COMPUTER VISION AND IMAGE UNDERSTANDING
Vol. 69, No. 2, February, pp. 196–201, 1998
ARTICLE NO. IV960585

Segmentation and Recognition of Dimensioning Text from Engineering Drawings

Dov Dori and Yelena Velkovitch

Information Systems Engineering, Faculty of Industrial Engineering and Management, Technion, Israel Institute of Technology, Haifa 32000, Israel
E-mail: dori@ie.technion.ac.il, elena@ie.technion.ac.il

Received July 25, 1995; accepted December 3, 1996

Recognition of dimensioning text in engineering drawings is an essential part of the drawing understanding process, as this text provides the exact dimensions and tolerances of the object described in the drawing. We consider engineering drawings produced according to either ISO or ANSI drafting standards. Text segmentation and recognition are preceded by orthogonal zig-zag vectorization, arc segmentation, and arrowhead pair recognition. Initial textbox extraction is done by a region growing process, performed on text-wire candidates. On the basis of textbox context (neighboring annotation wires) the drafting standard is detected. Raw textboxes are divided into logical textboxes, which are further decomposed into basic textboxes. A neural network based OCR algorithm is applied to each basic textbox. Finally, the OCR recognition results are verified by using contextual information and comparing the results with the measurements made directly on the drawing. © **1998 Academic Press**

Key Words: **text segmentation; text-graphics separation; text-graphics association; engineering drawing understanding; CAD conversion; optical character recognition.**

1. INTRODUCTION

Currently, engineering and industry make extensive use of computer aided design (CAD) for creating and processing of engineering drawings. However, by now a great deal of paper-based engineering drawings have accumulated, and a few designers still prefer to use the conventional drafting techniques for producing the drawings. Since maintaining, modifying, and updating paper drawings manually is an expensive and time-consuming process, it is desirable to have a system which will be able to understand paper-based engineering drawings and translate them into CAD representation [1].

An essential part of the drawing understanding process is recognition of dimension text. The problem of dimension text recognition may be roughly separated into the following subproblems: text and graphics separation, text string extraction, association of text with graphics, text string recognition and verification. While several studies have been conducted which are related to these problems, we are not aware of any research that has been directly targeted at this goal.

Fletcher and Kasturi [2] developed an algorithm for text string separation from mixed text/graphics images. It is based on the generation of connected components and the application of Hough transform to group together the components into logical character strings which may then be separated from the text. Lai and Kasturi [3] presented a system for detecting dimension sets in engineering drawings that are drawn according to ANSI drafting standard. It is also based on the generation of connected components and further composing them into text strings, which are associated with dimensioning lines. Neither one of these algorithms treats the problem of text/graphics connectivity nor do they detect text strings consisting of a single character.

Chai and Dori [4] proposed an algorithm for textbox extraction that is preceded by orthogonal zig-zag vectorization, arc segmentation, and arrowhead recognition. The textbox extraction is done by clustering the remaining short bars that are close to each other through a region growing process. As noted by the authors themselves, the algorithm is designed only for detection of text areas without string extraction and may lead to some detection errors.

This paper describes a subsystem for dimension text recognition from engineering drawings produced according to either ANSI or ISO drafting standard, which is integrated into the machine drawing understanding system (MDUS) [5]. The complete recognition is performed beginning with text segmentation and ending with verification of OCR results. In our work we assume that dimensioning text does not contain ANSI standard feature control frames [6] as part of the dimension text and that surface quality symbols are not present in the drawing.

2. BASIC CONCEPTS AND DEFINITIONS

In our work we define and use the following objects (see Fig. 1):

* *Textbox* is a minimal rectangle, enclosing text without any nontext element.

* *Basic textbox* is a textbox, in which the text is a single string.

* *Logical textbox* is a textbox, in which all elements are logically connected and refer to a common element of an engineering

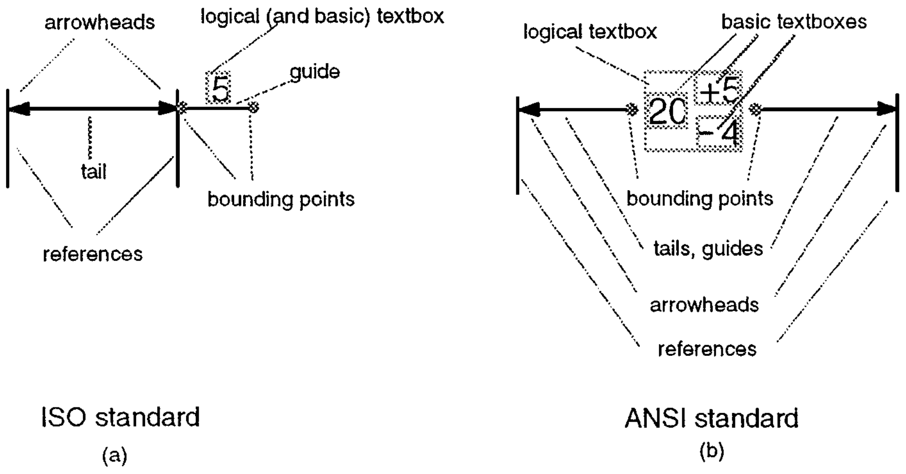

FIG. 1. Example of two dimension sets. (a) The logical textbox coincides with the basic textbox; the textbox guide set is a single bar which does not coincide with any one of the arrowhead tails. (b) The logical textbox consists of three basic textboxes; the textbox guide set coincides with the pair of tails, and the bounding points set is the pair of leader endpoints to the left and right of the logical textbox.

drawing. One logical textbox can contain from one to three basic textboxes. A logical textbox consisting of a single basic textbox may contain the nominal dimension alone or the nominal dimension with its tolerance (e.g., 5 ± 1) or a range (e.g., R6–7). To incorporate tolerances it may contain two basic textboxes (e.g., $25^{\pm 0.5}$) or three (e.g., $25^{+0.2}_{-0.3}$).

* *Subtextbox* is a textbox consisting of one or more basic textboxes enclosed within a logical textbox. The nominal dimension (e.g., 25 in the logical textbox $25^{+0.2}_{-0.3}$) and a pair of tolerance limits (e.g., $^{+0.2}_{-0.3}$ in the same logical textbox) are instances of subtextboxes.

A logical textbox is a part of a dimension set, which contains also a leader set (a single leader or a leader pair), reference for each leader, and a *guide set*, defined below. A leader consists of an arrowhead and a tail, which is a wire (bar or arc). A reference is a wire pointed to by an arrowhead.

* *Guide set* is a set of one or two wires which link the textbox with an appropriate leader set. The guide set consists of one wire which is not a tail of a leader in the case of an outward pointing asymmetric dimension set, as in Fig. 1a. It coincides with the tail or pair of tails if the direction from the textbox to the arrowhead is the same as the arrowhead pointing direction, as in Fig. 1b.

* *Bounding points set* is a set of one or two endpoints of a logical textbox guide or arrowhead tips. If the set has two bounding points, these points lie on opposite sides of the textbox and are closest to it (see Fig. 1).

3. THE TEXT SEGMENTATION ALGORITHM

The text segmentation algorithm consists of text-wire candidate selection, region growing, handling connectivity, standard determination, and standard-dependent processing.

3.1. *Text-Wire Candidate Selection*

A text-wire candidate is a wire (bar or arc) which may potentially belong to text. The process of text-wire candidate selection is governed by the parameter b_{text}, which determines the maximal character height in the drawing. Initially, all the wires are considered text-wire candidates. The elimination is done in two steps:

Step 1. Eliminate all wires which satisfy at least one of the following conditions:

(1) The distance between the wire endpoints is greater than b_{text}.

(2) The wire is a tail of a leader.

(3) The wire is a reference.

(4) Both wire endpoints coincide with tips of arrowheads comprising the common pair.

Step 2. Eliminate each wire in a text-wire candidate chain whose two endpoints coincide with endpoints of eliminated wires. All remaining wires are considered text-wire candidates.

3.2. *The Region Growing Process*

The region growing process is performed as described in [4], except that only the text-wire candidates are used as input to this process. Around each text-wire candidate, we construct a textbox as the minimal rectangle containing the wire. Before adding this new textbox to the textbox list, we examine all the textboxes already in the list and check whether any one of them is within a distance of b_{text} at most from the candidate textbox. If such a textbox x is found, the candidate is merged with it, and the corners of x are updated to accommodate the added candidate. Another pass is then performed to check if, as a result of the last merger, two distinct textboxes can now be merged. The result of this procedure is a list of textbox candidates. From

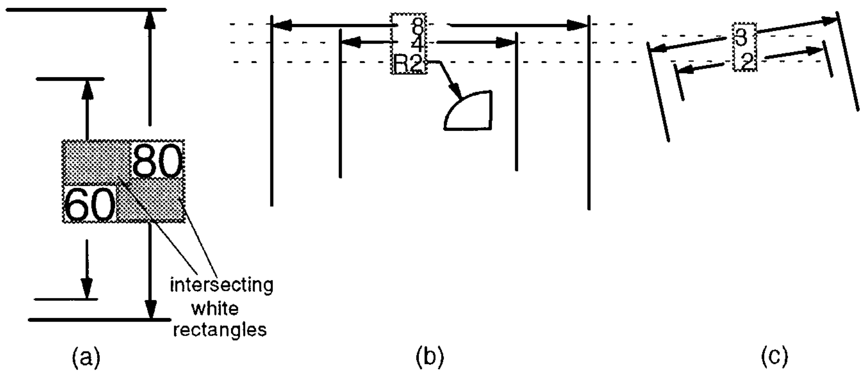

FIG. 2. Examples of textbox candidates containing more than one logical textbox in ANSI drawings. (a) First type: dividing the textbox candidate according to the two white rectangles in opposite corners (the white rectangles are denoted by grey color). (b), (c) Second type: dividing the textbox candidate according to the textbox axis of symmetry (denoted by dotted lines).

this list we delete the candidates which are too long and thin, and those whose area is too small.

Many of the textbox candidates obtained as a result of the region growing process do not exactly coincide with real textboxes due to one or more of the following reasons:

(1) Some textbox candidates may be false alarms.

(2) The detected textbox boundaries usually do not coincide exactly with the actual ones.

(3) The actual textboxes may enclose separate small angular dimension text symbols, such as $^\infty$, $'$, and $''$, lying outside the textbox candidates, since no wire was found for these symbols during the vectorization process.

(4) Two or more logical textboxes may be merged into a single textbox.

(5) All textboxes are considered horizontal, whereas in fact some textboxes in ISO drawings may be vertical or tilted.

(6) In ANSI drawings, some dimensions can be enclosed within rectangles, denoting basic dimension [6]. Such rectangles should be found and removed from the textboxes while keeping record of their existence.

3.3. *Handling the Text/Graphics Connectivity Problem*

In order to separate connected text and graphics elements we remove from the picture all pixels belonging to the graphic components, which may be connected with text. Candidates for such graphic components are bars, arcs and arrowheads located close to textbox candidates detected by the region growing process.

3.4. *Standard Determination*

We wish to determinate as early as possible the standard (which may be ISO or ANSI), according to which the drawing is produced. The standard determination is performed by considering the position of the detected textbox candidates relative to their presumed leader sets. During this process we account for possible errors in the initial textbox detection and detection of other primitives.

3.5. *Standard Dependent Processing*

Having determined the standard, we use the standard's characteristics to solve problems concerning textbox candidates obtained as a result of the region growing process as described below.

By considering leader sets and their neighborhood, we then associate a unique textbox with each leader set. During this process we also find a guide set and a bounding points set for each textbox. The bounding points sets are used to separate second type of merged logical textboxes corresponding to Figs. 2b and 2c. We separate such logical textboxes by detecting their horizontal axis—the line defined by the average of the Y-coordinates of the bounding points—and splitting the corresponding textbox candidate along white runs that are about equally spaced with respect to each axis.

Finding rectangles of basic dimensions and textbox boundaries adjustment. For each logical textbox candidate we check for presence of a rectangle (denoting basic dimension) around its text. If such rectangle is found, it is removed and its detection is recorded. Next, we adjust the boundaries of each logical textbox to coincide with the real ones by considering the pixel representation of the original engineering drawing image.

ANSI drawing processing. We first find those textbox candidates which contain two or more logical textboxes and split them into single logical textboxes. This is done in two steps, separated by the process of finding a unique leader set and bounding points set for each logical textbox candidate. Then a rectangle (denoting basic dimension [6]) is searched around each logical textbox. Next, the boundaries of each logical textbox are adjusted to coincide with the real ones. Finally, basic textboxes are extracted from the logical textboxes. The details of this process follow.

Merged logical textboxes separation. There are two possible types of textbox candidates containing more than one logical textbox, as shown in Fig. 2. The first type, shown in Fig. 2a, usually has vertical leader pairs, while the second type has leader

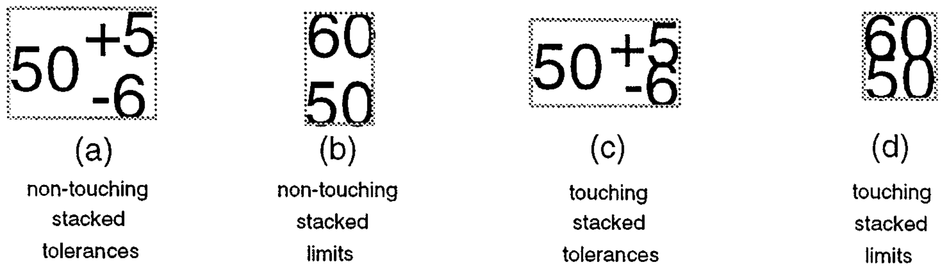
50 +5 -6
(a)
non-touching stacked tolerances
60 50
(b)
non-touching stacked limits
50 +5 -6
(c)
touching stacked tolerances
60 50
(d)
touching stacked limits

FIG. 3. Examples of logical textboxes containing more than one basic textbox in ANSI drawings.

pairs whose projection on the horizontal axis is longer than their projection on the vertical axis, as shown in Figs. 2b and 2c. Another case which looks similar but is actually one logical textbox with lower and upper limits but with no nominal value is presented in Fig. 6, where both dimension logical textbooks have this structure.

To extract logical textboxes from the textbox candidates of the first type, we search for intersecting white rectangles in opposite corners of the textbox candidate. If such rectangles are found, then we separate the textbox into two proper textboxes. This process is repeated for each of the two resulting textboxes, until no more intersecting white rectangles are found.

Basic textboxes extraction. Typical logical textboxes containing more than one basic textbox are shown in Fig. 3. The extraction of nontouching basic textboxes, shown in Figs. 3a and 3b is done by considering first white rectangles in the left upper and lower textbox corners for the case of stacked tolerances and then in its middle for both stacked tolerances and stacked limits.

The *textbox heights histogram* is a pseudo graphical tool that provides for examining the distribution of the heights of the textboxes. The height of each detected textbox is a value used in this histogram. The typical character height is determined by finding the mode (maximum occurrence) in the textbox heights histogram.

To separate basic textboxes in the case of touching stacked characters, shown in Figs. 3c and 3d, we use this typical character height. We then find the stacked limits logical textboxes and stacked tolerances right subtextboxes as those whose height is close to twice the character height and divide them horizontally into two equal height basic textboxes.

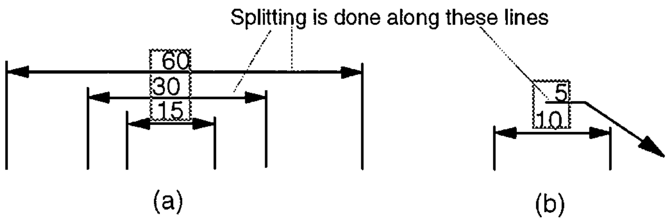

FIG. 4. Dividing merged logical textboxes in ISO drawings.

ISO drawings processing. For ISO standard drawings we first find those textbox candidates which contain two or more logical textboxes and split them into single logical textboxes. We then associate each logical textbox with a leader pair and find a bounding points set for each logical textbox. The bounding points location is used to obtain the actual orientation of the textboxes, which, in ISO, is not necessarily horizontal. Next, the boundaries of each logical textbox are adjusted to coincide with the real ones. Finally, basic textboxes are extracted from the logical textboxes. The details of this process follow.

Merged logical textbox separation. To separate logical textboxes which are merged into a common textbox candidate, we split each textbox candidate along each horizontal and vertical bar crossing at least one of its sides and penetrating sufficiently deep into it (see Fig. 4).

Detection of textbox orientation. First, we associate logical textboxes with leader sets. Each logical textbox is then assigned a new coordinate system, obtained from the original one by rotating it by an angle α, which depends on the position of the corresponding bounding points and the leader type. In the new coordinate system the logical textboxes are horizontal and text is written from left to right rather than diagonally. Now we construct these newly oriented textboxes in the original coordinate system and adjust their boundaries to coincide with the actual ones.

Basic textbox extraction. We first divide the logical textbox into left and right subtextboxes. This is done by considering the extent of variation the text top and bottom heights along the logical textbox. Then the right subtextbox is divided into upper and lower tolerance basic textboxes as in ANSI drawings.

4. TEXT STRING RECOGNITION

To perform the text string recognition we divide the string into single characters, recognize each one of them separately, combine the whole string of recognized characters, and improve the recognition result through geometric and numeric verification, as described below.

4.1. Single Character Segmentation

Usually, there is at least one white run separating the characters vertically from each other. However, sometimes the characters may touch one another. Therefore, for ANSI drawings, we first divide all the text strings in the drawing into minimal rectangles, separated by at least one white run. Most of them contain only one character; hence the histogram of the rectangle width has a maximum, corresponding to the most frequent character width w. Having estimated w, we find rectangles having an approximate width of $n \times w$ ($n = 2, 3, \ldots$) and divide these rectangles into n equal parts. For ISO drawings, two character sizes are possible, the larger for the nominal value and the smaller for the tolerance. We therefore group all the basic textboxes into two types according to the expected character size and perform the above process separately for each one of the two basic textbox sizes.

4.2. *Character Recognition Algorithm*

Twenty-three characters (and symbols) may be encountered in dimension text: 10 digits (from 0 to 9) and the following symbols: $\varnothing$, R, X, $\sim$, (,), $+$, $-$, $\pm$, ., ^{c}, $'$, $''$. To recognize them, we first identify each character as belonging to one of four classes according to the relative size of the minimal rectangle containing the character and its position in the string. The first class contains the 10 digits and the eight symbols $\varnothing$, R, X, $\sim$, (,), $+$, $\pm$. The second class consists of the three small symbols used in angular dimension sets: ^{c}, $'$, and $''$. Each of the third and fourth classes contains only one symbol, "$-$" and ".," respectively.

Having divided all the symbols into these four classes, we normalize the character size to a 12×12 image and use a previously trained neural network [7] to recognize characters within the first two classes. The two other classes require no further processing. The training set we used consisted of about 1700 samples.

4.3. *Geometric and Numeric Verification*

The character recognition verification consists of the following steps:

Step 1. *Contextual verification within single basic textboxes.* We classify each basic textbox as belonging to one of eight classes according to its position within its logical textbox. Basic textboxes of the same class must have a common structure and satisfy the same requirements. To check if a basic textbox satisfies these requirements and to correct the OCR results, a bigram lookup table first determines the possibility for symbols to follow one another and corrects the symbols automatically, if it is possible, or turns to the help of a human operator otherwise. Several rules, which cannot be expressed by the table, are then checked.

Step 2. *Comparison between recognized dimension values and values measured from the drawing.* First, we wish to determine the *combined scale,* i.e., the number of pixels per millimeter of the actual object described in the drawing (as opposed to resolution, which is the number of pixels per millimeter of the drawing). To do this, we construct a histogram of the ratio of the distance between arrowhead tips of the same arrowhead pair (measured from the drawing in pixel units) to the nominal value of the corresponding logical textbox. The ratio value corresponding to the histogram peak is chosen as the combined scale. Recognized nominal values of the distance and angle are compared with the corresponding distances and angles, measured from the drawing. If they are not close enough, help of a human operator is called for. When the upper and lower bounds of the nominal dimension are provided, instead of the nominal itself, the mean is taken as the nominal value for reference.

An interesting problem arises in ANSI drawings, where a decimal point may precede a value without a leading zero, e.g. ".75" rather than "0.75." It is only through this comparison between recognized dimension values and values measured from the drawing that such a dot can be detected. If there is a factor of approximately $1:10$ between recognized and measured value, the hypothesis of the existence of a misdetected decimal point is tested by searching for that point in the area where it is expected to show up.

Step 3. *Verification of tolerance values according to relative dimension values.* Tolerance values are supposed to be significantly smaller than the corresponding nominal value. If this condition is not satisfied, a human operator help is used.

5. EXPERIMENTAL RESULTS

Fourteen engineering drawings of size A4 containing an average of five dimension sets and 12 symbols were processed according to our algorithm. It should be emphasized that all 14 drawings were hand printed, some (as those in Figs. 5 and 6) with the use of templates and others manually without any auxiliary device.

The input data were files of scanned drawings at 300 dpi resolution in Sun-Raster or TIFF format. The research was done within the framework of the machine drawing understanding system (MDUS) [1, 5, 8, 9], which is written in C and C++ and operates on Sun and Silicon Graphics workstation in Unix/X-Window environment.

The recognition results for its main steps are summarized in Table 1. The percentage calculated for each step takes as a basis only entities recognized correctly in the previous step. Examples of recognition results are shown in Figs. 5 and 6. Figure 5 is an example of a medium complexity drawing, which was one of the 14 drawings tested.

TABLE 1
Recognition Accuracy for Various Steps of the Algorithm

Recognition step	Accuracy
Logical textbox extraction	89%
Dividing logical textboxes into basic textboxes	96%
Character recognition before verification	89%
Character recognition after verification	99%

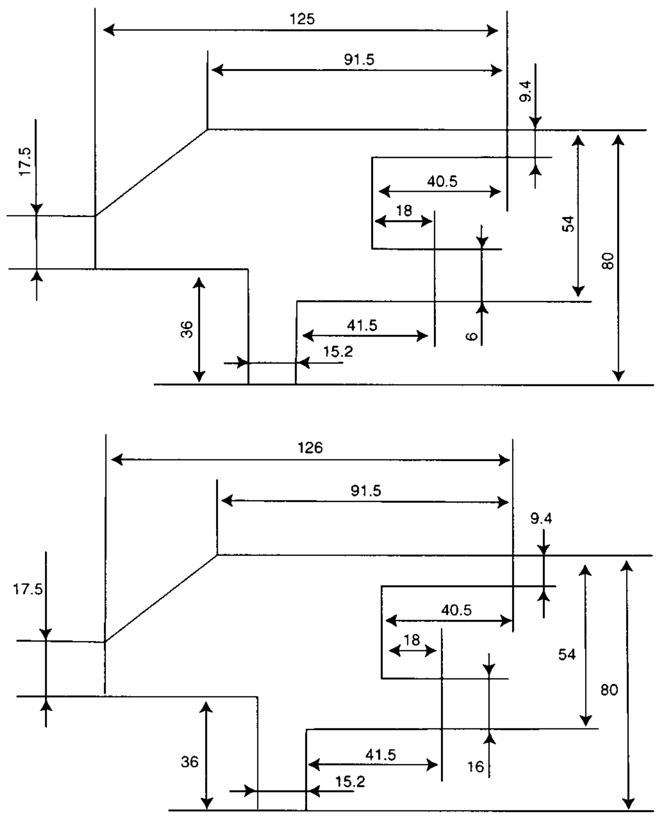

FIG. 5. An example of dimensioning text recognition from the ISO drawing (a) The original image. (b) The results of text recognition. There is one incorrectly recognized character, "5" in the string "125."

6. SUMMARY

An algorithm for recognition of dimensioning text in engineering drawings was developed and implemented. Initial text regions are obtained by region growing process, the drafting standard is determined and its features are used for text string extraction and association with the graphics. Next, a neural network based OCR algorithm is applied and its results are improved by applying a verification process.

Vector-based segmentation of text connected to graphics in engineering drawings is a hard problem which has recently been tackled successfully in [10]. The work does not deal with the recognition itself and should be combined with the work presented in this paper.

Further research should concentrate on both segmentation and recognition improvement. Text segmentation can be improved by using rules deduced after considering detection errors in a great number of real-life engineering drawings. The OCR algorithm which may also be improved by using a significantly larger set for training the neural network. Moreover, updating the neural network weights may be done automatically after each verification process, so that the system will in effect perform self-teaching.

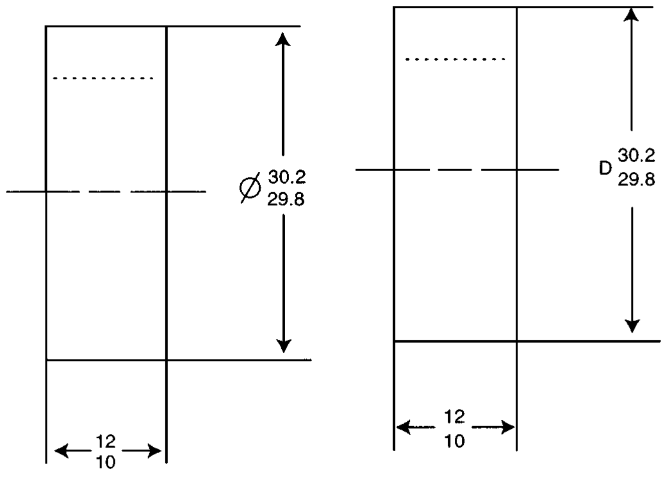

FIG. 6. An example of dimensioning text recognition from the ANSI drawing. (a) The original image. (b) The results of text recognition.

ACKNOWLEDGMENT

This research was supported by the Technion VPR Fund for the Promotion of Research at the Technion.

REFERENCES

1. D. Dori and K. Tombre, From engineering drawings to 3D CAD Models: Are we ready now? *Comput. Aided Des.* **27**(4), 1995, 243–254.
2. L. A. Fletcher and R. Kasturi, A robust algorithm for textbox string separation from mixed text/graphics images, *IEEE Transactions on Pattern Analysis and Machine Intelligence—T-PAMI* **10**(6), 1988, 900–918.
3. C. P. Lay and R. Kasturi, Detection of dimension sets in engineering drawings, *IEEE Transactions on Pattern Analysis and Machine Intelligence—T-PAMI* **16**(8), 1994, 848–854.
4. Ian Chai and Dov Dori, Extraction of text boxes from engineering drawings, *Proc. Society of Photo-Optical Instrumentation Engineers* SPIE **1661,** 1992, 38–49.
5. D. Dori, Y. Liang, J. Dowell, and I. Chai, Sparse-pixel recognition of primitives in engineering drawings, *Machine Vision and Applications* **6,** 1993, 69–82.
6. *Dimensioning and Tolerancing,* The American Society of Mechanical Engineers, New York, 1983 [ANSI Y14.5M-1982]
7. P. Baffes, *NETS—Neural Network Simulator,* Software Technology Branch NASA, Johnson Space Center. [ftp from ftp.technion.ac.il/pub/unsupported/dos/simtel/neurlnet/nasanets.zip]
8. D. Dori, Representing pattern recognition-embedded systems through object-process diagrams: the case of the machine drawing understanding system, *Pattern Recognition Letters* **16**(4), 1995, 377–384.
9. D. Dori, Arc segmentation in the machine drawing understanding environment, *IEEE Transactions on Pattern Analysis and Machine Intelligence—T-PAMI* **17**(11), 1995, 1057–1068.
10. D. Dori and L. Wenyin, Vector-based segmentation of text connected to graphics in engineering drawing, in *Advances in Structural and Syntactic Pattern Recognition* (P. Perner, P. Wang, and A. Rozenfeld, Eds.), pp. 322–331. Springer-Verlag, New York/Berlin, 1996. [Lecture Notes in Computer Science, Vol. 1121]