MVA'90 IAPR Workshop on Machine Vision Applications Nov. 28-30,1990, Tokyo

Analysis of Dimension in Mechanical Engineering Drawings

Suzanne Collin
Dominique Colnet
INRIA Lorraine – CRIN
Campus scientifique
B.P. 239
54506 Vandœuvre-les-Nancy CEDEX
France
Tel. +33 83.91.21.25
Email: {collin}@loria.crin.fr
{colnet}@loria.crin.fr

Abstract

Many paper engineering drawings are still in use and there is a demand for reliable means of transforming them into digital information and incorporating them into a CAD (Computer Aided Design) system. To be efficiently used, paper engineering drawings dimensions must be analysed allowing for example 3D model interpretation.

We describe the two majors parts of our system: vectorization and dimension identification. The first part, vectorization, uses a specific algorithm to recognize as soon as possible, text and arrows. This is a typical case where *a priori knowledge* can be introduced as well in low-level processing as at higher-level. The second part is achieved in the same way, finding as soon as possible some acceptable dimension. Two differents actors are introduce two implement the dimension identification: the "Grouper", which groups elements, trying to build dimensions, and the "Checker" which controls the validity of the set of grouped elements. These two actors are working alternately and share a common view of the drawing.

The 1^{st} low-level part, is already implemented in C. The 2^{nd}, the high-level one is implemented in Objective-C, which seems powerful for handling the complexity of the system. We conclude with remainings problems and furthers works.

1 Vectorization and Symbols Extraction.

The vectorization is the start point of the analysis system, and the whole interpretation depends of the precision of the extracted symbols. The vectorization must be as complete as possible. The basic vectorization is processed by the REDRAW system [1], which provide a set of vectors and a connected component tree including the relations between them. The advantage of this structure is that immediate access to topological relations and to pixels is always possible [11,3] Components are also defined with their contours.

However, this vectorization is not sufficient to allow the analysis of the dimensions. We must add more specific processing stage for taking into account the characteristic entities of the dimensions, which are *arrows*, *text* and *thin lines*. A good recognition of these elements are essential for the detection of the entire dimension.

Thin lines can be easily separated from the others by consideration of thickness, computed in the vectorization step, but arrows and text can be seen as composite symbols, and need a specific process [1]. *Text* is extracted from graphic by analysing the connected component structure, area and density [8], and further, characters are grouped into strings. This method do not require recognition of individual character, and work in any direction, even slanted one.

Often, *arrows* are very small symbols and their recognition is essential to go on the analysis. To avoid big and irreversible distortions, their detection must take place *before* vectorization, just after polygonal approximation. A lot of methods are known to solve this classical pattern matching problem: *subgraph isomorphism* [9,12], *Hough transform*, or looking for main domain features, in this case a triangular form. This is the method we have chosen [1].

The result is a set of possible arrows. Some of them are not arrow, but this is not a serious problem, because the next stage will detect them as bad ones. Figure 1 shows the results of text/graphic separation and arrows detection.

2 Dimension Analysis and Drawing Association.

Localization and recognition of dimensions is an essential contribution to a complete understanding of the whole drawing. We use two kinds of knowledge representation: knowledge represented by *rules* and knowledge represented by *grammar*.

Different elements can be a part of a dimension: text, arrow, shape, witness, tail and contour. Figure 2 shows these different parts.

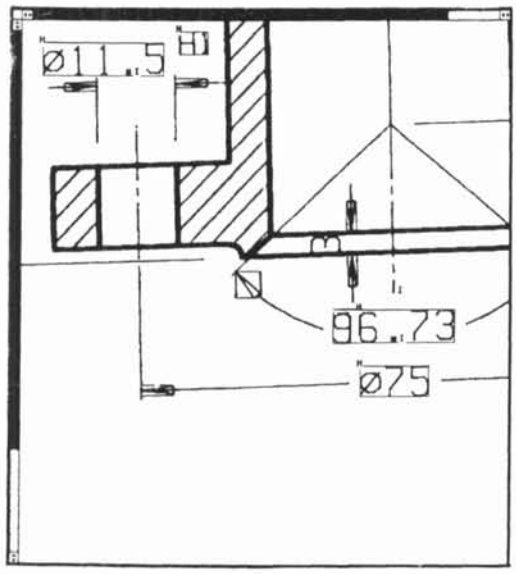

Figure 1: Arrows detection and Text/Graphic separation. Recognized arrows and texts are boxed on this figure.

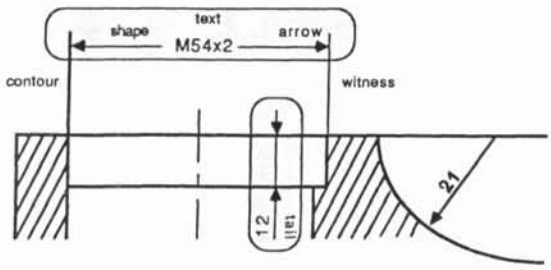

Figure 2: Different parts form a dimension: text, arrow, shape, witness, tail and contour.

The graphical representation used in technical documents are most of time very strongly structured, following a set of well known rules. Our idea is to use as much as possible these rules to lead the interpretation. Dori [6,7] has studied recognition of dimensions; his method consists of a *syntactical analysis* of the dimensions. A grammar describes all possible dimension types and there is a subgrammar for text recognition. The grammar's rules enable the recognition of any dimension in a mechanical drawing. The checking part of our work is based on his method.

On the other side, we use domain knowledge to *build gradually* dimensions. Our global strategy is first, to find ordinary dimensions, and later, to find more complex ones. It is important to provide as quickly as possible high-level entities for allowing to be more selective in the next choices. To achieve this strategy, we have introduced two "actors": the **Grouper** which try to build dimensions, and the **Checker** which controls the validity of the grouped elements.

A permanent dialogue is set up between the two actors, until a final solution is founded. The two actors are working alternately, and share a common view of the drawing. As the work is going on, their behaviour is changing with respect to previous success or fails in the grouping or recognition steps. They become more and more selective as the system is going on. Backtracking to previous data, pixels data for example, is even possible. Next two subsections give details about the Grouper and the Checker idea.

2.1 The Grouper

The Grouper handles essentialy domain knowledge rules, and dimension building algorithms. He has to suggest something supposed to be a dimension, applying knowledge rules. He takes into account the previous success or fails to suggest a new dimension. For example, text already included in a recognized dimension cannot be used once more. Previous suggestions which have lead to fail are shut out.

The Grouper must also give response to Checker requests. For instance, a request can be: looking for segment that is in contact with the arrow head. The research is provided by applying knowledge rules, and the Grouper computes only one group at each time, waiting for the Checker response. In the same time, the Grouper keeps in order a failure list (which contains the trouble of bad dimensions), and knows exactly the last primitive he has proposed, the methods that have been tested and the next dimension proposition to do.

2.2 The Checker

According to the grammar, the Checker analyses the Grouper's proposition. There are three kinds of possible responses:

- the candidate dimension is fully recognized by the grammar,
- a request to the Grouper is necessary to complete the dimension,
- the proposed dimension leads at any state in the grammar.

There are specific actions for each case: updating results when the dimension is recognized, searching for drawing to be associate to the dimension, or, in the last case, failure memorization.

Figure 3 gives a global view of the dialogue between the two actors.

3 Implementation, Results and Futur Works.

The usual language for image processing development is the C language. Vectorization and symbols extraction is implemented in C and provides all the primitives: vectors, arrows and text detection.

Dimension analysis and drawing association is currently under implementation in Objective-C [5]. It is easier to separate all the algorithms of the different steps, and to test quickly a lot of different heuristics with local changes [4].

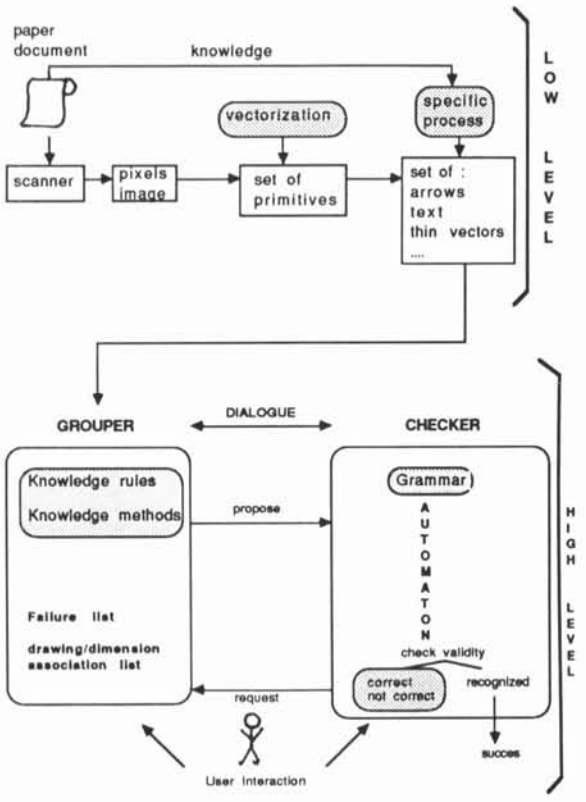

Figure 3: Dialogue between the two actors. A global view.

The *Object-Oriented* approach [10] allows data abstraction and code sharing. Obvious objects such as segment, vector, arrow, text... are described by different classes, without worry about the physical representation. Algorithms and test steps are easy to set up class by class. This approach also allows to describe separately dimensions categories even when characteristics are to be shared. Moreover, if this cutting is a good copy of the world, the recognition algorithms will be quasi similar to human raisoning.

During the development, as soon as a class become too complex, that is the methods it knows and the instance variables become too important, another class is often created to deal each other with an easier problem. Actually, three major classes implement the dialogue between the Grouper and the Checker: one for the Grouper, one for the Checker and the last one for the Drawing. The Grouper class contains the methods which build gradually dimensions according previous results (success or fails). The Checker class knows all methods that are in relation with the grammar and the automaton. The Drawing class deals with drawing information: where are arrows, is there corresponding text, what about actually recognized dimension... The bitmap instance variable is also known by the Drawing class.

For the time, the dialogue affects the Grouper and the Checker, and there is possibility to go back to the image pixels in case of ambiguity. It is therefore reasonable to add *user interaction* at the analysis process in order to correct errors and solve ambiguities [2]. This part will be integrated in the system and, to increase interaction between low-level and hight--level, the low-level part will be implemented too in Objective-C.

References

[1] D. Antoine, S. Collin, and K. Tombre. Analysis of Technical Documents: The REDRAW System. In *Pre-proceedings of IAPR Workshop on Syntactic and Structural Pattern Recognition*, pages 1–20, Murray Hill, NJ, 1990.

[2] O. Bergengruen. Generating and Manipulating Solid Models from Images. In *Proceedings of 6th Scandinavian Conference on Image Analysis*, pages 660–667, Oulu, Finland, 1989.

[3] S. Collin and K. Tombre. Image Segmentation Using Connected Components for Document Analysis. In *Proceedings of IFACS/IMACS/IFORS International Symposium on Advanced Information Processing in Automatic Control*, pages 166–170, Nancy, France, 1989.

[4] D. Colnet. *Prototypage, programmation objet : application à l'édition structurelle.* Thèse de l'Université de Nancy I, 1989.

[5] B.J. Cox. *Object-Oriented Programming.* Addison-Wesley, Reading, Massachusetts, 1986.

[6] D. Dori. A Syntactic/Geometric Approach to Recognition of Dimensions in Engineering Drawings. *Computer Vision, Graphics and Image Processing*, 47:271–291, 1989.

[7] D. Dori. Self Structural Syntax Directed Pattern Recognition of Dimensioning Components in Engineering Drawings. In *Pre-proceedings of IAPR Workshop on Syntactic and Structural Pattern Recognition*, pages 88–112, Murray Hill, NJ, USA, 1990.

[8] R. Kasturi, C. Shih, and L.A. Fletcher. An Approach for Automatic Recognition of Graphics. In *Proceedings of 8th International Conference on Pattern Recognition*, pages 877–879, Paris, 1986.

[9] P. Kuner. Efficient Techniques to Solve the Subgraph Isomorphism Problem for Pattern Recognition in Line Images. In *Proceedings of 4th Scandinavian Conference on Image Analysis*, pages 333–340, Trondheim, Norway, 1985.

[10] G. Masini, A. Napoli, D. Colnet, D. Léonard, and K. Tombre. *Les langages à objets.* InterEditions, Paris, 1989.

[11] K. Tombre. *La saisie automatisée de documents composites : reconnaissance, codage et interprétation des parties graphiques.* Thèse de doctorat de l'INPL, Centre de Recherche en Informatique de Nancy, Vandœuvre-lès-Nancy, 1987.

[12] S. Ullman. Filling in the gaps: The Shape of Subjective Contours and a Model for their Generation. *Biol. Cybernet*, 25:1–6, 1976.