An AI-Aided System for the Conversion of Paper-Based Technical Drawings into a CAD Format

J. M. Gloger
Daimler Benz AG
Research Center Ulm
Institute of Inform. Technology
Wilhelm-Runge-Str. 11
D–7900 Ulm, Germany

B. Pasternak, R. Sprengel
G. Gabrielides
University of Hamburg
Department of Computer Science
Bodenstedtstr. 16
D–2000 Hamburg 50, Germany

N. Luth
M. Timmermann
Institute for Production Systems
and Design Technology
Pascalstr. 8–9
D–1000 Berlin 10, Germany

Abstract

The paper presents a system, which enables the conversion of paper-based technical drawings into a format suitable for CAD systems. The system is based on a commercial available *scanning and vectorization* unit. Since the vectorization results are far from being perfect, a *postprocessing* component to remedy some of the weak points have been created. The improved vectorization results permit a *knowledge-based interpretation* of the drawing content along with error detection and drawing redesign.

1 Introduction

Paper-based technical drawings constitute a great potential of know-how in many companies. Nowadays, the process of construction and documentation of a product is based on CAD systems and is no longer paper-based. A gap exists between the old (paper-based) and the new computer-based information. Many paper-based drawings could be reused, if that information were accessible to the system. Providing this information both to documentation and CAD systems would guarantee *consistency* between the data used for *documentation* and *construction*. Converting drawing information on paper into a computer readable form can be achieved already by commercial vectorization systems. The resulting graphical primitives (polylines, circular arcs and circles, etc.), however, are not satisfying and additional postprocessing is necessary when knowledge-based interpretation should run successfully.

2 Vector data postprocessing

Recognition of graphical primitives by the underlying commercial system is not sufficiently robust. These inadequate results lead to problems during further knowledge-based interpretation. Therefore, postprocessing aims to improve the recognition results relating to graphical primitives, to recognize composite graphical primitives and to prepare the vector data for further drawing interpretation (See Fig. 1 for a summary of the tasks). The steps of section 2.1 up to section 2.4 are described in more detail in [Glo92].

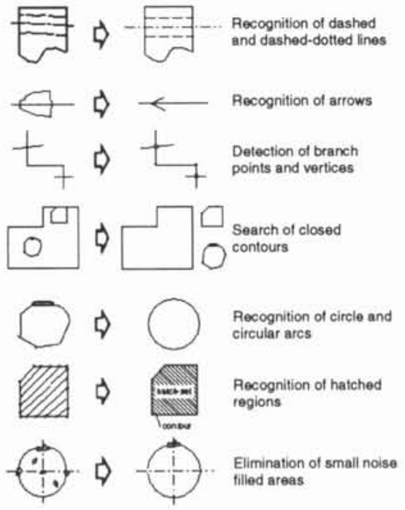
Recognition of dashed and dashed-dotted lines
Recognition of arrows
Detection of branch points and vertices
Search of closed contours
Recognition of circle and circular arcs
Recognition of hatched regions
hatch-set
contour
Elimination of small noise filled areas

Figure 1: Steps of postprocessing.

2.1 Detection of simple geometrical relations between lines

Some features of the line structures are needed for further drawing interpretation, e.g. *branch points*, *vertices* and *closed contours*. Branch points are important features of a drawing because they show certain relations between drawing elements. The problem of branch point detection can be reduced to the problem of determining the intersection point of two lines. Detection of vertices, as points of high curvature, and a corresponding decomposition of polylines is carried out. A chord is drawn between two points of a given polyline. The idea of the proposed method is to compare the orientation of all chords. A vertex occurs at the point where a local maximal divergence of the orientation can be detected. As a rule the closed contours in a drawing are transformed into a different set of lines and filled areas by the vectorization. Detection of closed contours within given clumps of different graphical primitives is crucial for:

- exact determination of the geometrical shape of elements (polygon, circle);
- recognition of actually present contours in a drawing (especially hatched regions);
- significant reduction in the amount of data.

The search for closed contours is based on the detection of the neighbourhood of elements. Neighbourhood is determined by the spatial proximity of element points. A closed contour can consist of different elements such as straight lines, arcs and/or filled areas. These elements can have various shapes and line thickness. For example, a closed contour for a hatched region can be a combination of thick straight lines and thin freehand lines.

2.2 Correction of the metrical description

The description of an *arrow head position* is often metrically inexact, e.g. the tip of an arrow head does not touch the subsidiary line or the orientation of an arrow is incorrect. These lacks have to be corrected.

A small filled area is a *noise element* which does not belong to a drawing but appears as a sequence of binarization and thinning of a raster image. Usually it is generated at locations in the raster image with extreme thickenings of lines (e.g. by intersection with another line) or changes of homogeneity. The size of a noise area is considered to be the size of the circumscribing rectangle.

2.3 Correction of the geometrical shape

In some cases, instead of a *circle* a set of circular arcs was detected or no circularity at all could be determined. There are two approaches for the detection of circles and circular arcs:

- recognition of a given set of circular arcs as a circle or a circular arc;
- recognition of a polygon or polyline as a circle or circular arc [Jan87].

2.4 Detection of graphical primitives

In the current state of postprocessing only the recognition of *arrows* is implemented. Arrows not recognized are transformed mostly into filled areas and thin lines. A recognition of arrows during postprocessing is possible if the shape of the filled area can be approximated by a triangle or a trapezium and a thin continuation line exists in close proximity to the wide side of the arrow head. The statistics about the sizes of detected arrows in the vector data is considered during the postprocessing recognition.

2.5 Recognition of composite graphical primitives

Performing this task includes the recognition of *dashed, dash-dotted* lines and *hatched regions*. Dashed lines are used to represent symmetrical axes, hidden lines or cut-planes. Our approach allows the processing of lines with various geometrical shapes. Moreover, the exact geometrical shape of recognized dashed and dash-dotted lines is determined. To reduce the computational costs a group of candidates is selected from the graphical primitives. These candidates represent the possible constituent elements of the dashed line. All subsequent steps are limited by the group of candidates selected. The group is broken down into appropriate subgroups. Every subgroup - if it has more than three candidates - corresponds to a dashed line or a segment of a dashed line. The process of creating a subgroup starts with the first element among the candidates. A continuation region for an element is defined. All remaining candidates are checked whether any of them intersects the continuation region. The additional classification of identified "dashed lines" can improve the results of the recognition process. Hatched regions - representing cut-planes - are used extensively in technical drawings. The set of n lines ($n \gg 2$) builds a hatch-set if the lines are parallel and equidistant. If the inner-part of a region is filled by a hatch-set, the region is called a hatched region. The hatch-set can be broken by inner holes. Recognition of hatched regions entails four main problems:

- detection of all parallel, equidistant and thin straight lines;
- the detection of all candidates for the hatch-set;
- location of the boundary of the hatched region (including the boundary of all inner holes);
- detection of holes in a hatched region.

3 The architecture of the interpretation system

The drawing interpretation system is based on a blackboard architecture [Eng88], [Gall88]. In blackboard systems the inference engine and knowledge base are organized in terms of specialized processing modules called *knowledge sources.* These knowledge sources operate like "small expert systems", which are responsible for the changes in a common working memory called ***blackboard***. Each knowledge source uses its own methods for drawing inferences from information on the blackboard and for inserting new information into the blackboard.

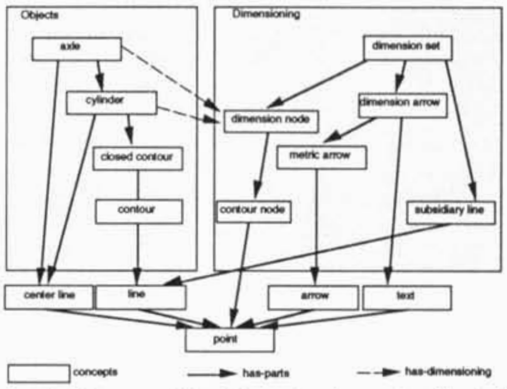

Figure 2: Decomposition hierarchy of generic objects from the domain of axles.

A change on the blackboard caused by inserting, modifying or deleting an object releases a signal to the knowledge sources called *event.* Events carry information about the type of the affected objects and the type of manipulation. Each knowledge source holds activation conditions that contain a list of events on which they react and a list of conditions that must be fulfilled by the object that causes the event. If a new event occurs, each knowledge source specialized for this event type is activated. If the activation conditions (e.g. geometrical constraints) are satisfied the appropriate entry in the agenda is made.

A scheduler determines which agenda entry is to be considered first according to its priority. At this point the action part of the knowledge source is executed. The action part usually contains instructions for inserting, modifying or deleting blackboard objects. The processing cycle is repeated until a knowledge source terminates the process or the agenda becomes empty.

4 Design of the interpretation system

The central task in building an interpretation system is embedding domain knowledge in the chosen architecture. The main kinds of domain knowledge are:

- Knowledge about *Objects* and their *Properties*
  After postprocessing, the drawing consists of basic objects (the graphical primitives). Knowledge about higher-level objects is formulated in terms of graphical primitives. The higher-level objects can be physical (like cylinder, axle, etc.) or non-physical (like metric-arrow, dimension-set, etc.).
- Knowledge about *Structural Relations*
  Structural relations between objects are essential for building higher-level objects from lower-level ones. Using the decomposition relation, a hierarchy of objects can be defined. Although the decomposition hierarchy is the central structural relation, additional structural relations may be required in specific domains. Figure 2 shows the decomposition hierarchy of concepts from the domain of axles.
- Knowledge about *Geometrical Constraints*
  The knowledge about structural relations determines which objects can be combined to higher-level objects. Typically, additional geometrical constraints have to be fulfilled if objects are parts of a higher-level object. Knowledge about geometrical constraints between blackboard objects is represented by knowledge sources.

5 The strategy of interpretation

The strategy of interpretation is described exemplarily by the domain of axles. Figure 3 shows the drawing of an axle that is interpreted by the system.

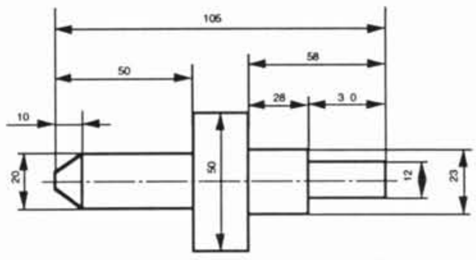

Figure 3: An engineering drawing of an axle.

While the graphical primitives are read, the corresponding primitive concepts of the knowledge sources are instantiated.

By the use of priority settings the interpretation is divided into two sequential paths. First the physical objects in the drawing are interpreted until the top-level concept axle is reached. Next, the dimensioning is interpreted up to the concept dimension-set.

5.1 Interpretation of physical objects

The steps of interpretation are (different knowledge sources are printed in *italics*):

- *ks-lc* tests the connectivity of the lines in the drawing. Lines that are open ended at one side are assumed to be subsidiary-lines and extended by *ks-sub* through crossings with arrows. Lines with both sides connected to other lines are candidates for contour lines.
- *ks-con* checks the proposed contour lines for building a closed contour loop.
- *ks-cyl* tests the closed contours for rotational symmetry with respect to a center line and instantiates the concept cylinder for each of these contours.
- *ks-subc* tests the lines of closed contours that do not build a cylinder for being part of a subsidiary line. Matching lines are added to the already built subsidiary lines.
- *ks-ax* combines adjacent cylinders to an axle and creates dimension-nodes for all axle and cylinder parts that need dimensioning information. The dimension nodes are connected to the objects via the domain specific HAS-DIMENSIONING relation, which is displayed as a dashed line in the hierarchy of figure 2.

5.2 Interpretation of dimensioning

In the second phase the dimensioning information in the drawing is interpreted.

- *ks-ma-i* and *ks-ma-o* combine arrows to the two types of metric-arrows, respectively.
- *ks-dim* builds the object's dimension-arrow from a metric arrow and a text string.
- *ks-ds* combines a dimension arrow with optional subsidiary lines and the dimension nodes built during physical object interpretation.

6 Results of the interpretation

In general, the results of geometric interpretation are a structural hierarchy of concept instantiations occurring in an engineering drawing. This hierarchy holds information about the role of the primitives in the context of the drawing and their classification to higher concepts, and is divided into two parts. The first part leads to a top concept with physical meaning (e.g. axle, plate), while the second part leads to a top concept that contains dimensioning information (i.e. dimension-set) related to the physical objects. The information contained by the hierarchical representation of the interpretation can be very useful for further processing. Our system enables dimension checking, redesign and a 3D visualization of the interpreted drawing.

6.1 Dimension checking

The system uses the concept of dimension-node in order to connect objects with dimensioning. Dimension-nodes specify the areas of the objects that need dimensioning information. Actually, they are sets of contour-nodes with coinciding projections on the two primary axes X and Y, respectively. Dimension-nodes have HAS- PARTS relations to dimension-sets and HAS-DIMENSIONING relations to all physical objects. In connection with a dimension-set, dimension-nodes indicate the two points attached at the ends of the dimension-set. The distance between these points is given by the dimension-set value. Regarding the dimension-sets describing the top-object as a graph with nodes representing dimension-nodes and arcs representing dimension-sets, the task of dimension checking of an object is reduced to a graph checking problem.

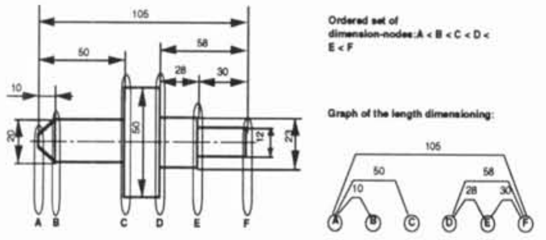
105
50
58
28
30
10
20
50
12
23
A
B
C
D
E
F
Ordered set of dimension-nodes:A < B < C < D < E < F
Graph of the length dimensioning:
105
50
58
10
28
30
A
B
C
D
E
F

Figure 4: Axle with marked dimension nodes (length), ordered set of nodes, and dimensioning graph.

Figure 4 shows the dimension-nodes that are relevant for length checking of an axle and the derived graph that is used for dimension checking.

Incompleteness occurs when there are two dimension-nodes with no path between them. For example, let us assume that the top-most dimension set (valued 105) in figure 4 were missing. Then there would be no path between the nodes A and F. As a result the length dimensioning of the axle would be incomplete.

Redundancy occurs when there are at least two paths between two dimension-nodes. The drawing in figure 4 has redundant dimensioning between the nodes D, E and F.

Inconsistency of Type A occurs when there is redundancy and the different paths do not have the same value.

Inconsistency of Type B occurs when there is a path from dimension node X to dimension-node Y with a negative value, but $X < Y$ (paths against the ordering result in negative values). Figure 4 illustrates this type of inconsistency. The value of the path from C to D has a negative value ($-50 + 105 - 58$) in spite of $C < D$.

The idea of the dimension checking strategy is to build a graph of the dimension-sets and examine all possible paths of the graph in order to detect incompleteness, redundancy, and/or inconsistency. If any of these occur, the user is prompted by the system to select the incorrect dimension-set and/or to provide the correct value. The checking cycle is repeated until no more incorrect dimensioning can be detected. As a result, redundant dimension sets are removed from the interpretation and user inserted values replace the existing ones.

6.2 Drawing redesign

A major problem in using scanned and vectorized images in CAD systems is the poor quality that may result from pure vectorization. The extracted lines do not meet exactly at intersection points, no uniform line thickness of lines belonging to the same class is guaranteed, the lines can be slightly slanted with respect to the X-Y axes, and the size of the drawing may have a scale different from the one given in the dimension sets. Having a complete interpretation of the drawing with correct dimensioning, a redesign of the axle can be done that satisfies all the mentioned requirements of quality. The knowledge source *ks-a-dsg* uses the set of rules for redesigning axles, taking into account any user inserted corrections. In the redesigned drawing the inconsistency caused by the dimension-set value 105 has been corrected to 125, the redundant dimension-arrow with value 28 has been removed, the sides of the rectangle representing the cylinder at end of the axle have all the same thickness and the entire drawing has been slightly rotated so that its center-line is parallel to the X-axis (for more details see [Past92a], [Past92b]). Finally, the input drawing has been redrawn according to the scale determined by the dimension-set values. Figure 5 shows the derived 3D view of that redesigned axle.

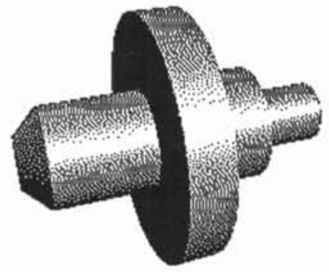

Figure 5: 3D visualization of the axle.

References

[Eng88] Engelmore, R., Morgan, T.: Blackboard Systems, Addison Wesley, 1988.

[Gall88] Gallagher, K.Q., Corkill, D.D., Johnson, P.M.: GBB Reference Manual – Version 1.2, COINS Technical Report 88-66, 1988.

[Glo92] Gloger J., Luth N., Timmermann M.: Scanning, Vectorization and Postprocessing, in: Integrated Management of Technical Documentation: The System SPRITE, Springer Verlag, Berlin, 1992 (to be published).

[Jan87] Jansen H., Timmermann M.: Handskizzeneingabe und Rekonstruktion von 3D-Modellen mit CASUS, in: ZwF82(1987)7, pp. 398-404 (in German).

[Past92a] Pasternak B., Gabrielides G., Sprengel R.: WIZ - A Prototype for Knowledge-Based Drawing Interpretation. Proc. 5th IEA/AIE-92 Paderborn/Germany , pp.164-173, 1992.

[Past92b] Pasternak B., Gabrielides G., Sprengel R.: WIZ - Design and Development of a Prototype for Knowledge-Based Interpretation of Technical Drawings. LKI-Bericht Universität Hamburg, LKI-M-92/1, 1992.