Feature Center: Getting the Picture from Documents and Drawings

Robert Thibadeau, Rick Romero, and David Touretzky
July, 1995
CMU-RI-TR-95-32

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

ABSTRACT

The problem of technical document conversion embodies the broadest range of image processing problems, since technical documents, by their nature, span a highly diverse set of printed forms of communication. Our experience suggests that a proper solution path is to shift focus away from particular algorithms and toward a systemic architecture that permits readily addressing specific conversion problems with manual guidance. The result is a system in which the user can establish the goals he has for conversion, based on his own requirements and the attributes of the conversion tools he has at hand.

Keywords: document image conversion, non text document conversion, system architecture

Feature Center:
Getting the Picture from Documents and Drawings

Robert Thibadeau, Rick Romero, and David Touretzky

Imaging Systems Laboratory
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania

1 INTRODUCTION

This paper describes an automated document conversion system that explores the interactive domain of document image conversion. Our laboratory, with close corporate and government support since 1982, has had diverse experience in implementing complete automated document conversion systems in commercial environments (see [1], [2], [3]). We have constructed systems for optical character recognition, handwritten character conversion, forms recognition, printed wiring phototool conversion, mechanical drawing conversion, electrical wiring drawing conversion, architectural drawing conversion, oil-well log conversion, diagram and graph drawing conversion. There are several reasons why we have become interested in interactivity for automated document conversion.

One characteristic that appears to be universal to real-world systems is what we call "clumping." A system that may prove cost effective for one clump of documents may fail to prove cost effective on another clump. This coarse deficiency is not a property of a swing from one predefined type of document to another but appears to be a property of a much more subtle, and less controllable, shifting of the context in which the document clumps originate. For example, an executive may want newspaper articles scanned and converted one month, and, another month, he may want technical books scanned and converted. As another example, in the case of mechanical drawings, accepted drawing practices are different year to year, manufacturing site to manufacturing site, design bureau to design bureau. Even for the exceptions, such as drawings that represent tooling, such as printed wiring board artwork or mylar templates, there are subtle clumps that are just as damaging to automated conversion -- in annotation, metrological assumptions, shape interpretation, and revisioning policies. In a common instance, printed wiring tooling is assumed, by the workers in the printed wiring board shop, to have defects that they will manually fix during manufacture. The tooling, until the last minutes before its use, is not, actually, authoritative. The consequence of all this clumping is that fully automated conversion of documents has been, in general, not achievable.

Document conversion systems therefore provide a means for manually editing a solution when the automatic system, for one contextual reason or another, fails. There has been a practical problem here as well. Existing systems will provide a means for editing, but the editing can become tedious. It is not uncommon to find service bureaus, in-house or independent, going out of business because the work of manually converting documents is such poor quality work!

One solution to these problems is to provide a programming language interface to the document conversion package, so that, when a poor performance clump is encountered, special code can be written to improve performance to acceptable limits. But it happens that very few people are trained well enough in the intricacies of document pattern recognition to make this work. Many packages allow the selection of options, usually with check boxes or the designation of "interest areas" that permit some customization to a job. Our experience is that "check box options" and "full programming languages" go too far to either extreme. The person that is employing the conversion package either quickly bores of it, and the menial labor it requires, or finds it too difficult to program successfully.

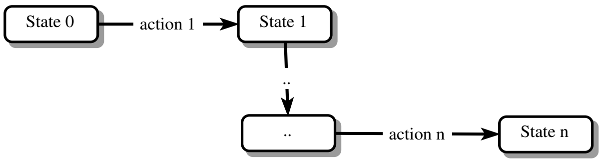
Feature Center was designed to provide a framework for easily customizing any particular system to the next "clump" of documents requiring conversion. It is not a replacement for tailored systems for broad areas of application (e.g., printed wiring board conversion versus office document conversion), but is, rather, designed to provide a single methodology that can be applied in most of these application contexts. It is not a replacement for writing code -- it sacrifices generality for simplicity and transparency. It's purpose is to provide a meaningful level of control to the naive user. The experiment is to define a tool that a personal computer user will find interesting to use while significantly expanding the scope of application of a given conversion package.

A final observation that we have made is that the user usually has some ideas about why the pattern recognition failed. He readily develops a naive understanding of the document conversion system's characteristics and problems. We have found Feature Center interesting because it requires us to put a rigorous framework on user views of document conversion. This mapping helps us to view pattern recognition in a different light -- showing areas where, perhaps, new research is "naturally" invited and can be understood for its importance in practical application.

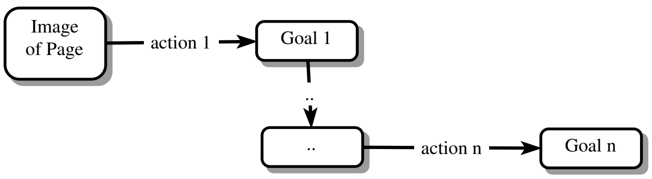
2. Definition of Feature Center

The term "Feature" derives from the idea of a feature of a page that must be converted from a pictorial description of the page, the image, into a symbolic description of the page. While Feature Center incorporates constructs for dealing with multipage documents, a page is assumed to be the basic level of discourse, and it is assumed that the page can be marked up, or annotated, symbolically.

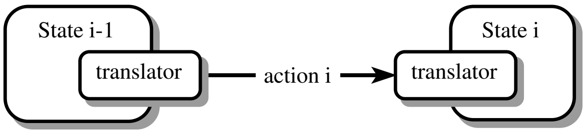
The underlying process model is intended to provide for many different "document conversion engines." We assume that processing can be characterized as a partially ordered sequence of states with state transitions determined by actions,



Furthermore the initial state is always the image feature for the whole page, and all other states are characterized as goal states, so,

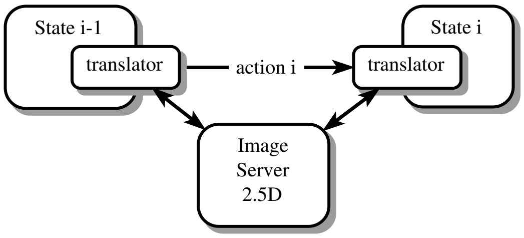


Actions take as input the prior states in the sequence and a proscription for the desired (next) goal state. Feature center provides for "foreign actions." These are programs or objects that may require a translator and executive. So, uniformly, this is given as:



Feature center is a representation which is a completely uniform two and one half dimension representation for document images. The image data and symbolic data are represented in a form that support a display mechanism for multiple transparent, registered, overlays or graphic layers. One overlay is assigned to each Feature instance.

This implies that there is a image and symbolic data representation and storage mechanism. We have called this an 'image server' because it is patterned after similar repositories in document image management systems. The representation of the Image Server data is compatible with a tabular relational database. So, for example, there is a relational entry for a feature instance with its overlay or layer number, its location in page space, and its symbolic typing information. It is of interest that the 2.5D representation is manipulated by the input and the output translators,



This implies, rightly, that the goal states are given by static representations that point to data in the image server. Furthermore, the executive and the input/output translators act to set Goal state values. A Feature Center script is, then, a description of the proscriptive state contents (the Goals) along with the named actions that achieve these goals.

An action can be as complicated as an entire OCR program (e.g., Caere OCR) or as simple as a single classifier (e.g., "is the image of the letter X?"). *Furthermore, a particular action may take as input more than one Feature instance.* The user interaction is seen as the user's assertion of one or more page conversion goals, followed by the system's suggestions as to how to achieve those goals. This is then followed by the user's selection among those suggestions, and the test of that selection.

Feature Center is constructed to permit the user to establish goals for the conversion of a page. The user proscribes what he wants from the conversion process. *A goal always constitutes exactly one feature instance,* F_i, except that several goals, and therefore features, may be developed sequentially. Partial ordering is allowed, but not to the extent that a single sequence cannot be proscriptively specified.

The conversion process is then seen as a conversion from one type of page feature to another by a collection of actions. Generally, any named action can be rewritten with an alternative (e.g., font recognition method A versus method B), as in,

$a_i \dashrightarrow a_j$

or as an expansion, as in,

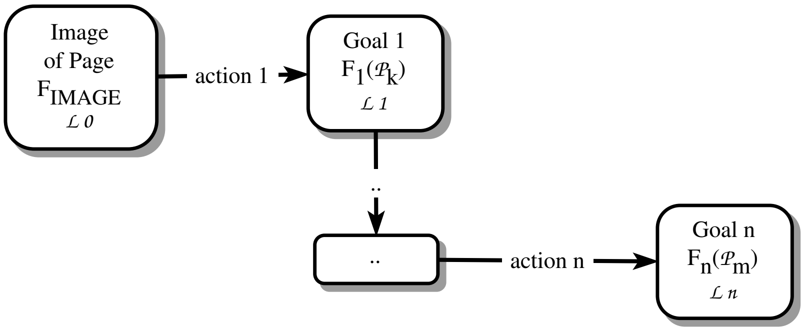
$a_i \dashrightarrow a_j\ Fj\ a_k$

In Feature Center, the features of a page are always given with a finite set of propositional annotations, *P*. Formally, any annotation, p_i, is a Proposition that has truth value with respect to a feature of the page (e.g., "This image is the letter X") . A feature instance, $F_{i,}$ is defined by a set of such propositions, as in F(P={p_1 .. p_n}) (e.g., "This image is the letter X.", "This image font is Helvetica"..). Each proposition, p_i, for a Goal Feature is proscriptive. It may be proscriptively True, False, or Not Decided.

The evaluation of a feature instance signifies that a proscriptive goal has been achieved by an action. Such evaluation provides a set of location instances, *L,* on the page for which the feature instance holds. So, F(A) => *L*. For all location instance, l_i, in *L*, *at least one* proscriptively True *or* Not Decided proposition p_i is True by reference, and *none* of the proscriptively False propositions p_j is true by reference. It follows, then, that for every location instance, l_i, the list of referentially True p_i annotate that location symbolically. So, for example, all the characters on a page may be proscribed to be Roman characters and Helvetica, not Times Roman. But the Ohlfs font may be Not Decided. It is now possible in the resulting set of location instances *L* to have a location instance on the page that is the letter X in Helvetica, and another location that is the letter X in Ohlfs. But it is not possible to have the letter X in Times Roman, since Times Roman was a prohibited annotation.

All location instances are localized on the page in bounding rectangles. This is the case no matter what the type of feature, whether it be an image feature of the page, a text feature, a line feature, or any other. The image server must be capable of rapidly accessing, updating, and manipulating such location instances. An algorithm developed for this purpose is described in [2].

The subclasses of the class Feature are distinguished by the specific annotations, P_m, available to the subclass. In the current implementation there are fourteen: IMAGE, TEXT, LINE, SYMBOL, FORM, TEXTURE, RULER, DISSECT, DIALOG, DETAIL, LEARNED, FORMAT, and LINK. The notation, F(P_{TEXT}) is equivalent to F_{TEXT}. The TEXT Feature is distinguished by the annotations, both proscriptive in F and descriptive in *L*, possible for a 'text' feature. Generally, then, document conversion in Feature Center is given as,



So, for example, a page IMAGE may be decomposed into a number of TEXT blocks, each annotated as to any of a finite number of types. As another example, a graphics image may be decomposed into a number of LINEs, including straight and dotted lines.

Any instance of the class Feature has a location on a page, a subclass designation, and an associated set of propositions that annotate the instance. Instances are collected together in only one way. All instances that share the same subclass and set of truth-proscribed propositions, but different locations on a page, are given by the Feature instance. The Feature instance is special in that all document conversion is defined as the derivation of one Feature instance from one or more other Feature instances.

3 The Feature Classes

This section briefly surveys the feature classes established for the general document image conversion of technical documents. Each class is described as to its underlying rationale and an example of the propositions that annotate an instance of the Feature class.

1. IMAGE Features have to do with arrangements of rectangular arrays of pixel data. The normal action associated with a transition from one IMAGE instance to another is segmentation or page decomposition. Other actions are image scaling, rotation, and binarization from color or grayscale. Among the predicates are TEXT_IMAGE, GRAPHICS_IMAGE, PICTURE_IMAGE, TEXT_LINE_IMAGE, CHARACTER_IMAGE, DESKEWED_IMAGE, DESPECKLED_IMAGE, INVERTED_IMAGE and BLACK_AND_WHITE_IMAGE. The unary predicate, SCALED(percent) is also an option.

2. TEXT Features have to do with arrangements of ASCII or Unicode symbols. An IMAGE instance may be converted into a TEXT instance using optical character recognition. Among the predicates are PRINTED, HAND_WRITTEN, PLOTTED, TABLE_TEXT, FIGURE_TEXT, DIAGRAM_TEXT, SCHEMATIC_TEXT, INTEGER, REAL, CURRENCY, FONT, FONT_SIZE, FONT_STYLE, SPACING, ENGLISH_MEASURE, METRIC_MEASURE, WORD_GRAINED, LETTER_GRAINED. There is also a "dictionary..." constraint for the words that may be known to be in use on a page (True), known not to be in use (False), or not decided.

3. SYMBOL Features have to do with arrangements of named symbols that may be more arbitrarily named than ASCII or Unicode. The predicate REPEATED indicates that a symbol may appear once or several times on a page. Symbols are handled like words of text in the dictionary, there are "special_characters..." for scalable symbols, "stamps" for non-scalable symbols, "scribbles..." for noise symbols, "fitted..." for symbols that are found by iterative fitting operations on their drawing parameters.

4. LINE Features have to do with arrangements of lines. All lines have WIDTH(value in page space). Other predicates include STRAIGHT, DOTTED, CIRCLE, ARC, RECTANGLE, POLYLINE, POLYGON, ELLIPSE, SMOOTH_CURVES (Bezier), DOUBLED, BORDER, VERTICAL, HORIZONTAL, ANGLED, and ROUNDED_OFF.

5. FORM Features have to do with simple spatial arrangements of IMAGE, TEXT, SYMBOL, and LINE locational instances. A Figure caption may be defined as a FEATURE as a TEXT block that must contain the dictionary word "figure" under an IMAGE block of a GRAPHICAL_IMAGE.

6. TEXTURE Features have to do with textured regions. The two components of texture are a bounding polygon unless the texture is a background texture and the texture description. The predicates include SOLID_FILLED, UNIFORM_PATTERN, BACKGROUND_TEXTURE.

7. RULER Features have to do with rescaling locational results to an imposed number system. The new number system endures with the locational instances. A RULER Feature has the locational instances that are the raw (observed) grid points and the normalized grid points. The basic operation is projective histogramming in order to extract periodicity and offset (patent). Predicates include UNIFORM_GRID, VISIBLE_GRID, IMPLICIT_GRID, VERTICAL, HORIZONTAL, SCALE_TO(linear transform) and a dictionary of "grid_units...".

8. MOVED Features have to do with the movement or distortion of one kind of locational instance against a reference. The reference can be a RULER NORMALIZED grid, in which case there is a SNAP or BILINEAR_FIT to grid. Other movements, such as a distortional correction using a distorted RAW set of grid points as input and a SNAP or BILINEAR_FIT. In either case the MOVED Features take on the unit locational values of RULER.

9.DISSECTED Features have to do with the subtraction of one kind of locational instance from a feature instance, thereby creating a different feature instance. So, for example, one can dissect the text from a graphical drawing, leaving only the image of the drawing.

10. FORMAT Features have to do with formatting a feature instance for input into another system. Predicates include AUTOCAD_DXF, POSTSCRIPT, and WORD. Formatted output can ADD_TO_FILE(name).

11. DIALOG Features allow any property of a locational instance of any feature to be copied and then changed manually, including its location. The result is a new feature instance of the same type as the input locational instance. For example, TEXT becomes corrected TEXT. An IMAGE becomes a corrected IMAGE. A DIALOG can ADD_TO_FILE(name) the training set, and therefore it can be employed to build training data over a number of pages. In order to permit a locational instance to be 'deleted,' a locational instance can have a null location.

12. DETAIL Features provide a means of creating a feature instance that contains a pattern of low level predicates for each locational instance that has been provided to it. These include a vector of predicates for HORIZONTAL DENSITY PROJECTION and VERTICAL DENSITY PROJECTION, HORIZONTAL LENGTHS, VERTICAL LENGTHS, EDGE_CONTOURS, PIXEL_VALUES, PIXEL_DEPTHS, and NORMALIZED(limit).

13. LEARNED Features take as input the training set developed in one or more DIALOG Feature instances and an optional the feature instance, usually DETAIL, with an exactly corresponding locational instance set from the training set. The LEARNED Features will be like an automated DIALOG: The output will be a new feature instance of the same type as the input locational instance, but with annotations (including location, if necessary) changed to conform as closely as possible to the "corrected" feature instance.

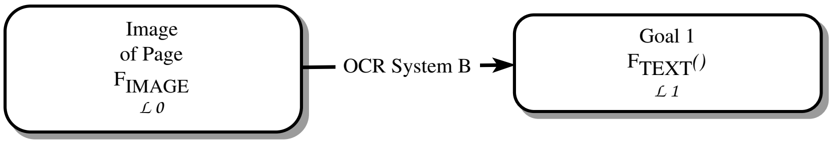
14. LINK Features provide 'continuity links' across pages. Each link has an anchor locational instance and a target possibly on a different page. LINKs exist for READING_ORDER, REFERENCE_LINK (e.g., "see page 20"), and HYPERLINK (e.g., "www.fcc.gov"). A READING_ORDER and REFERENCE_LINK link are directional to a locational instance target but can be traced backwards. A HYPERLINK may have a named target that is not a locational instance in the feature center representation of a another page.

4 Scenario Experiments

The purpose of scenario experimentation is to take different technical publication conversion scenarios and manually work them out based on pattern recognition tools that we have developed or obtained from commercial sources, or which have been described in literature. The scenario experiments further constrain system design, and also make the first steps in confirming the generality of the document conversion architecture for the purpose intended. This section will briefly present a few scenarios that consider a breadth of applications to graphical aspects of technical publications.

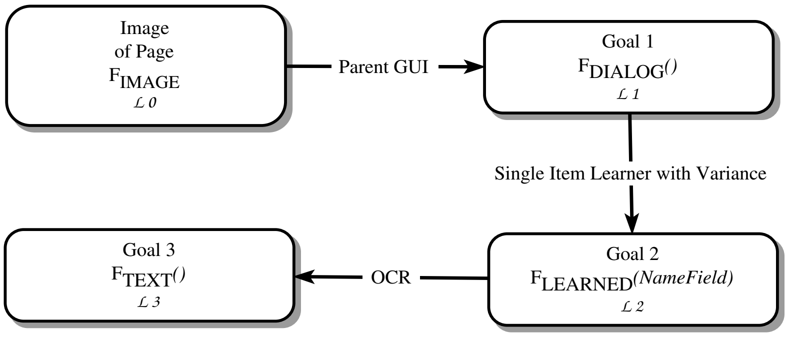
4.1 Different OCR subsystem.

For example, suppose the default system utilizes Caere/Calera OCR and the idea is to substitute Xerox Information Systems OCR (perhaps the document is in Russian). It is important to note that the goal is TEXT with every proscriptive attribute "undecided" since the intent is to just let a different system "have at" the image. Also, the list of text objects, $\mathcal{L}1$, is assumed to be available to the internal blackboard and therefore does not require a formatted write in a foreign format.

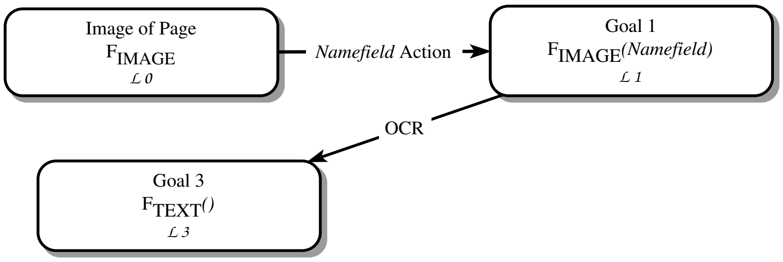


4.2 Select area for OCR

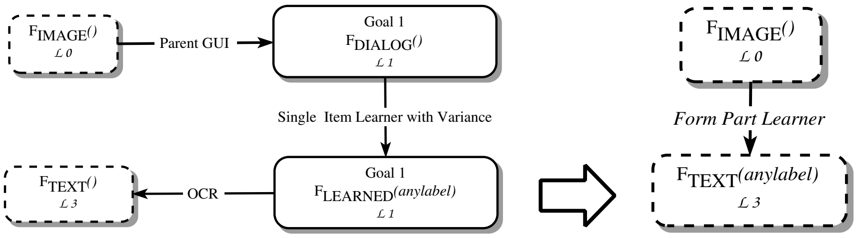
Another common desire is to identify a region for applying OCR. This is particularly common in the task of scanning many forms of the same type with an accurate scanner. The explicit definition, assuming the user wants a new image type, called the "namefield" of the form, would be:



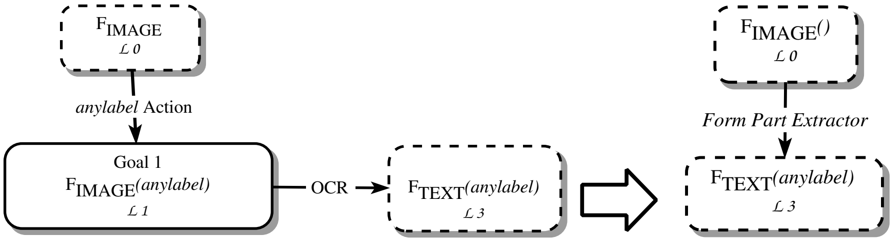
On the first image of a form we assume that a single exemplar learning routine is available that applies a standard notion of scanner precision variance. (There are common algorithms for this). Then on every other form the following:



But because actions can represent complex state transitions, the Feature Center interface might also provide a shortcut to 'learning' parts of an image (the dotted boxes imply a variable state):

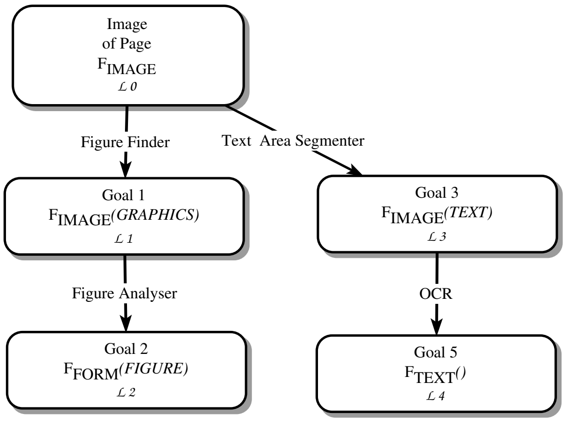


Alternatively, a single script may be assigned to an action by the user, with the same result. Furthermore, the user or system may assign a special script for the actions learned:

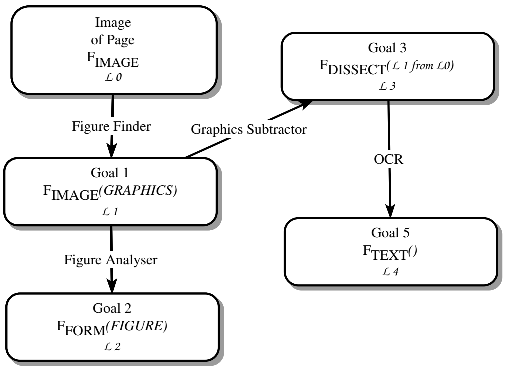


4.3 Extract Graphics for Separate Processing

It is common for OCR packages to do the "wrong thing" with figures. They will output a few bits of text from the figure while also outputting a ’picture region’ (or worse, several of them). A figure finder and a figure analyser are relatively straightforward. As are text area segmenters. This implies an option to restructure the processing for OCR as:



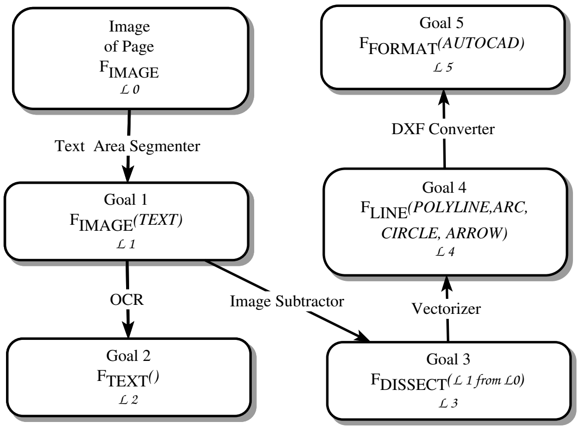
But, suppose, the intention is to extract the figure from the page and then apply OCR to everything else because the OCR package is good with Tables and with segmenting out Pictures and other non-text, and non-figure, areas. Then Feature Center’s script would involve a dissection of the figure from the image:



Now suppose the Figure Finder did not work as well as hoped. Perhaps the Figure Analyser works only on a subset of Figures. It should be clear that a DIALOG utilized to create a training set of correctly and incorrectly identified figures could be inserted. Then DETAIL extracted from the GRAPHICS IMAGEs could be LEARNED employing a discriminant analysis or nearest neighbor in an effort to further filter the IMAGEs submitted to the figure analyser.

4.4 Vectorization of Mechanical Drawings in Technical Articles

As a final example we consider applying existing vectorization technology. It is known that at least one class of mechanical drawings is best treated as lines and text in the following scheme. Assume the vectorization action is any one of at least a dozen commercially available packages.



5 Conclusions

The present architecture for non-text conversion of technical articles continues to improve as we consider other desirable conversion scenarios. It is the compendium of such scenarios and the ease by which people can construct others to their own desires that would validate this approach to image document understanding.

REFERENCES

[1] R. Romero, R. Berger, R. Thibadeau and D. Touretzky, "Neural Network Classifiers for Optical Chinese Character Recognition," In *Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval*, University of Las Vegas, pp. 385-98, 1995.

[2] R. Romero and R. Thibadeau, "Object Manipulation for Document Conversion," In *Proceedings of the IEEE 1995 International Conference on Image Processing*, October 23-26 , 1995.

[3] R. Thibadeau and D. McNulty, "Two Systems For Converting Raster Data To Numerical Control Data" In *Proceedings of 1990 IAPR Workshop on Machine Vision Applications,* November 28-30, 1990.

ACKNOWLEDGEMENTS

This work was sponsored in part by the Office of Research and Development and Visus Technologies, Inc. This paper was also published in the *Proceedings of the International Workshop on Graphics Recognition,* International Association for Pattern Recognition, State College, PA , July 1995.