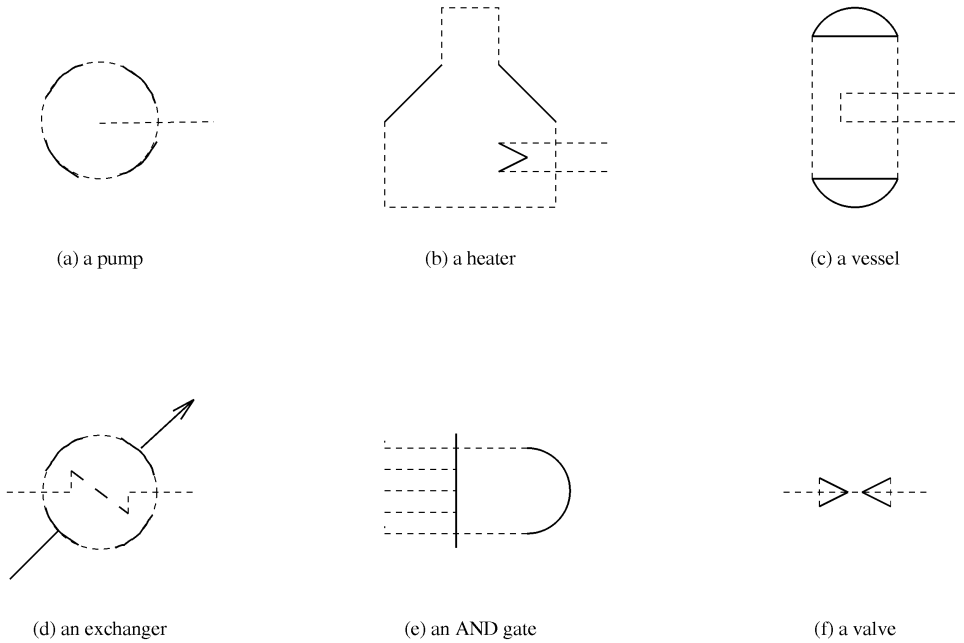


Fig. 11. Examples of fragmented symbols with missing loops.

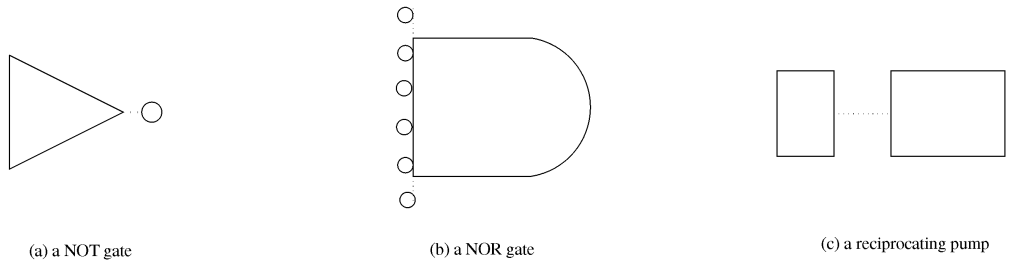


Fig. 12. Examples of fragmented symbols with missing PMLs; the missing parts are shown in dotted line.

3.3.1 Loop and Stroke Matching

The loop classification is based on scaled template matching. Each loop is scaled to 64×64 bitmap, where the interior pixels of the loop are 1s and the rest are 0s. Given a test loop (t), and a library loop (s), the normalized goodness measure (GM) is defined by:

$$GM_{loop} = 0.5 \times \left(1 - \frac{A_m}{A_s}\right) + 0.5 \times \left(1 - \frac{A_e}{A_t}\right),$$

where, A_s is the area of the library loop, A_t is the area of the test loop, A_e is the area of the region given by the points in the test loop which do not overlap the points in the library loop ($t - s$), and A_m is the area of the region given by the points in the library loop which do not overlap the points in the test loop ($s - t$). This formula provides a normalized difference between the set of interior points in the test and the library loops. Fig. 8 shows a test loop and a library loop; the interior points of the loops are shaded. The missing region ($s - t$) and the extra region ($t - s$) are also shown in the figure. The term A_m / A_s reflects the part of the symbol that is missing in the test loop and the term A_e / A_t measures the part of the test loop that is extra, i.e., not in the symbol. Thus, the formula uses the appropriate ratios to measure the goodness measure. If $s = t$, $A_e = A_m = 0$, the goodness measure is one. If $s \cap t = \phi$, $A_e = A_t$, $A_m = A_s$, the goodness measure is zero.

Strokes are classified into the four categories—horizontal, vertical, right-diagonal, and left-diagonal—based on the angle formed by the starting and ending points of the stroke. For each stroke type, the goodness measure is defined as the Euclidean distance between the two end points of the stroke divided by the length sum of all line segments in the stroke, i.e.,

$$GM_{stroke} = \frac{L_{1,n}}{\sum_{i=1}^{n-1} L_{i,i+1}},$$

where $L_{i,j}$ is the distance between the two points, i and j. A perfect straight stroke will have a goodness measure of 1.0.

3.3.2 Symbol Component and Symbol Matching

We extend the goodness measures for the loop and stroke entities to a PS component by taking the weighted average of goodness measures of its constituent entities, where the total line segment length of an entity is used as its weight. If a PS component passes the test according to this measure, then we use a secondary distance measure for symbol components based on a comparison of features in the library. The distance measure of a test PS component is the mini-

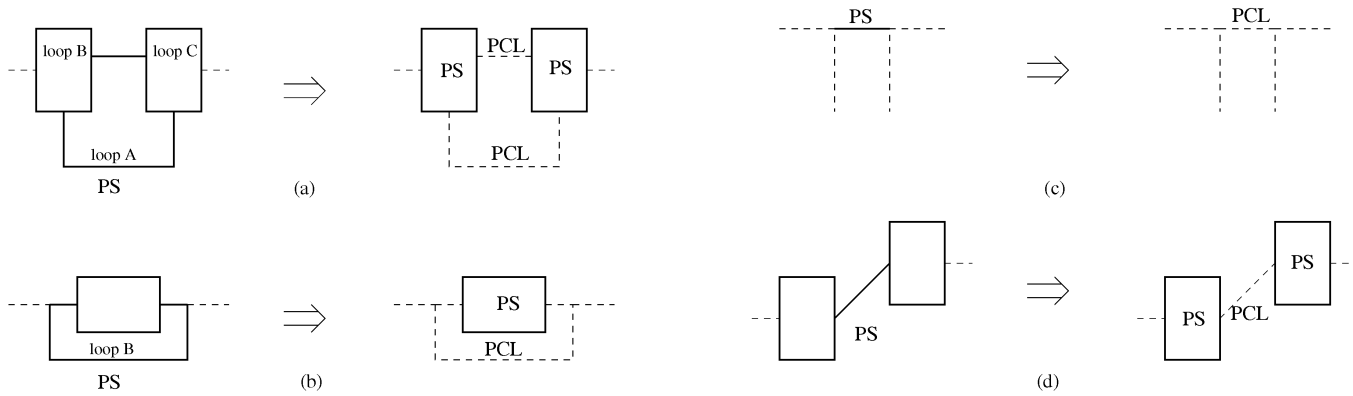


Fig. 13. A few examples of switching PS entities to PCL.

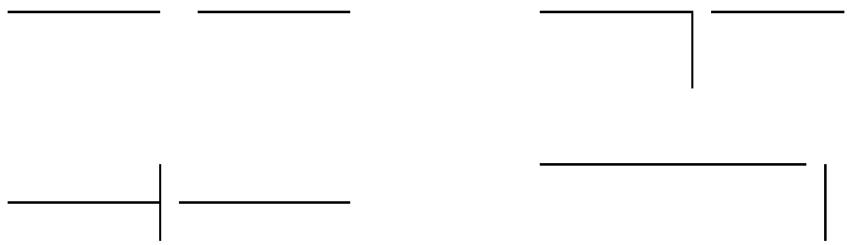


Fig. 14. A few examples of gaps along PCLs that can be interactively closed.

mum sum of the number of extra or missing elements between the test component and the library components. A symbol is considered to be recognized only if all its components are recognized. In this case, the sum of the distance measures of the components is used as the measure of match for the symbol.

Once the distance measure d of a PS component is known, we classify the result of matching into four categories, as follows:

1) A perfect match if $d = 0$.
2) An *over match* if $d \neq 0$ and the number of extra entities is more than the number of missing entities. If the PS component contains more entities than the library symbol component, it may contain CLs or more than one symbol component. This often happens when the symbol component is under-segmented.
3) An *under match* if $d \neq 0$ and the number of missing entities is more than the number of extra entities. The PS component may be part of a symbol component that is fragmented.
4) A *mixed match* if $d \neq 0$ and the number of missing entities is equal to the number of extra entities. Entities may have been misclassified. The PS component may match the symbol component if the misclassification is corrected.

Several best matches for a PS component are ordered according to the distance criterion and subjected to additional tests to decide whether a PS component should be split or several PS components should be merged for automatic error correction. These tests include checking number of connections, comparing relative sizes, verifying the existence of the main entity, and examining if any small loops have been misclassified.

Knowledge of a priori probability of symbol frequency may be useful in the symbol matching. It will be particularly useful when the symbols are small, e.g., valves in chemical plant diagrams. This information can be obtained in our system in the same manner as the symbols are captured. An expert can view the drawing and point to all instances of a symbol in the drawing, instead of just one. This way, the symbols and their frequency can be obtained in one pass through the drawings. This idea can also be extended to matching of symbol loops.

3.4 Drawing Traversal and Error Correction

An overview of drawing traversal was provided earlier in Section 3.1. Here, we provide further details of the process with reference to Fig. 9.

We classify the PS components as recognized or unrecognized, depending on the match distance and the goodness measure of their match with the library components. Theoretically, only those PS components that have a high value of the goodness measure and are perfectly *matched* ($d = 0$) can be considered as "recognized." However, this criterion was found to be too strict in reality as it prevented recognition of well-matched PS components due to artifacts present in the original or the scanned image. Therefore, we use a relaxed criteria to include those *matched* PS components, with $d \leq 1$, and $GM \geq 0.9$ as "recognized" and the rest as "unrecognized." The threshold has been verified to work well in a large number of experiments.

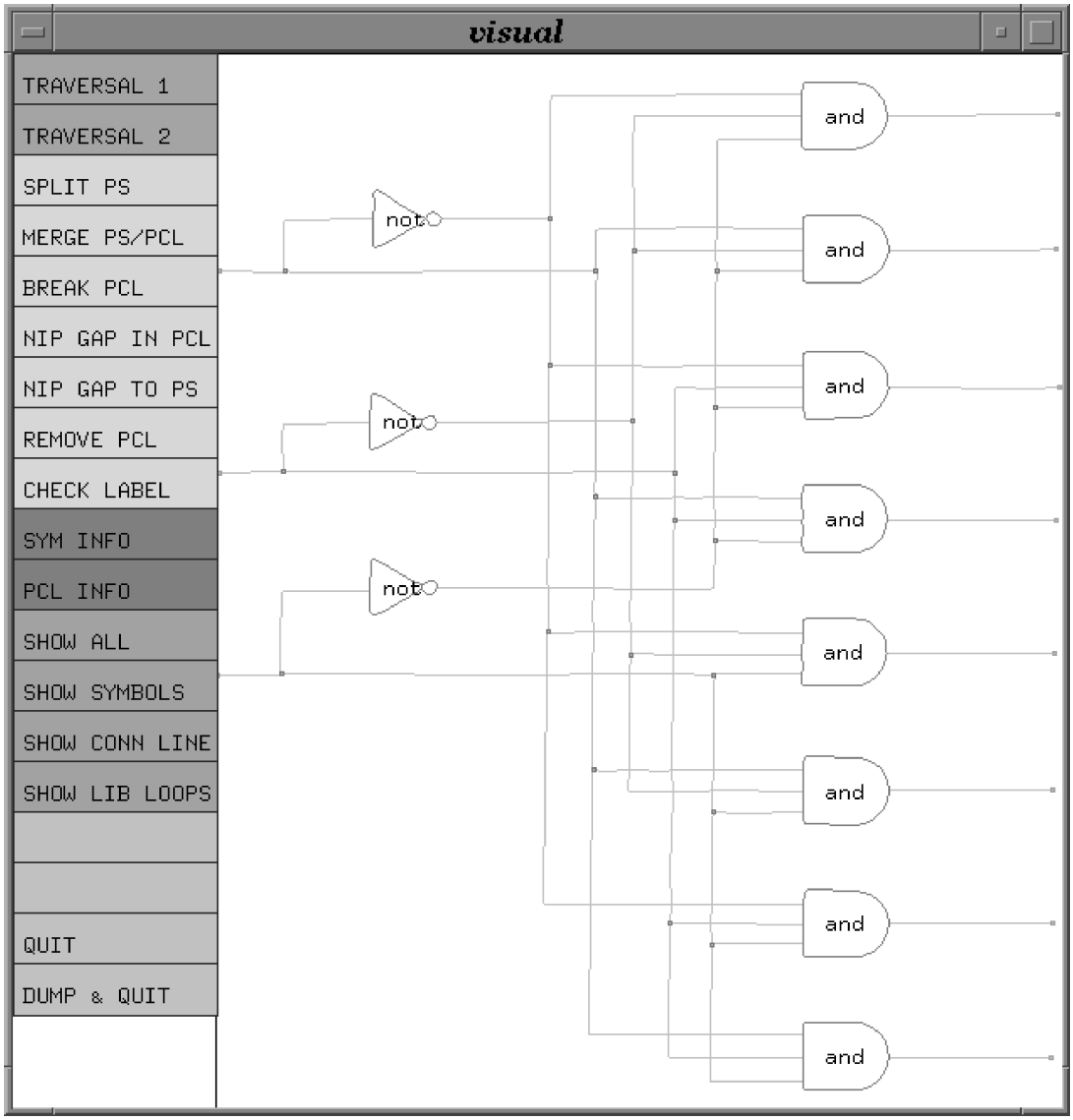


Fig. 15. Graphical user interface for interactive error correction.

The initial set of recognized PS components is ordered in the decreasing order of their goodness measure. These components are selected, in turn, as the starting point for drawing traversal. When a new recognized PS components is reached during the traversal, it is added to the ordered list. When an unrecognized PS component is reached, a decision is made whether it should be split into two or more components or merged with some neighboring components. This decision is based on the component's match value. If the value is "over-match," it is split and, if it is "under-match," it is merged with other unrecognized PS components in its neighborhood; interconnecting PCLs are also included in the merged component. The results of split or merge are accepted only if the resultant segmentation improves the understanding of the drawing. Otherwise, the changes are undone. In certain cases, it is possible that the unrecognized component will be recognized if a different route of traversal is taken. Further details of the split and merge procedures appear in the following sections.

3.4.1 Splitting Procedure

Under-segmented PS components are those that contain more than one symbol component. They are formed when CLs are misclassified as PSs and several symbol components are joined together. The misclassified PS entities can be loops or PMLs and are more likely to be larger than other entities in the same PS component because they connect symbols. Fig. 10 shows a few examples of under-segmented PS components.

To break an under-segmented PS component, we can remove a large entity from the entity set of the PS component, and test if the new PS components match better with the library symbol components. Among the large loop entities, if one is badly matched with loops in the symbol library, it is likely to be an extra entity because large loops should match well with library loops. In addition, a loop that contains a side collinear with an adjacent PCL is more likely to be an extra loop.

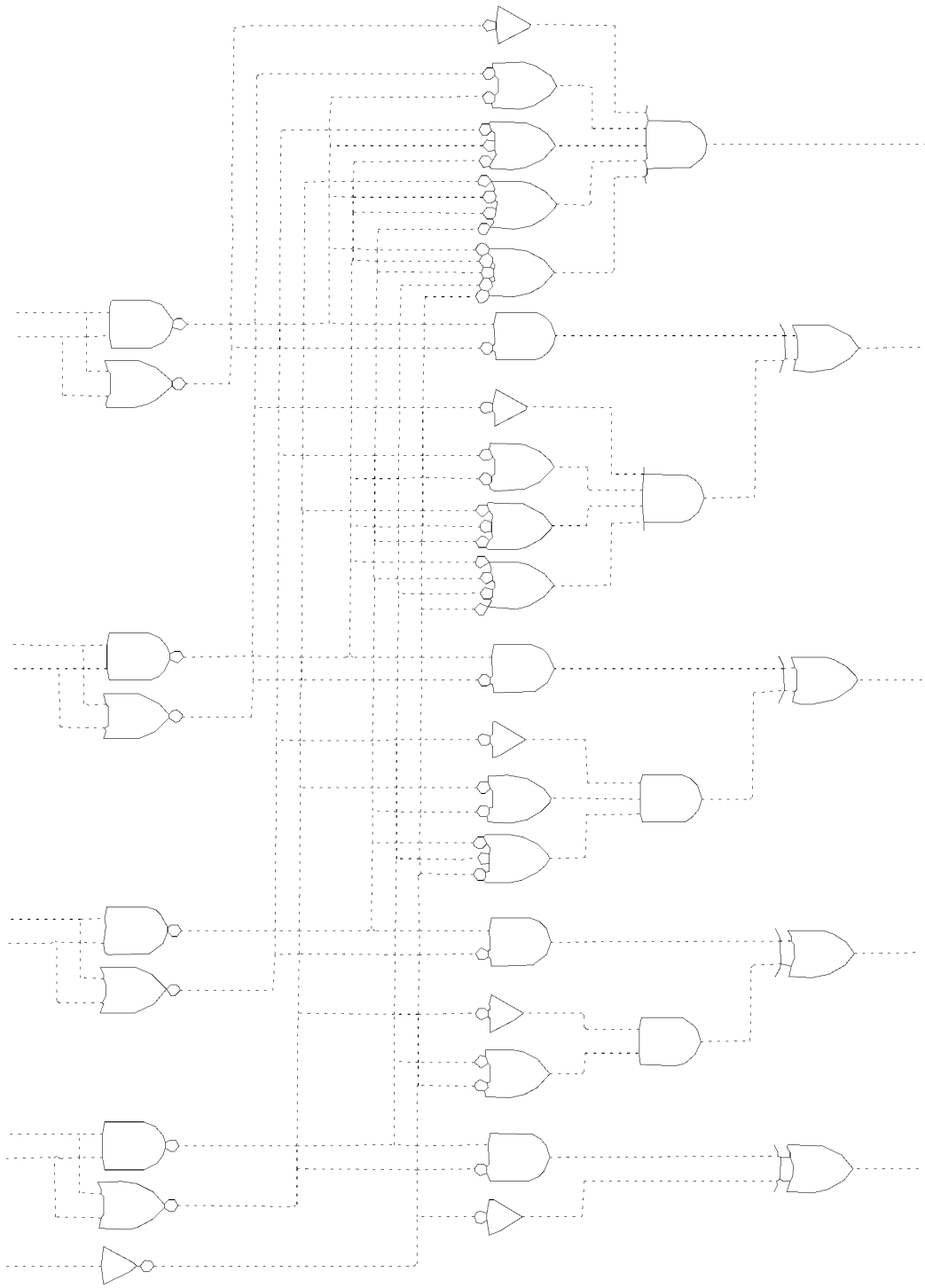


Fig. 16. Results for a logic circuit diagram.

To avoid checking too many entities, the algorithm considers only the three largest loops and the largest PML in the PS component. The results are compared and the one that gives the best overall match is picked. If the best pick gives improvement over the old PS component and generates more than one new PS component, then the new PS components and PCLs are updated.

3.4.2 Merging Procedure

Fragmented PS components are caused by the misclassification of one or more loops or PMLs that belong to a symbol. Under such circumstances, a single PS component may break up into multiple PS components. In some cases, the size of the PS component may be reduced. Both the cases result in *under-matched* PS components. Figs. 11 and 12

show examples of fragmented PS components. In some cases, parts of the symbol may be classified as PS. A symbol PML may also be completely misclassified if it is completely horizontal-vertical. It may be partially misclassified if it contains both horizontal-vertical and slant sections. Fig. 12 shows a few such examples. Our system can automatically correct most errors caused by partially misclassified symbol loops or PMLs.

The fragmented PS components are recovered by merging an unrecognized PS component with neighboring unrecognized PS components and the PCLs connecting them. Under some special conditions, as explained below, a neighboring recognized component may also be merged. If the merging improves matching, the PS components and PCLs are combined into one PS component. Otherwise, they are reverted back to their original classification. The merging algorithm is a two-step hypothesis and test procedure:

1) selecting potential candidates for merging and
2) testing to see if the result is a matched symbol.

The criteria for selecting and merging fragmented PS components are based on existing PS components and are given below:

- If the two ends of a PCL connect to the same PS component, the PCL is included in the set of entities for merging. The vertical lines in the arrows in Fig. 11f satisfy this condition.
- Another PS component is connected through a short PCL, then the PCL and the PS component are included, even if the PS component is recognized (Figs. 12a and 12b).
- If another unrecognized, but not *over-matched*, PS component is connected through a PCL with at least one degree-two point, and, if the PCL size is comparable to the PS component sizes as described earlier, then the PCL and the other PS component are included (Figs. 11a, 11d, and 11e).
- If another unrecognized, but not *over-matched*, PS component is connected symmetrically through a PCL, then the PCL and the PS component are selected (Figs. 11b, 11c, 11e, 11f, and 12c).

The set of PS components and PCLs should contain at least one short PCL or one degree-four horizontal-vertical crossing point after all of them are collected. To match the collected PS components and PCLs, we first extract loop and PML entities from the line segments contained in the set. Loops formed by CLs are not likely to exist in the set, and we do not open up those that contain degree-four horizontal-vertical crossing points. A new set of loop and PML entities are then matched by SC-matcher with the library symbol components. The new PS component is accepted if it is recognized or if the matching is improved.

3.4.3 Correcting Interconnection Errors

Artificial gaps, either from the original drawing or introduced during copying or scanning, exist on many CLs. They cause interconnection errors and they are closed after the drawing traversal for accurate interpretation of drawings. Two types of interconnection errors are corrected in our system: those in CLs that separate the line into two parts and those at ends of CLs that disconnect the lines from symbols. To correct such errors automatically, we check the gaps that are smaller than a certain threshold and close them. The threshold was determined experimentally by analyzing the short gaps in a test set of 24 drawings after segmentation [22].

Gaps in Connection Lines: In some drawings, when two CLs cross each other, one of the crossing CLs is broken into two and a gap is generated. In addition to these cases, a gap may be introduced during copying or scanning. When finding the connection relationship between symbols, the gaps on CLs need to be closed. Frequently, these gaps are small, and lines on the two sides of the gap are collinear. If we know the size distribution of such artificial gaps, we can choose gaps in the size range and close the gaps automatically. To find the size distribution of artificial gaps, we studied the gaps between degree-one points that are shorter than 0.25 inch. We found that all gaps on PCLs smaller than 1/16 inch can be closed without a collinearity test because no real gaps are in this range, and gaps between 1/16 and 1/8 inch can be closed with a collinearity test.

Gaps at Ends of Connection Lines: In some drawings, mostly in flowcharts, a CL may come close to a symbol without touching it. A gap is created between the end point of the CL and the symbol. If such a gap is not large, it can be automatically closed, so that the logical connection between symbols is correct. Again, we studied such gaps smaller than 1/4 inch, and found that a gap smaller than 1/16 inch can be closed automatically. While this distance will vary from one type of drawings to another, it can be determined by statistically analyzing the gaps in a manner similar to ours.

Most of the algorithms used in the preprocessing are of order n or $n \log n$, where n is the number of vectors derived from the drawing after vectorization. Since the drawing is first segmented into PSs and PCLs, the number of objects is reduced and the search is simplified. Therefore, we chose not to use sophisticated environments, such as the blackboard system. The drawings in the training and test set are processed within a minute on a Sun 670MP in the worst case. Since the computing power of computers is constantly increasing, the processing time of our system is not a major issue.

4 Residual Error Correction and Measurement

The user can correct residual errors interactively through a menu-driven graphical interface. As the corrections are being made, the system logs differences from the result produced by the symbol classification system described in the last section. The differences are analyzed automatically at the end of interactive correction to arrive at measures of the residual error. The interactive error correction module is designed independently so that it can bypass some or all of the symbol classification steps shown in Fig. 2. Thus, it augments the automatic recognition system to allow error-free conversion. At the same time, its error measurement

capabilities can help in evaluation of alternative strategies for symbol classification. We discuss below the functionality of the user interface and the basic ideas behind error measurement.

4.1 User Interface

Among the residual errors to be corrected through the user interface, the first type of residual errors are *segmentation* errors not corrected in the automatic classification and error correction. These errors can be corrected by the following functions in the user interface:

Switch PS: Either a part or all of a PS component is manually identified as CLs. The former case involves breaking of a loop, which may share boundary with other PS entities. The latter one applies to a polyline PS entity. Any new PS components are reclassified. For example, when loop A in Fig. 13a is switched to PCL, two new PS components and two new PCLs are generated. When loop B in Fig. 13b is switched to PCL, there is a new PS component and a new PCL. For a PS entity that is a CL, the whole entity is switched and one new PCL will be added as shown in Fig. 13c and Fig. 13d. In the case of Fig. 13d, two new PS components are also generated as the result of the switch.

Switch PCL: To switch PCLs, a set of PS components and PCLs is selected and merged together into a single PS component. This set may contain more than one element, or it may contain only one PCL. In the former case, the entities are merged together to form a new PS component. In the latter case, the single PCL is switched to a PS. To make the operation easier, when a PCL is selected, the PS components at the end points of the PCL are also selected. To switch PCLs, the user may select as many PCLs and PS components by clicking. The selected items are highlighted and matched by the system to update their labels.

A second type of error occurs when there are artificial gaps that break a CL or separate a CL from a symbol. Two functions are provided in the user interface to take care of these types of *interconnection* errors:

Bridge PCL gap: Our automatic gap closing mechanism checks only along collinear lines for gaps between two degree-one points and for very small gaps. Gaps along a PCL may not have been bridged because the gap is too large or because the gap is at a crossing point, branch point, or a turning corner. After the gap is closed, if two different PCLs meet at the gap, they are merged into one PCL entity. Fig. 14 shows a few examples of gaps that may be interactively closed.

Bridge gap to PS: A gap between a CL and symbol component that is not closed in the automatic gap closing is closed. In this case, the degree-one point in the PCL and a PS component are selected, and the gap is closed.

A third type of error occurs when a symbol is assigned a wrong label by the matchers. Such *labeling* errors are corrected by the following function:

Relabel a symbol: Some symbols are labeled incorrectly because the original drawing quality is poor or because our matcher makes an error in identification. To correct such errors, the label of each symbol is displayed next to the symbol. The user can make a correction by selecting the misclassified symbol and a label to replace the incorrect label. A window with all the symbol names is shown so the user can select the correct label for the misclassified symbol.

Additional short cuts are included in order to minimize user interaction. However, these are equivalent to a combination of the functions already described. Fig. 15 shows the layout of the graphical user interface for interactive operation and error correction. The functions described in this section are incorporated into more intuitive options in the interface.

4.2 Error Measurement

As described above, errors can be classified into three types: segmentation errors, interconnection errors, and labeling errors. Among these, interconnection errors are measured simply by counting the number of gaps closed interactively. Labeling errors are also measured easily by counting the number of manually replaced labels. Segmentation errors are measured either by the number of new PS components and new PCLs that result from the switching operations or by the percentage of line segments switched during the interactive error correction. A detailed discussion on error measurements is given in [22].

5 Results

Our database of test images came from four domains: logic circuits (13 drawings), electrical circuits (15 drawings), chemical plant flow diagrams (22 drawings), and flowcharts (14 drawings). Most of these were photocopied from books and scanned at two resolutions: 150 dpi and 300 dpi using HP ScanJet IIC. They were manually edited to remove text, then thinned and vectorized using MicroImages' MIPS package [13]. The recognition system was first developed on a different set of images and then tested on the above images that were not used during the development phase. After automatic recognition, errors were corrected interactively, as described in the last section. The performance measures reported below come from analyzing the data logged during the interactive correction phase.

5.1 Overall Performance

The overall summary of results appears in Table 2. The table includes result for each image scanned at the two resolutions. The table summarizes the results for all the images. Figs. 16, 17, 18, and 19 show a sample of the final results for each domain. The total execution time for the segmentation and understanding subsystems ranges roughly from a few seconds to a minute for our test drawings on a Sun 670MP. The interactive residue error correction increases the total execution time by a small percentage when it is needed. For the sample drawings shown in Figs. 16, 17, 18, and 19, the execution times are 24.7, 3.4, 68.1, and 5.8 seconds, respectively.

The recognition system performed better on the logic circuits and flow charts than on the electrical circuits and chemical plant diagrams. For logic circuits, 17 out of 26 images are perfectly segmented and labeled. The errors are low by any of the measures discussed above. For example,

TABLE 2
OVERALL ERROR COMPARISON FOR DRAWINGS OF DIFFERENT DOMAINS IN THE TEST SET

Drawing category	total # images	total # symbols	# images w/ no errors	percent[1] new PS	ratio[1] new PCL	percent[2] switched line segments	percent[1] label error
logic circuit	26	492	17	5.9	0.098	3.6	0.4\
electrical circuit	30	860	1	25.8	0.153	8.2	12.3
chemical plant	44	776	1	11.5	0.272	3.7	7.1
flowchart	28	452	25	0.0	0.007	0.1	0.0

[1] *compared to the total number of symbols in all drawings in the category.*
[2] *compared to the total number of line segments in all drawings in the category.*

TABLE 3
SYMBOL SEGMENTATION ERROR DISTRIBUTION FOR DRAWINGS IN DIFFERENT DOMAINS OF THE TEST SET

Drawing ID	number drawings	percent of new PS						
		0	1-10	10-20	20-30	30-40	40-50	> 50
logic circuit	26	22	1	1	1	0	0	1
electrical circuit	30	4	5	12	3	2	3	1
chemical plant	44	14	12	12	2	1	3	0
flowchart	28	28	0	0	0	0	0	0

TABLE 4
ERROR COMPARISON FOR DRAWINGS SCANNED AT 150 dpi AND 300 dpi

Drawing category	dpi	total # drawings	total # symbols	# dwgs w/ no errors	percent[1] new PS	ratio[1] new PCL	percent[2] switched line segments	percent[1] label error
logic circuit	150	13	246	8	6.1	0.098	3.1	0.4
logic circuit	300	13	246	9	5.7	0.098	3.9	0.4
electrical circuit	150	15	430	1	27.9	0.165	10.5	15.1
electrical circuit	300	15	430	0	23.7	0.142	6.7	9.5
chemical plant	150	22	388	1	11.9	0.271	3.8	7.7
chemical plant	300	22	388	0	11.1	0.273	3.7	6.4
flowchart	150	14	226	12	0.0	0.009	0.1	0.0
flowchart	300	14	226	13	0.0	0.004	0.0	0.0

[1] *compared to the total number of symbols in all drawings in the category.*
[2] *compared to the total number of line segments in all drawings in the category.*

as a percentage of the total number of symbols in all the drawings in this category, only 5.9 percent new symbols were created during interactive correction. For the flowcharts, 25 out of 28 images were perfectly processed. All the PS components in the 28 flowcharts are correctly separated. These drawings are simpler compared to the electrical circuit diagrams and chemical plant flow diagrams.

For electrical circuit diagrams, only one out of 30 is without errors of any kind, and, for chemical plant flow diagrams, only one out of 44 is without errors. Furthermore, new PS from interactive correction represent 25.8 percent and 11.5 percent of all symbols for electrical circuit diagrams and chemical plant flow diagrams, respectively. Many symbols are relabeled, 12.3 percent and 7.1 percent for electrical circuit diagrams and chemical plant flow diagrams, respectively.

Of all the errors, the number of new PS due to interactive error correction is the key index because correct segmentation is the basis for drawing understanding. Table 3 summarizes the error distribution of drawings in the test set. For the flowcharts, all symbols are correctly segmented. For the logic circuits, all symbols in 22 out of 26 drawing images are correctly segmented. The number of drawings with no error in symbol segmentation is four out of 30 and 14 out of 44 for electrical circuit diagrams and for chemical plant flow diagrams, respectively. Most drawings are seen to have 20 percent or fewer new PS components. Poorer performance is essentially limited to the two domains of electrical circuits and chemical plant diagrams.

5.2 Scanning Resolution

Table 4 compares the results of the two different resolutions. The 300 dpi images are recognized only slightly better than corresponding 150 dpi images, with the exception of electrical circuit diagrams. This may be because the inductors in a circuit diagram consist of many small oval loops as shown in Fig. 20. These loops tend to be misclassified at the loop matching phase because of the distortion introduced during vectorization and piecewise linear approximation. Correct classification of small loops is a problem in other drawings as well. Our system does not extract circular arcs in the vectorization step. Better vectorization that can extract small arcs may improve the performance. In general, we found that a minimum scanning resolution is required so that gaps are not introduced during scanning. For most drawings, we found the 150 dpi resolution to be adequate.

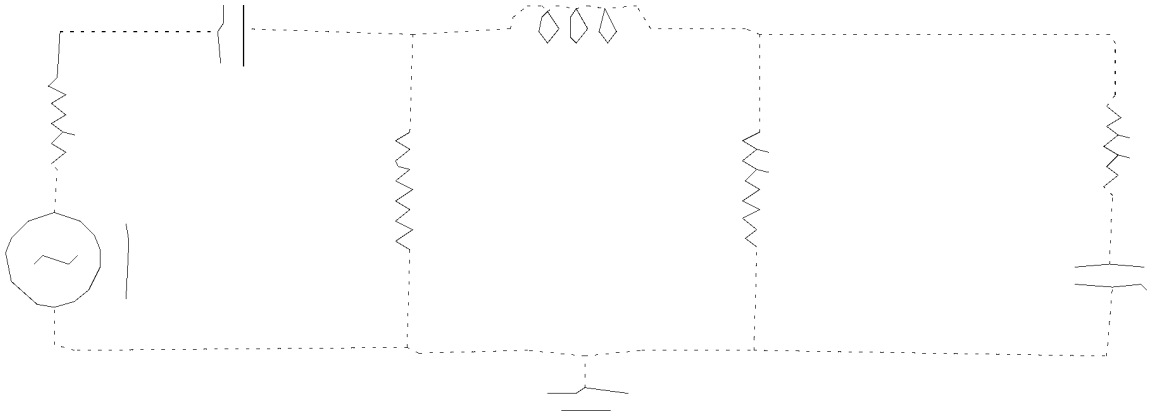


Fig. 17. Results for an electrical circuit diagram.

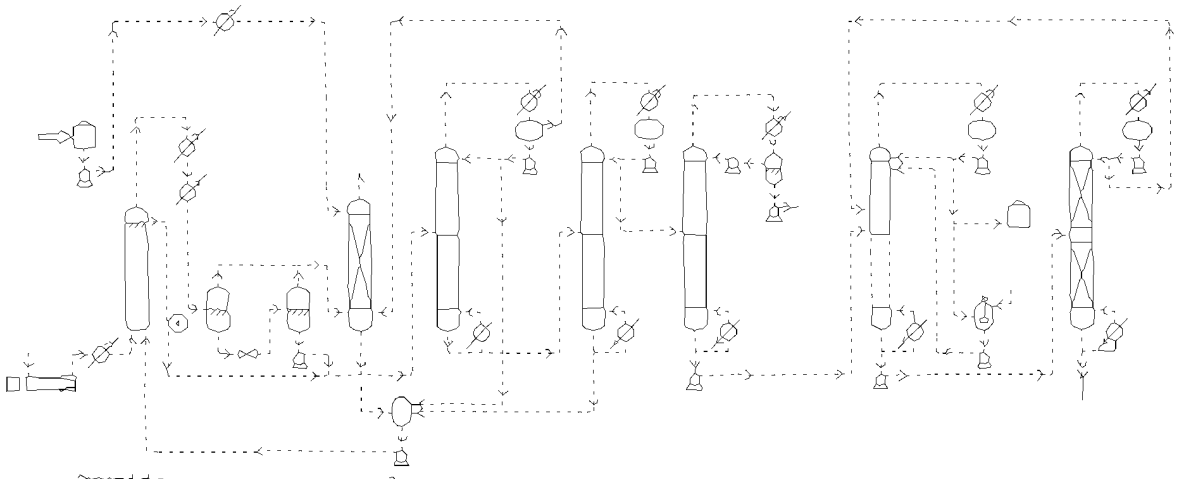


Fig. 18. Results for a chemical plant flow diagram.

5.3 Drawing Size and Quality

All our test drawings come from books and could fit within 8.5 inches × 11.0 inches. However, they differ substantially in size and quality. In terms of the area, the ratio of the largest to the smallest drawing in the four domains of logic circuits, electrical circuits, chemical flow diagrams, and flow charts were, respectively, as follows: 28.6, 11.5, 7.8, and 5.0. Some of the chemical flow diagrams and vacuum-tube circuits come from old books with poor quality of paper and drawings. While we have not carried out a systematic study of the impact of the drawing size and quality on the system performance, we offer the following observations based on the limited data at hand.

The physical size of a drawing, to the extent that it denotes the complexity of symbols and interconnections, will have a proportional impact on the execution time. The gap-size parameters might also need to be tuned for larger size drawings. The symbol-recognition accuracy, on the other hand, might actually improve for drawings of larger size because the system is more apt to misclassify small entities than large entities. This is a consequence of the normalized template based matching used in symbol recognition. We expect that for drawings of very poor quality, further image enhancements or more sophisticated matching algorithms will be required.

5.4 Performance Impediments and Improvements

Even within a domain, our test drawings come from a variety of sources. As such, they do not follow a common drawing standard and otherwise lack consistency that one would expect in a typical conversion task in a production environment. We found it hard to build libraries for a group of drawings that use different conventions. The size checks can not be executed effectively due to the same reason.

The current system is also limited in performance by the simplicity of matchers we implemented. Better matchers would surely improve performance, especially for the more complex electric circuit and chemical plant diagrams. The modular design allows for easy incorporation of better matchers in our system. It is worth noting, however, that even the simple matchers yield an acceptable performance because the system is able to consider alternative matches in light of high-level information during symbol classification.

The present implementation does not include a learning mechanism that would allow the system performance to improve in time on similar drawings. However, automatic measurement of errors during the interactive correction phase provides the appropriate data for dynamic adjustment of parameters, thus incorporating a form of learning. Additional useful data for learning can be compiled during the automatic error correction phase.

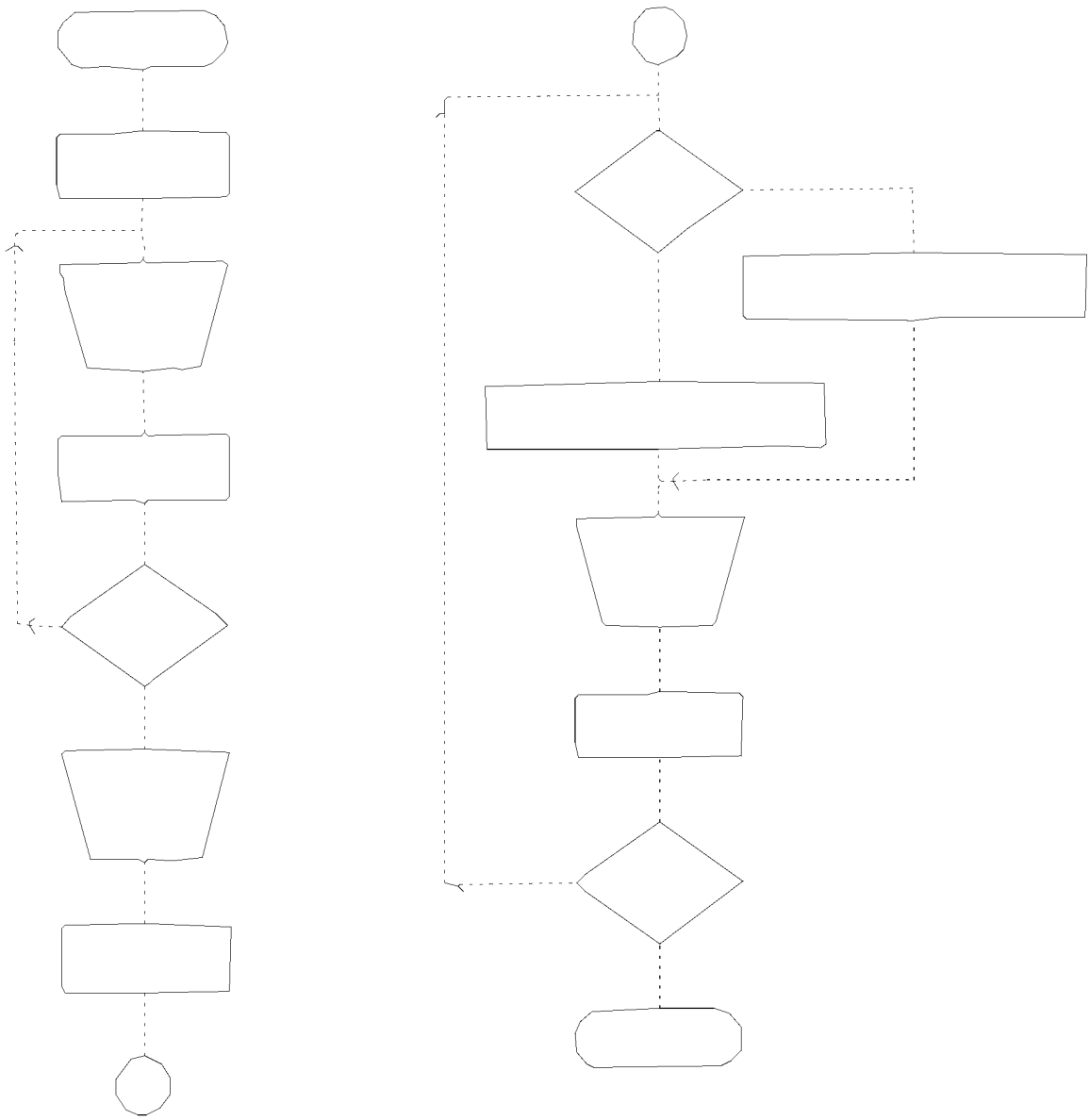


Fig. 19. Results for a flowchart diagram.

The present implementation is also not complete in that it assumes that available techniques will be used to automate the manual removal of text. These techniques cannot always remove characters that touch a connection line or a symbol or may remove such characters only partially. Characters or character fragments that touch a connection line are likely to be classified as unidentified symbols. A character or fragment touching a symbol may either be recognized as an unidentified symbol or be merged with the touching symbol without affecting its recognition (because of the tolerance embedded into the matching algorithm). As the system does not explicitly allow for touching characters, any errors caused because of them would need to be corrected interactively.

One of the deficiencies of our system is that we approximate the curves by a series of strokes. In a flow drawing arcs may appear in three forms:

1) a connection line is an arc or has parts that are arcs,
2) a symbol component has a loop parts of which are arcs,
3) a nonloop symbol component is an arc or has parts that are arcs.

Recognizing the arcs that are parts of loops does not improve our system, since a template matching scheme is used for symbol recognition. Using arcs to represent connection lines is uncommon and, hence, finding arcs in connection lines is also unlikely to provide much benefit to the system. However, for the entities in the third category, using an arc representation is better than our approximation by a series of strokes. The recognition of such symbols will be improved using this representation.

Finally, as in any other drawing recognition system, the performance deteriorates with the quality of drawings, especially when the drawing is crowded as in Fig. 20.

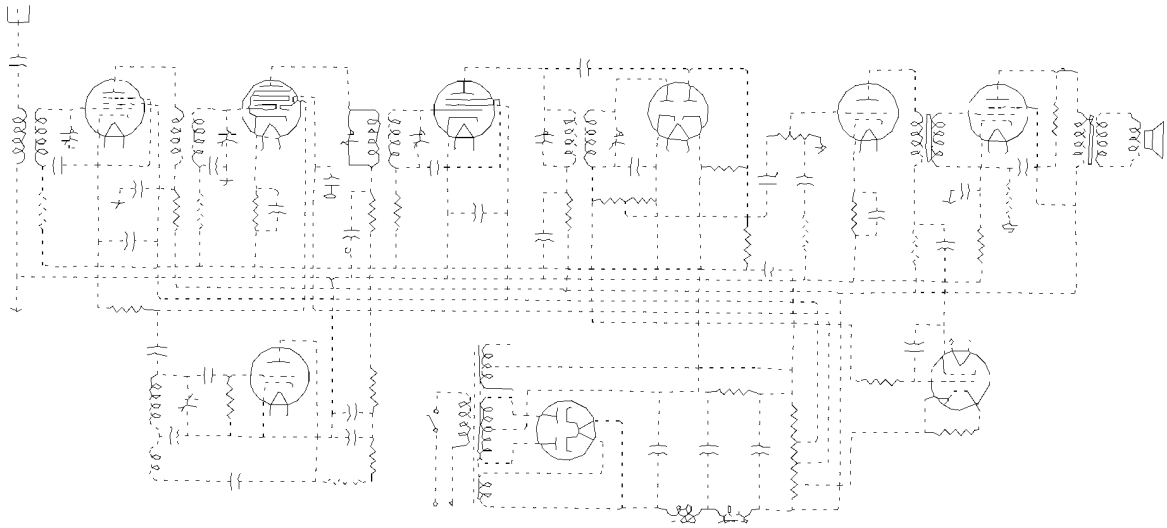


Fig. 20. A more complicated electrical circuit diagram after segmentation and traversal. It was scanned at 150 dpi.

5.5 Netlist Output

One of the intended outputs of the system was a description of the drawing in the netlist format. Clearly, if the drawing is not completely recognized, this description is not very useful. However, with the interactive correction module, a given drawing can be fully recognized. The system then can easily produce the netlist but currently produces an output that is very similar to the netlist format. This output can be tailored to produce the netlist or any other desired format.

6 Comparison with Prior Work

Most of the earlier drawing analysis systems targeted a single domain of drawings [1], [4], [6], [16]. Recent work has emphasized the need for systems that can be adapted to different types of drawings, even though experimental results are presented for only one domain [7], [9]. It is not clear how these systems would adapt to more complicated drawings in other domains. Furthermore, automatic drawing analysis systems seldom produce perfect results but error measurement and system evaluation issues have been rarely discussed in the literature.

Lin et al. [12] is an early work that deals with *connection diagrams* of various types, including flow charts, logic and electrical circuit diagrams, and piping and instrument diagrams. Thus, the target applications are the same as reported here. The key idea in this work is to examine the bilevel scanned image under a resolution comparable to the typical line width, and label each square according to the pattern of black pixels found at its boundary. The authors show that, when a proper mesh size exists and is used, only 51 different characteristic patterns can occur; an additional "?" pattern label is added to catch deviations from these that can occur with an improper mesh size. All further analysis of the image is carried out entirely in terms of the coarser resolution determined by the characteristic patterns. Based on this analysis, a *feature point graph* is constructed; it has vertices corresponding to feature points (ends, corners, and intersections) and edges corresponding to straight-lines or smooth curves joining two feature points. The image is then segmented into *symbols*, *connecting lines*, and *characters* by applying a set of heuristic rules to the feature point graph. In the final step, symbol loops are identified by normalized area features, symbols are recognized, and the connection-relation between them is extracted.

In comparing our approach against Lin et al. we note that our use of a commercial program for thinning and vectorization in the preprocessing stage is also motivated by the desire to avoid looking at fine pixel-level data in subsequent processing. However, our approach avoids having to make a manual decision about the critical mesh-size parameter. The segmentation rules used by Lin et al. are more complex than ours and make use of several threshold parameters, e.g., for the lengths of loops and lines, and the area of vertices in the feature point graph. In their system, segmentation results are not subject to change during symbol identification, as they are in our system.

To compare our system to one that makes explicit use of domain knowledge, we briefly describe the system presented in [19]. The system is designed for understanding of underground wiring diagrams.[3] It employs relational structures to represent objects, a blackboard to recognize the symbols, and specialized techniques to recognize lines, triangles, circles, and text since the drawing contains only these entities. Our approach also uses linear entities in the analysis. However, we group all the closed shapes into one category called "loops." Knowledge in our system is organized as a few domain independent rules for segmentation and an iconic symbol library to capture their shape information.

3. Note: This is not a flow drawing, but the observations are valid.

Pasternak [18] has also developed a system for engineering drawing recognition. Objects are represented in a hierarchy with inheritance and are described using a declarative language. A blackboard system is used to aid the recognition process. The system presented here has some major differences. Our system understands the drawing at a higher conceptual level. There is a clear notion of inputs and outputs and connectivity between symbols that is absent in [18]. Therefore, it is easier to fix errors during the understanding phase of the system.

Our system has a segmentation step that uses domain independent knowledge before symbol recognition. In [18], there is no distinct step. The system directly attempts to recognize the symbols. Since we exploit the inherent properties of flow drawings, we are able to minimize the complexity and efficiency of high level understanding. After the segmentation process, the drawing is reduced to isolated grouping of potential symbols and connection lines. Thus, the symbol recognition process is local and is much simpler. The blackboard system, by its very nature, induces more processing and control.

As a consequence of the use of properties of the flow drawings, our approach is not as general as the one presented in [18]. However, many of its aspects can be easily incorporated into our sytem. The matcher is one important example. Our system is designed to allow any type of matcher to seamlessly integrate. We model the symbols at a geometric level using loops and strokes as the primitives. We believe that for 2D symbols, our representation is adequate. However, the declarative style used to represent knowledge in [18] can be incorporated into our system.

One of the advantages of using a blackboard system is the ease with which feedback is obtained. In our system, there is no direct feedback between segmentation and understanding. However, we have a structure similar to a blackboard which is used in the understanding process. During the split-and-merge phase of the processing, the system does support a form of feedback.

The system described by Nardelli et al. [15] for interpretation of raster images of maps and office documents has organizational similarities to our system. Their three-step recognition method proceeds from lexical objects (characters and lines) to basic structured objects (that are building blocks of all structured objects), to documents themselves. In this framework, distinction between connection lines and symbols would be made only at higher stages of processing. In contrast, the segmentation rules in our system allow the separation of "lexical units" that belong to connection lines from those that belong to symbols with high degree of reliability, in the first stage of processing.

CELESSTIN [21] is another system based on blackboard architecture designed for document interpretation. Its goal is to derive a CAD model from a mechanical engineering drawing. "Specialists" are used to analyze and update the blackboard and a linear strategy is used to control the processing. It is demonstrated for a narrow subdomain of mechanical engineering drawings. The system uses extensive domain knowledge that is "difficult to manage, in terms of consistency rules and clarity." In contrast, our system uses only a few simple rules for segmentation that are easy to understand and manage. The symbol descriptions are stored in a modular and easily organized library.

Mapsee [14] describes a general approach to computer vision that combines a model-based representation for visual knowledge with constraint satisfaction techniques. Mapsee's knowledge base is a collection of schema models tied together by compositional and specialization hierarchies. The scene interpretation problem is formulated as finding an assignment of labels to the nodes of a graph that does not violate any constraints specified as predicates over the node labels. There are analogs for most parts of Mapsee in our system, even though our discussion is not presented in terms of scene constraint graphs. As in Mapsee, our labels are hierarchically organized, e.g., electrical circuit composed of symbols and connection lines, symbols composed of either single-component symbols, such as inductors and transistors, or multiple-component symbols, such as a capacitor, and so on. Also, in common with Mapsee, we use a combination of data driven and model driven approach to scheduling computation. Data-driven computation is exemplified in the initial stages of symbol recognition when well-matched symbols become the starting points of search. The split and merge procedures as well as recognition of multicomponent symbols, on the other hand, exemplify model-driven search.

Truve [20] presents a general approach to computational vision and demonstrates it with an application in blocks world. While there are similarities between the parsing-interpreting-pruning steps in this approach and segmentation-understanding steps in our approach, they are fundamentally different. The approach proposed by Truve is a strictly a syntactic approach based on a generalization of attribute grammars. It is difficult to adapt such an approach to practical domains of complex and noisy drawings addressed by our approach.

There is a large body of literature in engineering drawing analysis that addresses domains other than flow drawings. Here we provide pointers to significant milestones for the interested reader.

Engineering drawings that are drawn by hand, without the aid of drawing tools, are not explicitly considered in our work. The issue of on-line vs. off-line recognition is just as important for such drawings as it is for handwriting. In principle, our system should work for off-line recognition of hand-drawn flow drawings, though we provide no experimental data to back this claim. Kozima and Toida [10] describe an interactive method for recognizing hand-drawn figures by recording and analyzing adjacent-strokes features. Off-line recognition of hand-drawn electrical circuit symbols of arbitrary size and orientation is considered by Lee [11].

Chemical structure diagrams resemble flow drawings in many ways but lack the concept of flow. A prototype system for chemical graphics recognition has been developed by Casey et al. [2]. Okazaki and Tsuji [17] illustrate their rule based recognition system in terms of chemical structural formulas. The rules are activated using a data-driven strategy. In principle, their system can be adapted to other classes of drawings by a redefinition the hierarchical rule base.

7 Conclusion

Most engineering drawing analysis systems are designed to process only a specific type of drawings. In this paper, we described an engineering drawing understanding system that processes a variety of drawings. The system is a framework made of domain-independent algorithms and domain-specific knowledge bases. Addition of an interactive correction module makes the system complete (except for some low level modules, e.g., text removal). The segmentation algorithm, the symbol libraries, the symbol classification algorithm, their implementation, and results form the main contribution of this research. The performance is high for simpler drawings and good for complicated drawings. Changing from one domain of drawings to another requires only a change of the libraries.

We demonstrate that a general system using a few generic rules and a simple matcher can understand, i.e., recognize the symbols, determine their layout and connectivity, in complicated flow drawings fairly well and recognize simpler drawings completely. Instead of limiting the system to only one domain of drawings, our system handles drawings of similar nature but of different domains. The alternation of symbols with mostly rectilinear connections is the primary constraint that a domain must satisfy for our scheme to apply. The results in the paper demonstrate that the admissible domains include electrical and logic circuits, chemical plant diagrams, and flowcharts. Other drawings that are likely to satisfy the constraint are pipe and instrumentation diagrams, wiring diagrams, PERT charts, and entity-relationship charts.

Glossary

CL: Connection Line, the line that connects two symbols.

PCL: Potential Connection Line, a non-closing sequence of line vectors that is likely to be a connection line.

PS: Potential Symbol, a set of line vectors that is likely to be part of a symbol.

SC: Symbol Component, a set of connected line vectors that is part of a symbol.

S: Symbol, a complete symbol, it may consist of more than one symbol components that are detached from each other.

GM: Goodness Measure, they are defined for closed loop and non-closing strokes and weight-averaged based on entity length when entities are combined.

PML: Poly Maximal Line, a sequence of maximal lines connected by degree-two points. A maximal line is a maximal set of connected line segments that define a straight line in the drawing. The two end points of a poly maximal line should be distinct.

Software

Researchers interested in a copy of the software developed in this project may contact any of the authors directly.

Acknowledgments

We thank Prof. George Nagy for suggesting the topic and providing much critical feedback during this research. We are appreciative of help from MicroImages, Inc., Lincoln, Nebraska, in allowing the use of their MIPS package for thinning and vectorization; Dr. Lee Miller and Mr. Mike Unverferth very graciously responded to our call for help. We would also like to thank the Mathematics and Computer Science Department of Arkansas State University for letting Yuhong Yu use their facilities to finish her dissertation. We are also indebted to the reviewers for their careful, detailed, and constructive comments. We also thank the Center for Communication and Information Science at the University of Nebraska-Lincoln for financial and laboratory support.

References

[1] D. Benjamin, P. Forgues, E. Gulko, J.-B. Massicotte, and C. Meubus, "The Use of High-Level Knowledge for Enhanced Entry of Engineering Drawings," *Proc. Ninth IEEE ICPR*, pp. 119-124, 1988.

[2] R. Casey, S. Boyer, P. Healey, A. Miller, B. Oudot, and K. Zilles, "Optical Recognition of Chemical Graphics," *Proc. Second Int'l Conf. Document Analysis and Recognition*, pp. 627-632, Oct. 1993.

[3] D. Dori, "Object-Process Analysis: Maintaining the Balance Between System Structure and Behavior," *J. Logic Computation*, pp. 1-23, 1995.

[4] C.-S. Fahn, J.-F. Wang, and J.-Y. Lee, "A Topology-Based Component Extractor for Understanding Electronic Circuit Diagrams," *Computer Vision, Graphics, and Image Processing*, vol. 44, pp. 119-138, 1988.

[5] F.C.A. Groen, A.C. Sanderson, and J.F. Schlag, "Symbol Recognition in Electrical Diagrams Using Probabilistic Graph Matching," *Pattern Recognition Letters*, vol. 3, pp. 343-350, 1985.

[6] R.W. Haddad, "Drip: A Schematic Drawing Interpreter," Technical Report 95/1, Western Research Laboratory, Palo Alto, Calif., Mar. 1995.

[7] A.H. Hamada, "A New System for the Analysis of Schematic Diagrams," *Proc. Second Int'l Conf. Document Analysis and Recognition*, pp. 369-372, Oct. 1993.

[8] R. Kasturi, S.T. Bow, W. El-Masri, J. Shah, J.R. Gattikermes, and U.B. Mokate, "A System for Interpretation of Line Drawings," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 978-992, Oct. 1990.

[9] S.H. Kim, J.W. Suh, and J.H. Kim, "Recognition of Logic Diagrams by Identifying Loops and Rectilinear Polylines," *Proc. Second Int'l Conf. Document Analysis and Recognition*, pp. 349-352, Oct. 1993.

[10] H. Kojima and T. Toida, "On-Line Hand-Drawn Line-Figure Recognition and Its Application," *Proc. Ninth Int'l Conf. Pattern Recognition*, pp. 1,138-1,142, Nov. 1988.

[11] S. Lee, "Recognizing Hand-Drawn Electrical Circuit Symbols with Attributed Graph Matching," *Structured Document Image Analysis*, H. Baird, H. Bunke, and K. Yamamoto, eds., pp. 340-358. Springer Verlag, 1992.

[12] X. Lin, S. Shimotsuji, and M. Minoh, "Efficient Diagram Understanding with Characteristic Pattern Detection," *Computer Vision, Graphics, and Image Processing*, vol. 30, pp. 84-106, 1985.

[13] MicroImages, Inc., *TNTmips V4.40: Map and Image Processing System*. Lincoln, Neb., 1994.

[14] I. Mulder, A. Mackworth, and W. Havens, "Knowledge Structuring and Constraint Satisfaction: The Mapsee Approach," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 10, no. 6, pp. 866-979, Nov. 1988.

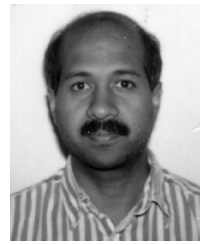
[15] E. Nardelli, M. Fossa, and G. Proietti, "Raster to Object Conversion Aided by Knowledge Based Image Processing," *Proc. Second Int'l Conf. Document Analysis and Recognition*, pp. 951-964, Oct. 1993.

[16] A. Okazaki, T. Knodo, 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*, vol. 10, no. 3, pp. 331-341, May 1988.

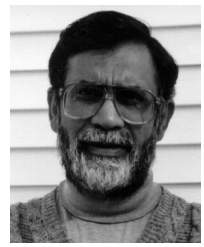
[17] S. Okazaki and Y. Tsuji, "Knowledge-Based Approach for Adaptive Recognition of Drawings," *Proc. Pattern Recognition: Fourth Int'l Conf.*, pp. 627-632, Mar. 1988.

[18] B. Pasternak, "Processing Imprecise and Structural Distorted Line Drawings by an Adaptable Drawing Interpretation Kernel," *Proc. IAPR Workshop Document Analysis Systems*, pp. 349-363, Kaiserlautern, Germany, Oct. 1994.
[19] S. Shimotsuji, O. Hori, and M. Asano, "Robust Drawing Recognition Based on Model-Guided Segmentations," *Proc. IAPR Workshop Document Analysis Systems*, pp. 337-347, Kaiserlautern, Germany, Oct. 1994.
[20] S. Truve, "Image Interpretation Using Multi-Relational Grammars," *Proc. IAPR Workshop Document Analysis Systems*, pp. 313-321, Kaiserlautern, Germany, Oct. 1994.
[21] P. Vaciviere and K. Tombre, "Knowledge Organization and Interpretation Process in Engineering Drawing Interpretation," *Proc. IAPR Workshop Document Analysis Systems*, pp. 313-321, Kaiserlautern, Germany, Oct. 1994.
[22] Y. Yu, "A System for Engineering Flow Drawing Understanding," PhD thesis, Univ. of Nebraska-Lincoln, Aug. 1994.
[23] Y. Yu, A. Samal, and S. Seth, "Isolating Symbols from Connection Lines in a Class of Engineering Drawings," *Pattern Recognition*, vol. 27, no. 3, pp. 391-404, Mar. 1994.

Yuhong Yu received her BS degree in chemical engineering from the National Taiwan University in 1974 and her MS degree in chemical engineering from Purdue University in 1977. She worked as a chemical engineer for Amoco, the C.W. Nofsinger Co., and Bechtel, respectively, until 1986, when she began her computer science career. She received her MS and PhD degrees in computer science from the University of Nebraska-Lincoln in 1989 and 1994, respectively. Since then, she has been working in the area of computer networking and is currently with Lucent Technologies in Naperville, Illinois.



Ashok Samal received the BTech degree in computer science from the Indian Institute of Technology at Kanpur, India, in 1983. He received his PhD in computer science from the University of Utah in 1988 in the area of parallel computer vision. He has since been with the Department of Computer Science and Engineering at the University of Nebraska-Lincoln, where he is currently an associate professor. His primary research interests are document analysis, computer vision, and distributed computation. He is a member of the IEEE, ACM, and Pattern Recognition Society.



Sharad C. Seth received his PhD degree in electrical engineering from the University of Illinois at Urbana-Champaign in 1970. He has since been with the Department of Computer Science and Engineering at the University of Nebraska-Lincoln, where he is currently a professor. Dr. Seth serves on the editorial board of the *Journal of Electronic Testing: Theory and Applications* (JETTA). He is a fellow of the IEEE and a member of the Computer Society of the ACM and the VLSI Society of India. His current research in document image analysis is concerned with map understanding and page segmentation schemes.