CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

Computer Interpretation of Process and Instrumentation Diagrams

By

T. Binford, T. Chen,
J. Kunz, and K. H. Law

CIFE Technical Report #112
August, 1997

STANFORD UNIVERSITY

If **you would like to contact the authors please write to:**

c/o CIFE, Civil and Environmental Engineering Dept.,
Stanford University,
Terman Engineering Center
Mail Code: 4020
Stanford, CA 94305-4020

SUMMARY

CIFE TECHNICAL REPORT #112

Header: CIPID

Title: Computer Interpretation of Process and Instrumentation Diagrams

Authors: T. Binford, T. Chen, J. Kunz, and K.H. Law

Publication Date: July, 1997

Funding Sources:

- **Name of Agency: CIFE**
- **Title of Research Project: Seed Research Project**

1 Abstract:

This report describes a prototype system that interprets Process and Instrumentation Diagrams (P&IDs). The system takes a P&ID in a vector CAD file in DXF format and produces an intelligent P&ID model that has symbolic interpretations for system components (e.g., check valve, pumps, etc.) and symbolic attributions (cold water, propane, etc.).

The system finds structure among vectors, recognizes graphic symbols as physical components from a database of components (e.g., valves), interprets annotations that assign text to graphic symbols. Representation of diagrams is a hierarchy of classes based on cellular topology and geometry (the Area-Curve-Vertex graph (ACV)) at the symbol level. Presently, there are about 100 graphic symbols in the database.

The system has recognized 95% of covered graphic symbol instances for a full page of an industrial drawing, with 89% of graphic symbol instances covered (in the database). A drawing with 5,000 vectors required 20 minutes on an SGI Indy 4400SC (approximately 10% faster than a P5 100MHz on SPECint92). For symbols that are not yet in the database, a "learning" mechanism and a GUI is being developed to construct the ACV data structure of the unknown symbol.

2 Subject:

- **What is the report about in laymen's terms?** Turning vector CAD P&IDs into symbolic models.
- **What are the key ideas or concepts investigated?** Representation of symbol geometry, effective symbol recognition, computational complexity feasible for industrial applications.

- **What is the essential message?** A theoretical-based scalable interpretation system has been developed. The system is effective, feasible, and promising for short-term exploitation.

3 Objectives/Benefits:

- **Why did CIFE fund this research?** IAB mentioned this as one important problem.
- **What benefits does the research have to CIFE members?** Possible near-term applications of the prototype for drawing interpretation, demonstration of success in automated P&ID interpretation, dealing with legacy drawings, compliance with as-built, retrofitting, interpretation of vectorized drawing with symbolic models.
- **What is the motivation for pursuing the research?** Value of applications, technical interest and challenge.
- **What did the research attempt to prove/disprove or explore?** The problem was originally thought to be too difficult; there had been expensive failures by the industry. Our research demonstrates effective geometric computation for a broad class of diagrams.

4 Methodology:

- **How was the research conducted?**
  Phase I: feasibility demonstration - on the order of 10 symbols. Phase II: a prototype system - on the order of 100 symbols (including the development of a "learning" mechanism for the identification of unknown symbols). Phase III: scale-up.
- **Did the investigation involve case studies, computer models, or some other method?** An industrial, typical, single-sheet P&ID was used for the investigation. A theoretical class hierarchy based on previous work was developed for the representation of graphic symbols.

5 Results:

- **What are the major findings of the investigation?** The approach is well-founded, very successful, and extensible.
- **What outputs were generated?** A conference paper, a journal article (accepted), and a software prototype.

6 Research Status:

- **What is the status of the research?** Seeking funding to continue development and to link with other applications.

- **What is the logical next step?** Develop industrial prototype, add further capabilities (i.e., learning new and unusual symbols).
- **Are the results ready to be applied or do they need further development?** The results so far are ready for initial productizing.
- **What additional efforts are required before this research could be applied?** Build a product and develop a pilot project to exploit and aid in productizing.

Computer Interpretation of Process and Instrumentation Diagrams

T. Binford, T. Chen, J. Kunz, and K.H. Law
Stanford University, Stanford, CA 94305, USA

July 30, 1997

1 Introduction

Process and Instrumentation diagrams (P&IDs) describe the components of a plant and their logical connectivity. Figure 1 shows one sheet of a representative P&ID. Some lines represent pipes. Clusters of vectors indicate components, e.g., valves and pumps. Some of the graphic symbols are annotated with text and symbols, e.g., to indicate sizes of pipes and valves or functions of elements. A P&ID for a large process plant may include 100-1000 sheets. A small fraction of new designs are made in intelligent P&ID design systems (said to be 1%), but the bulk of new and existing designs may be made with non-intelligent CAD format. There is also a legacy of P&ID designs on paper only. The motivation for interpretation of P&IDs is to build intelligent, symbolic P&IDs for automated analysis, in support of retrofitting, for maintenance, for as-built specification, and for compliance with regulations.

Across the spectrum of P&IDs, we expect there are typically a few hundred symbols that occur frequently, a similar number that occur infrequently in many P&IDs, and a few special symbols that are non-standard and may occur only in an individual drawing. We have implemented and tested a powerful mechanism to accommodate most of these symbols: generic representation of symbols, generation of generic symbol hypotheses, and a structured symbol database. We have made an effective implementation that was supported by a hierarchical class representation based on cellular topology.

Figure 2 shows the plant composition model that represents structures throughout a process plant multi-sheet drawing. Each arc in the figure represents an interpretation step. Thus, for example, given a set of vectors, Step 2 of the interpretation process identifies the geometric structure and annotation text. The hierarchy of levels in the plant model covers symbolic, functional levels and geometric, graphic levels. Levels up through identification of components and assignment of annotation are more or less local and graphic in that they deal with information that is local in the diagram or implicit in conventions. Levels from circuits and up are non-local in that they deal with interpretations that are transmitted along connected components and pipes, potentially

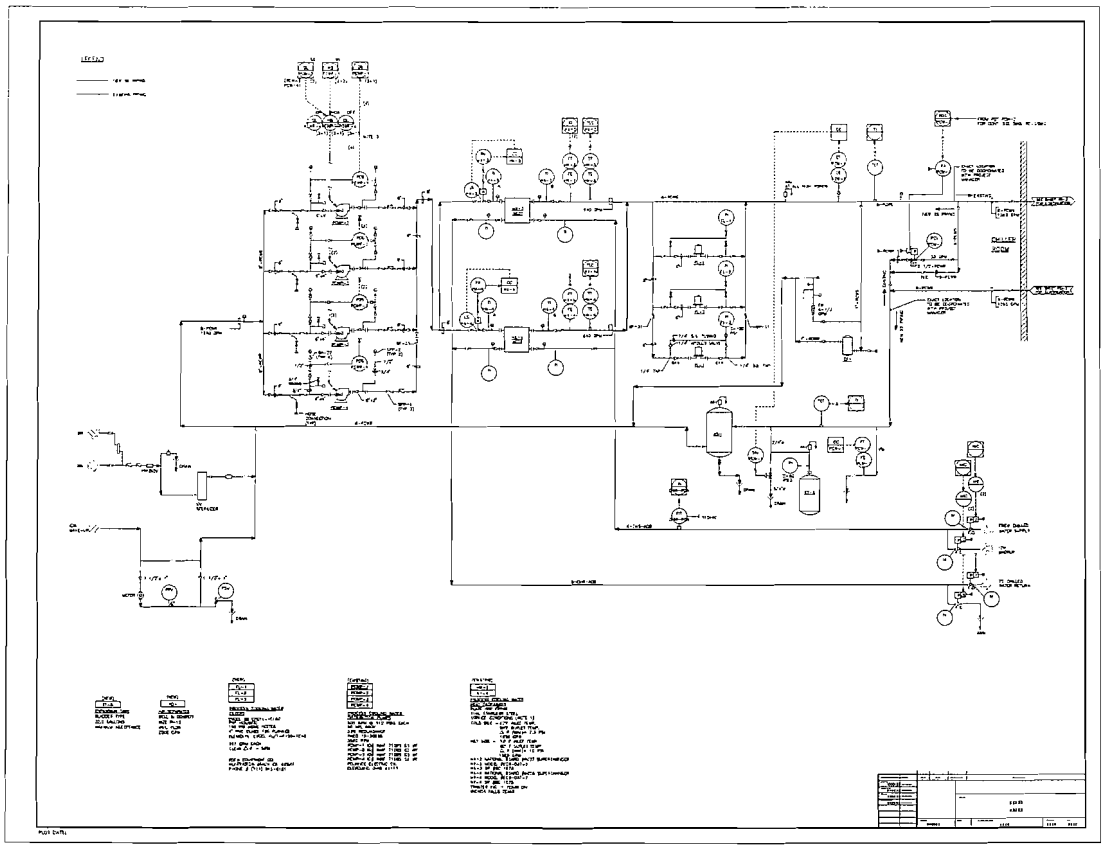

Figure 1: One sheet of a Representative Industrial Process and Instrumentation Diagram (P&ID). Most lines on the drawing represent symbols for plant components or pipes connecting components; some lines are part of text annotations, and others divide the drawing visually into logical sections.

over large parts of the diagram. These levels deal with constructs that are functional, e.g., chilled water subsystem.

The P&ID interpretation system accepts a vector-based representation of a P&ID. The system has a generic database that represents defined components in terms of areas, curves and vertices (ACV). The system analysis methods first create an ACV data structure for a set of input vectors; then hypothesize components that may fit the data; and finally, match the input ACV with the hypothesized ACV's to find the component definitions that best fit the data. The resulting symbolic plant model represents the identified components (i.e., pipes, valves, pumps), their connectivity (i.e., pipe connections) and their identifiable attributes (e.g., dimensions, constituent fluids) that can be identified as annotations in the original P&ID drawing.

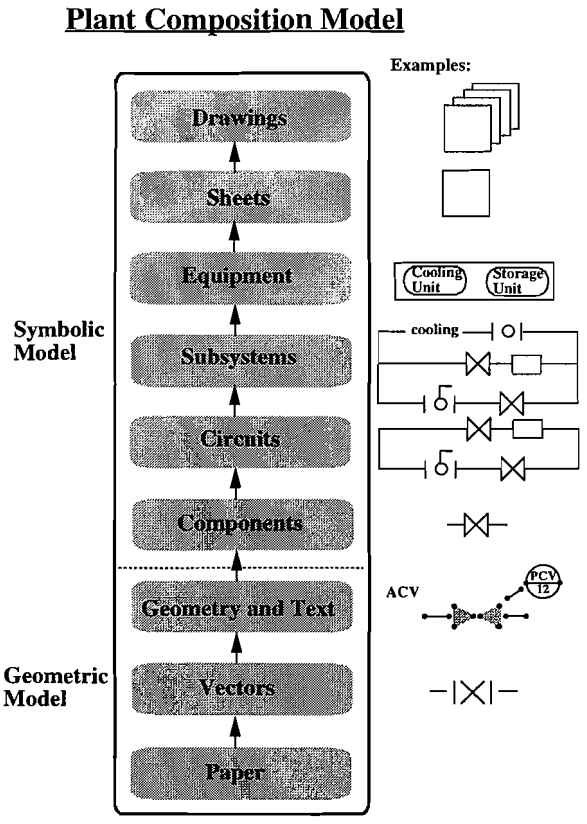

Figure 2: Plant Composition Model and Examples. The plant composition model is used to represent a process plant. The different levels in the model are conceptually part of the geometric model or symbolic model. The geometric model is used to facilitate graphic interpretation (vector-to-symbol recognition). The symbolic model enables functional interpretation of P&IDs.

2 Status

The system now analyzes full, single sheets of P&IDs. A user defines connections between sheets interactively using a GUI. An input parser reads DXF files to extract entities and blocks. A display utility renders entities and blocks with pan and zoom. A module determines colinear relations among vectors that determine redundant, overlapping vectors that are very common; the module also determines colinear relations across gaps. A cellular topology structure is constructed after determining intersections between vectors, i.e. , L, T, and X vertices. From intersections, areas, curves, and vertices are linked in an ACV graph that defines the cellular topology of hypotheses. A generic hypothesis generation mechanism determines clusters of vectors that are hypotheses for components. The database of component symbols is structured in a hierarchy based on the ACV. Recognition is done by comparing the ACV of the component hypothesis to the component model. Joints are added for later use.

Annotations are analyzed in a similar way. Annotation graphic symbols are structured in an ACV graph for recognition. Annotation graphic symbols and bounding box of text strings are used to assign text to component symbols, i.e., pipes, valves, etc. Parsing and interpretation of text is underway for instruments and components.

Two phases of development efforts have been demonstrated. A Phase 1 demonstration (feasibility) was done with a handful of components on a small part of a diagram. A Phase 2 demo was done with a full sheet from an industrial P&ID to demonstrate scaling up of the algorithms. The test was realistic in that it was a real P&ID. The graphic symbol database has about 100 symbols including component symbols and annotation symbols. Some statistical analysis of drafting in the test P&ID were used to make decisions about hypotheses and about computational complexity. The test diagram has about 5,000 vectors, with 356 symbol instances in 41 symbol classes. Of those symbol classes, 30 are covered in the symbol database and 318 (i.e., about 89%) symbols are covered instances. Of those covered symbol instances, 302 were recognized, i.e., 95% recognition of covered symbol instances. Overall recognition was 85% of all symbol instances. The system is developed in Allegro Common Lisp (with no declarations) on an SGI Indy R4400SC (150MHz) with performance about 92 SPECint92. The interpretation takes about 20 minutes. The efficiency can certainly be improved dramatically as we are continuing development of new algorithms and improving the implementation. For symbols that are not yet in the database, a "learning" mechanism is being developed to construct the ACV data structure of the unknown symbol.

3 Establishing Geometric Structure

A P&ID diagram contains a large number of vectors. The first task is to find relations among vectors such as colinearity and intersection.

For computational complexity reasons, it is useful to find colinear relations among vectors. At vector level (see Figure 2), colinear relations establish redundant, overlapping vectors (Z-vertices) (see Figure 3) which occur frequently with input and output pipes in a block overlapping pipes connecting components, and with the same vector occurring in multiple layers. At component level, a pair of colinear opposed T-vertices often corresponds to the I/O ports (see Figure6(a) for example) of a component and is used as a strong evidence for generic component hypothesis generation. At circuit level, cascading components are often connected by colinear pipes.

In the next step, colinear sets of vectors are used to build a cellular structure, which allows us to find L-vertices, T-vertices, and X-vertices efficiently.

3.1 Colinear Relations

P&IDs have predominantly horizontal and vertical vectors, but there are also vectors at various other angles. In some diagrams, a part of the drawing is set at an angle to give intuition about the setting in the plant. For computational efficiency, it is valuable to give preferential treatment to horizontal and vertical vectors, but the mechanism must

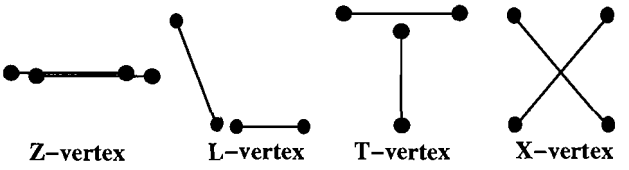

Figure 3: L-vertex, T-vertex and X-vertex. These are the four categories of connections commonly found in P&IDs.

treat relations among vectors at all angles.

We map vectors into colinear sets of horizontal, vertical, and other orientations. Colinear sets are stored as RB trees which are balanced binary trees with color on each node to ensure tree balancing[2]. Figure 4 (a) shows a horizontal colinear set of four line segments. The coordinates of the left endpoints are used as keys for sorting. Note that L1 and L2 are overlapped. Figure 4 (b) shows the binary tree for the colinear set. Each node contains a key for sorting and a pointer to the actual object. Overlapped lines (Z-vertices) can be found easily by performing tests within binary trees.

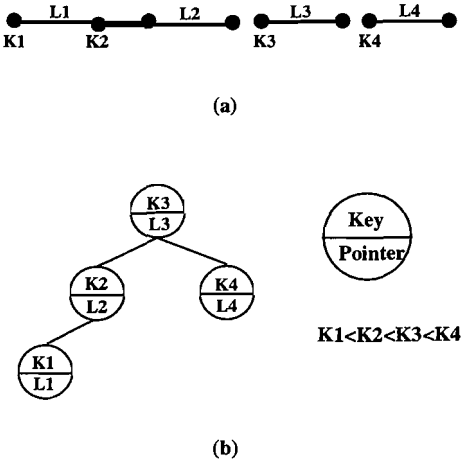

Figure 4: Colinear set and Binary tree. Figure (a) shows a horizontal colinear set and Figure (b) shows the binary tree for the colinear set.

3.2 L, T, and X vertices

Colinear relations are used in setting up the cellular topology using projection. All colinear sets are projected onto the horizontal axis and the vertical axis, as illustrated in Figure 5. A cellular structure is built up as an anisotropic grid adapted to the input drawing.

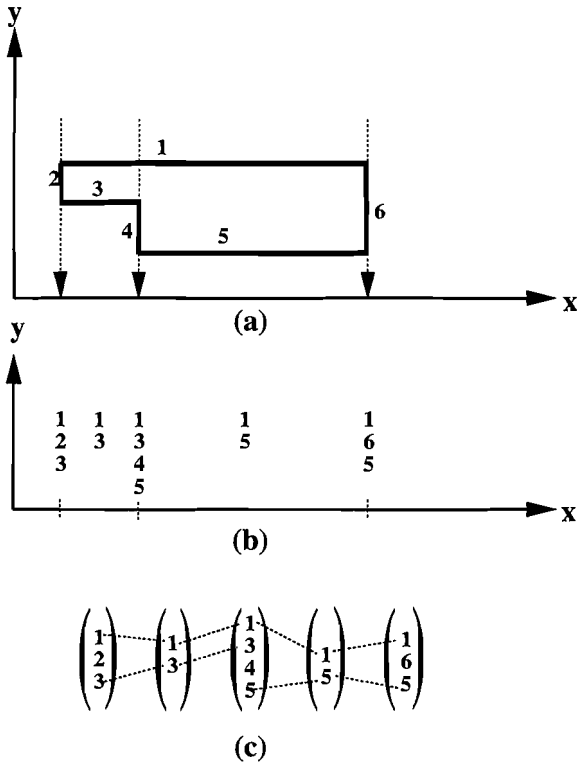

Figure 5: Projection and Cellular Structure. Figure (a) shows a circuit and the projection of the pipes (colinear sets) onto the horizontal axis. Figure (b) shows the stacks of lines produced by vertical projection. Figure (c) shows the partial cellular structure. A full cellular structure is constructed using both vertical and horizontal projections. Note that line segment 0 is linked to line segment 1 by colinearity.

As shown in Figure 3, two vectors may intersect at the endpoint of one (T-vertex), at endpoints of both (L-vertex), or at the interior of both (X-vertex). To find L-vertices, T-vertices, and X-vertices, searches in the cellular structure are performed.

As an example, the L-vertex formed by line segments 3 & 4 in Figure 5 can be found by performing two searches in the cellular structure. First, use the x coordinate of the right endpoint of line segment 3 as a key to search the stacks shown in Figure 5 (c), and locate the stack (1 3 4 5). Second, use the y coordinate of the right endpoint of line segment 3 as a key to search the stack (1 3 4 5) to find out that line segment 4 is connected to line segment 3. The overall computational cost is very low compared to brute force searches. Note that the data structures used for searching are RB-trees, which greatly facilitate the computations.

T-vertices, and X-vertices are found in a similar fashion. The computational cost is slightly higher for these vertices because we may need to perform searches in several stacks to find a vertex.

4 Graphic Interpretation

It would be possible to recognize graphic symbols by brute force model-based methods that match each component model in the database with each set of the corresponding number of vectors in the drawing. This approach would have the complexity:

$$N_m * \left(\begin{array}{c} N_v \\ m_v \end{array} \right) \left(= 100 * \left(\begin{array}{c} 5000 \\ 6 \end{array} \right) \right)$$

where N_m is the average number of component models in a P&ID drawing, N_v is the number of vectors in the drawing and m_v is the number of vectors in the component model. A typical drawing such as the one shown in Figure 1 may consist up to 5,000 vectors, contain up to 100 component types and, on the average, has 6 vectors per component model. The complexity is about $2 * 10^{21}$ model matches. We decided from the start that a brute-force method was computationally unacceptable. Such a method is also not useful because it does not identify relations among symbols that could be used for classification, e.g., in a specific valve class[1]. It does not indicate which vectors and symbols form a component when the component is unknown. The scientific approach chosen results in roughly a factor of 10^{19} fewer model matches.

The issues mentioned above motivate synthesizing a generic component hypothesis generator. Its objective is to generate hypotheses about the classes of components that best fit a cluster of vectors. Generic hypothesis generation will be especially important in learning new components and structures. The class definition also describes relations among classes and thus can be used to establish relations among hypotheses. The functional concept of input and output ports (flanges) was used to delineate components as connected vectors bounded by ports (see Figure 6(a)). The ACV graph for components is the area-curve-vertex graph of areas bounded by the connected vectors in a component (see Figure 6(b)). The ACV graph is a hierarchical description of graphic symbols as a structured network. ACV graphs are the basis for the database hierarchy and for symbol recognition.

The database is structured as a hierarchy with topological constraints first, then geometric constraints (see Figure 7). Areas are considered before vectors. The descriptors are: the number of ports, number of areas, shape of areas, orientation relationship between areas, and connectivity of vectors. These descriptors are computed from ACV graphs directly.

Symbol recognition is the equivalent of finding subgraph isomorphisms which is known to be NP-complete. Various methods have been proposed for matching graphs; most of them match planar graphs with no specific structures to a flat database of graphs. In other words, the graph being matched is compared to every graph in their databases, and the best match is then found according to some metric of closeness. Although those methods have some applications, their performance is not totally satisfactory. Those methods also do not provide a learning mechanism.

[1] We make use of similarities in valves to make a class hierarchy of valves. Similar class hierarchies are constructed for other component types.

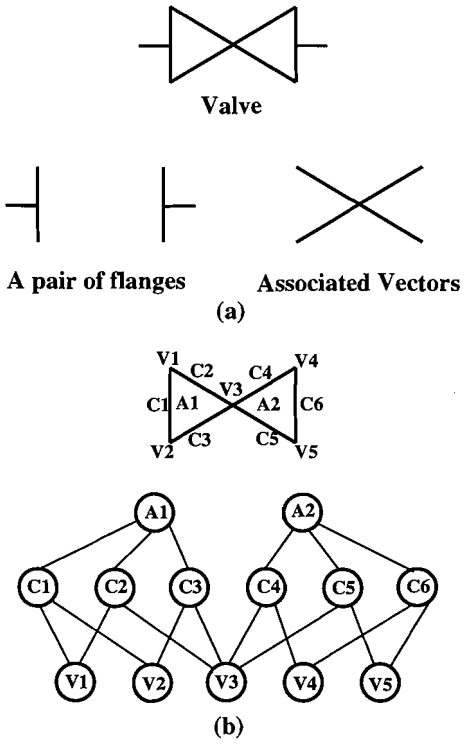

Figure 6: Generic Component Hypothesis Generation. Figure 6(a) shows a pair of flanges and associated vectors that constitute the graphic symbol of a valve. The pair of flanges, corresponding to the I/O ports of the valve symbol, is used as a strong evidence for generic component hypothesis generation. Figure 6(b) is the ACV(Area-Curve-Vertex) geometric description of the valve symbol.

In our approach, the hierarchical class definitions of similar objects greatly facilitate the symbol recognition process; symbol recognition is achieved by traversing the database hierarchy according to the topological and geometric constraints (computed from the ACV graphs) of the graphic symbols being matched. At least as important, in developing the GUI and in the future implementation of learning, the ACV graph of a newly observed component hypothesis is a new instance to be inserted into the model database of graphic symbols. Thus, the learning system will present the user with a component that belongs in a certain place in the hierarchy, e.g., a valve of some unknown type, similar to a known valve.

The focus of this work is to exploit the structures of graphs to enable reliable and efficient graph matching. If these were completely general graphs without domain knowledge and without topology and geometry of symbols, the recognition and learning problems would be extremely expensive computationally. In particular, ACV graphs are used instead of general, planar graphs. When constructing the database, symbols are inserted into the database hierarchy according to their ACV graphs. When matching an unknown symbol to the database, the database hierarchy is traversed according to

the ACV graph of the unknown symbol.

4.1 Automatic Construction of Unidentifiable Components in the Database

Even though the data structure for most of the known components can be established manually, unidentifiable symbols are still encountered in practice. This situation occurs relatively frequently in day-to-day industrial applications where drafting personnel may simply add a "new" symbol for convenience. Another issue is that manually constructing a database for an application containing very large number of components can be very time consuming. For a system to accomplish the full function of converting legacy unintelligent documentation to an intelligent model, the system must provide mechanisms to make adding a substantial number of new symbols easy. Substantial progress has been made in implementing that facility. A semi-automated procedure has been developed to construct the topological and geometric structures of unidentifiable components (i.e., symbols that are not known in the database) and to store the data structure in the database.

The basic procedure for each graphic symbol to be added to the database can be summarized as follows:

1. Delineate the symbol hypothesis.
2. Compute the ACV graph.
3. For each of the areas (subgraphs)
   compute the shape
   compute the size
   compute the order
   End
4. For each of the curves
   record its type
   End
5. For each of the vertices
   compute the degree of the vertex
   compute geometric relations of vectors at the vertex
   (e.g., two vectors are incident at a vertex and form a right angle)
   End
6. Compute relations of the areas(e.g., two areas may share a common vertex or a common vector (edge)).
7. Create a database class instance and build a path from the root of the hierarchy to the instance. (A path has segments that correspond to the topology and geometry of the symbol being added to the database hierarchy (see Figure 7).)

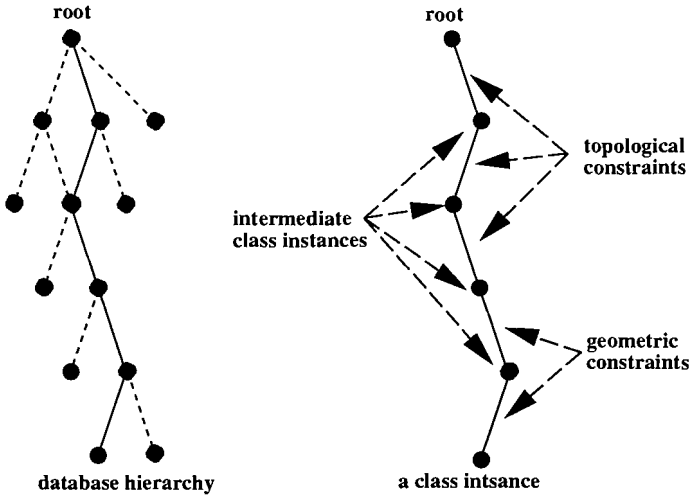

Figure 7: Left shows a class hierarchy; right shows a path in the hierarchy. Note that the nodes correspond to class instances and the arcs correspond to topological and geometric constraints.

A symbol is a graph consisting of a set of vertices and a set of vectors (edges) that connect them. The number of vectors is called "size", and the number of vertices is called "order". A vertex can be characterized by the number of incident vectors (degree) and a vector can be associated with its type (e.g., a line segment, a circular arc, or a circle).

The sizes and orders of areas (subgraphs) are used in building the first few segments; then, the relations among the areas are used in the following segments. Note that these segments characterize the topology of the graphic symbol being added; graphic symbols that share this set of segments are isomorphic to each other. In order to distinguish graphic symbols further, the geometric constraints of symbols (e.g., shapes of areas) are used.

The procedure used above requires minimal interaction by an user. Step 2 - Step 6 are executed automatically. In Step 1, the users only need to select a cluster of vectors using mechanisms provided by the GUI, and enter class name (and functional descriptions) of the symbol being added.

4.2 Symbol Recognition

Symbol recognition is achieved by generating symbol hypotheses and traversing the database hierarchy to find the best matches. Hypothesis generation amounts to segmenting out a few vectors that possibly constitute a symbol, from a sea of vectors. For example, a symbol hypothesis can be a cluster of vectors bounded by a pair of T-vertices (flanges). The purpose of hypothesis generation is to avoid the need of brute-force matching, matching all database model symbols against all tuples of vectors in

the diagram.

The symbol recognition procedure can be summarized as follows:

1. Generate generic component hypothesis.
2. Compute the ACV graph.
3. Perform Steps 3 to 6 of the symbol construction procedure as described in the previous section.
4. Traverse the database hierarchy using the topological and geometric information for a cluster of vectors established in Step 3.
5. Report a match or partial matches.

Note that the entire hierarchy is the space we seek to find a match. In the recognition process, each segment (of a path) in the hierarchy prunes the search space further, such that the search space becomes smaller and smaller as the recognition process proceeds.

The recognition strategy is fast because only a few class instances are actually compared to the hypothesis being matched. Topological constraints eliminate symbols that are not likely to match. The computational cost for topology checking is low. Geometric constraints differentiate isomorphic symbols and give the best match. If a full match cannot be found, partial matches are returned. For partial matches or unidentified symbols, user interaction is needed to update the database as described in the previous section. After the database is updated, the recognition procedure is invoked again.

5 Interpretation of Text and Annotation

Text and annotations carry attributes of components, e.g., the size of a pipe. Text and annotations are fundamental for reasoning about functions of components and subsystems in an intelligent P&ID. Three operations are essential in interpretation of text and annotation: 1) translation of text symbols; 2) interpretation of annotation symbols; 3) assignment of text and annotation to graphic components or subsystems. We have obtained an ANSI spec for instrumentation drawings[7]. A special parser and translator is being constructed for instrumentation and components.

Annotation symbols are interpreted using the same mechanism as interpretation of graphic symbols. Assignment of annotation to graphic components combines interpretation of annotation symbols with geometric information from the text string. Text can be in-line with, aligned with, enclosed in, above or below a graphic symbol. Annotation symbols point to or indicate graphic symbols (see Figure 8). Presently, the system automatically generates a bounding box to enclose each text annotation on a drawing. A semantic based (natural language) parser will be developed to interpret the text and attached the interpreted results as attribute information of the "intelligent objects."

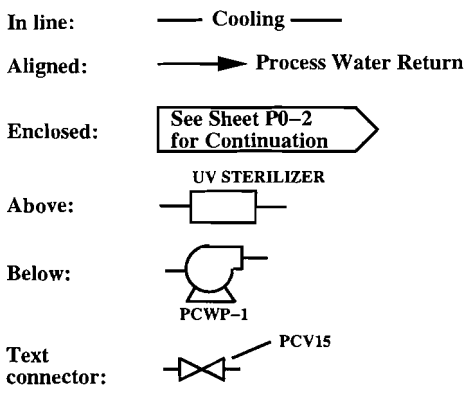

Figure 8: Texts and Graphic symbols. Text can be associated with graphic symbols in various ways as shown above. Annotations allow higher level systems to interpret functionalities of P&ID components and systems and also enable the linking of multiple sheets into a logically unified drawing.

6 Results and An Illustrative Example

The following provides an example to illustrate the capabilities of the prototype system. The example drawing is shown in Figure 9 which represents a portion extracted from the sheet shown in Figure 1. Figure 10(a) shows the legend sheet showing the symbols in the existing database for this illustration. The results are shown in Figure 10(b), where the solid lines show all the identified symbols and the dashed lines show the symbols that are not yet identifiable.

As described in the previous section, a GUI has developed that can be used to allow the user to automatically construct the topological and geometric structure of a symbol or component. First, the user can simply put a bounding box around an unidentifiable symbol, which is then displayed as shown in Figure 11. With the GUI as shown in the figure, the user can then add the necessary information about this new symbol. Once the user defined information is completed, the new symbol can be added to the database; Figure 12(a) shows the legend of the updated database, which includes the new symbol. Also, the system can point out where the new symbol fits in the component hierarchy, simplifying the effort of the user.

Finally, the user can now apply the new component database to the drawing. The results are displayed as shown in Figure 12(b). Note that all the instances of the new symbol are identified using the new database. This illustrative example clearly shows that the "learning" mechanism can be established through the semi-automated symbol construction process described earlier.

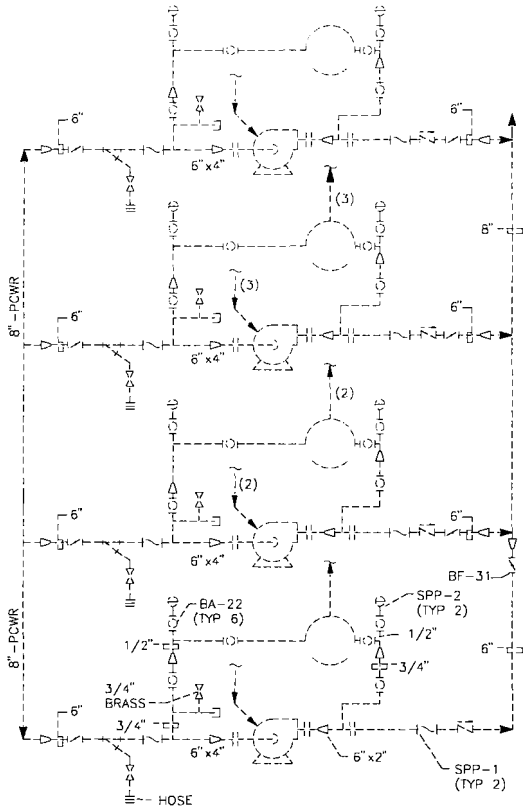

Figure 9: A portion extracted from the sheet shown in Figure 1. All symbols are shown in dashed lines to indicate that they are CAD vectors to be interpreted.

7 Functional Interpretation and Applications

The output from our graphic interpretation phase is a list of all recognized symbols and their immediate connectivity. We use this information to generate a single symbolic model of the process plant, as interpreted from the input drawings.

7.1 STEP Format

We use the emerging STEP standard, Application Protocol #227[5], as a language for our output model. Note that AP227 is a geometry standard, not the P&ID standard AP221. (AP 221 is yet to be adopted as ISO standard. Our results can be immediately translated into AP221 format when it becomes available.) Other output formats (JSPACE) have been implemented to connect the intelligent model to analysis and design tools. It has required modest effort to implement outputs to these standards. The intelligent diagram model facilitates translation into various standard outputs. In a similar way, electrical diagrams can be translated into appropriate IETM or other standard formats. Thus, our interpretation system can interface with many end-user tools, since STEP is being adopted worldwide as a single standard for the electronic exchange of

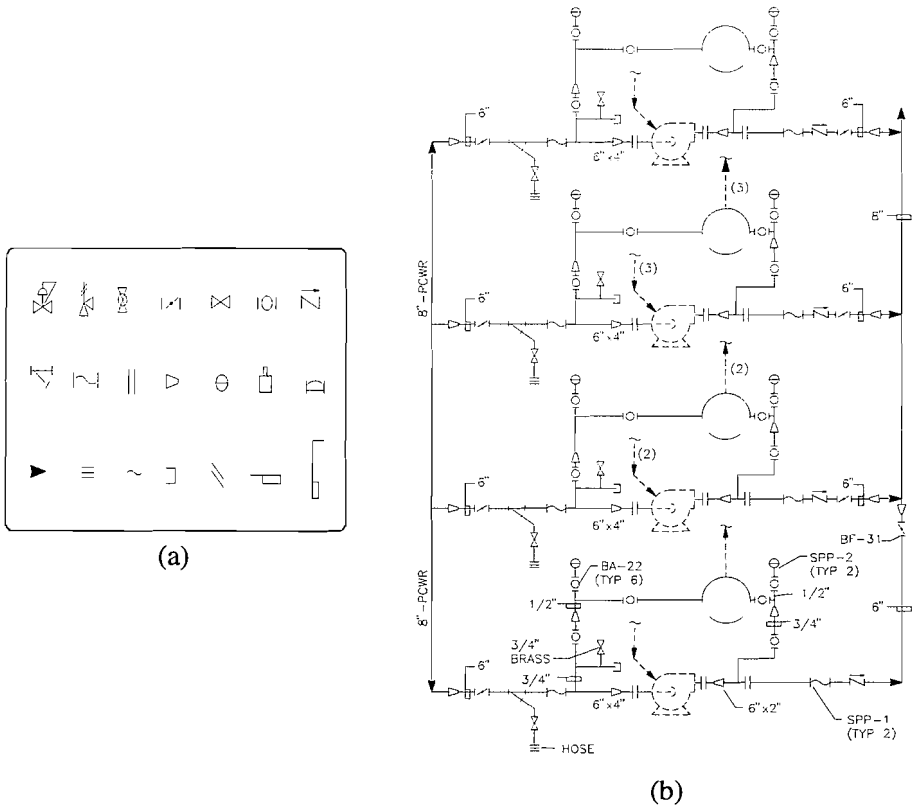

Figure 10: (a) All the symbols in the database are shown. A database can be created from legend sheets by the interaction of users with a GUI. (b) All the identified symbols are shown in solid lines. The four pumps are not identified and are shown in dashed lines.

product and process information.

7.2 Applied Engineering Knowledge

While our graphic interpretation engine can identify plant component symbols and their connectivity from the input drawing, it does not perform any interpretation of the input P&ID using engineering knowledge. To detect drawing errors such as missing components, inconsistent attribute annotations, and impossible system configurations, we export our interpreted model into the PlantSpace system developed by Jacobus, Inc. [6].

This system provides us with a CAD-based interface for interactive connection of multiple P&IDs, beneath which the system maintains the required links between the various symbolic models interpreted for each sheet. This requires that we use the proprietary PlantSpace model format in addition to the open STEP standard. As STEP has yet to be assimilated into the major plant companies information systems,

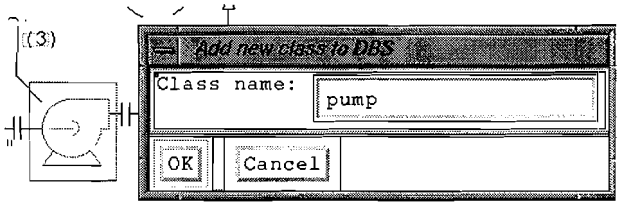

Figure 11: With the GUI, the user can put a bounding box around an unidentifiable symbol (the pump symbol in this case), type in necessary information about this new symbol, and then add it to the database.

the popularity of the PlantSpace system ensures that we have a platform for direct transfer of interpreted models into state of the art industrial software environments.

In addition to providing a means of linking multiple P&ID sheets, the PlantSpace system has an inference engine with which we can implement various rules to detect drawing errors. We have already detected two inconsistencies in a drawing. Figure 14 shows an example of a P&ID that we have encountered in an industrial drawing, in which there are annotation inconsistencies for both the pipe diameter attribute of the particular pipe line, and for the flow direction indicators. High level engineering knowledge about the system is needed to detect such logical errors in the input drawing.

8 Conclusion and Future Work

In earlier work, we demonstrated the feasibility of computer interpretation of P&IDs[4]. We have since improved the interpretation process by developing a more generic and robust methodology, which achieves a high success rate of symbol recognition for a single, typical industrial P&ID. In addition, a GUI was used to make easy constructing a component database and dealing with unidentifiable components. The output of the graphic interpretation was then used to generate a symbolic model for the process plant.

The effect of uncertainty in measurements and ambiguity in notation must be taken into account by a coherent decision process. In dealing with scanned vectors from paper diagrams, errors are appreciable and enter into the decision process. In annotation, functional evidence, and circuit level interpretation, multiple ambiguous evidence must be integrated. Traditionally, Bayesian inference has been the well-founded basis for decision under uncertainty. There have been a series of theoretical papers showing that Bayesian probability is a unique, well-founded calculus for decision.

The ACV and the object hierarchy are a network of topological, geometric, and functional relations (physical and engineering). Those relations combined with uncertainty in measurement and ambiguity in reference form a network of uncertain relations or constraints. In the SUCCESSOR system[3] at Stanford, we have translated such relations automatically into Bayesian networks solved by standard solvers. In general, constraints on the ACV translate one-one into Bayes nets. We have models for the

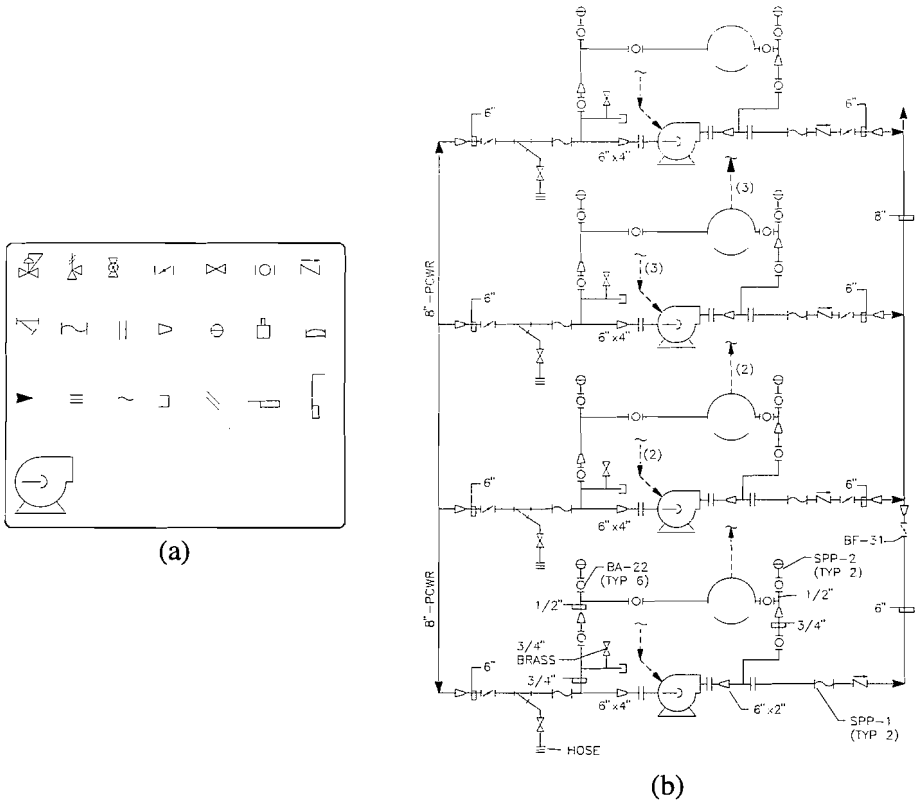

Figure 12: (a) After the user defined information is completed, the pump symbol is added to the database. (b) All the instances of the new symbol (the four pumps in this case) are identified using the new database.

uncertainties in the ACV induced by measurement uncertainties. In the course of the research, we will formulate a core of physical and engineering models that facilitate interpretation of annotation and text and that supply implied information. Thus, there is an automatic mechanism for globally consistent comprehension.

Acknowledgments

We wish to thank Intel and Bechtel Corporations for providing us with several P&IDs. We also gratefully acknowledge the support of Stanford University's Center for Integrated Facilities Engineering which funded this research.

References

[1] AutoCAD Release 12 (1993), Advanced Tools Manual, Chapter 6, Autodesk Inc.

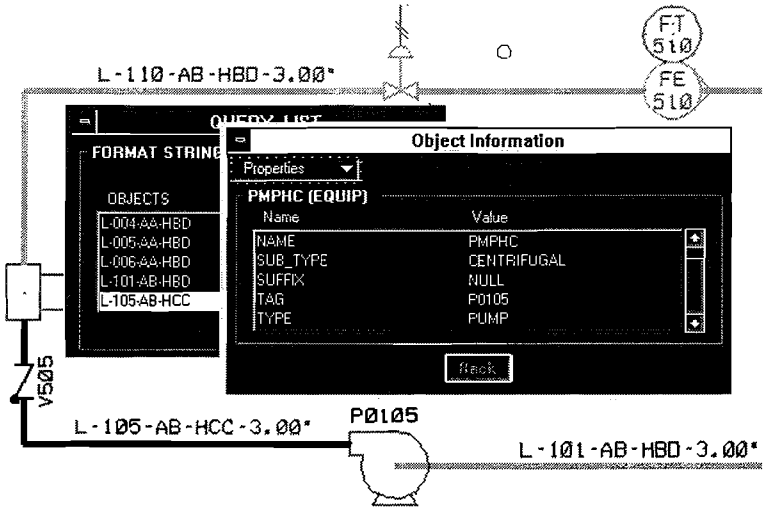

Figure 13: The CAD-based interface of the PlantSpace system is shown. To utilize the applied engineering knowledge, our interpreted model is fed into the PlantSpace system.

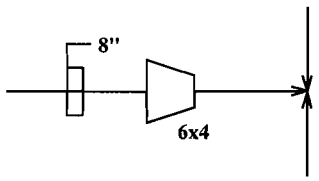

Figure 14: High level engineering knowledge to detect logical errors in the input drawing.

[2] Cormen T.H., Leiserson C.E., Rivest R.L., "Introduction to Algorithms", MIT Press, McGraw-Hill, 1990

[3] Binford T., Levitt T., Mann W., "Bayesian Inference in Model-Based Machine Vision", Uncertainty in Artificial Intelligence, Elsevier Science Publishers B. V., 1989

[4] Howie C., Kunz J.C., Binford T., Chen T. and Law K.H., "Computer Interpretation of Process and Instrumentation Drawings", Developments in Computer Aided Design and Modeling for Civil Engineering, 13-20, CIVIL-COMP PRESS 1995, Edinburgh, UK

[5] Exchange of Spatial Configuration Information of Process Plants, ISO 10303

[6] JSpace Technology Overview, Jacobus Techonology Inc.

[7] The International Society for Measurement and Control, ANSI/ISA-S5.1-1984 (R 1992)