# Graphics Recognition Techniques

# 15

Josep Lladós and Marçal Rusiñol

## Contents



J. Lladós (✉) • M. Rusiñol
Computer Vision Center & Computer Science Department, Universitat Autònoma de Barcelona, Bellaterra, Spain
e-mail: josep@cvc.uab.es; marcal@cvc.uab.es

D. Doermann, K. Tombre (eds.), *Handbook of Document Image Processing and Recognition*, DOI 10.1007/978-0-85729-859-1_18,

**Abstract**

This chapter describes the most relevant approaches for the analysis of graphical documents. The graphics recognition pipeline can be splitted into three tasks. The low level or lexical task extracts the basic units composing the document. The syntactic level is focused on the structure, i.e., how graphical entities are constructed, and involves the location and classification of the symbols present in the document. The third level is a functional or semantic level, i.e., it models what the graphical symbols do and what they mean in the context where they appear. This chapter covers the lexical level, while the next two chapters are devoted to the syntactic and semantic level, respectively. The main problems reviewed in this chapter are raster-to-vector conversion (vectorization algorithms) and the separation of text and graphics components. The research and industrial communities have provided standard methods achieving reasonable performance levels. Hence, graphics recognition techniques can be considered to be in a mature state from a scientific point of view. Additionally this chapter provides insights on some related problems, namely, the extraction and recognition of dimensions in engineering drawings, and the recognition of hatched and tiled patterns. Both problems are usually associated, even integrated, in the vectorization process.

**Keywords**

Dimension recognition • Graphics recognition • Graphic-rich documents • Polygonal approximation • Raster-to-vector conversion • Texture-based primitive extraction • Text-graphics separation

## Introduction

Graphical documents are instances of graphical (visual) languages, i.e., valid expressions of a diagrammatic notation. Graphical languages consist of terms, i.e., lexical units, and relational rules involving a certain type of grammar. The most common terms extracted from graphical documents are textual labels, lines and arcs, loops, solid regions, dotted lines, and hatched patterns. Relational rules define valid bidimensional relations between terms depending on the domain. Combined with domain-dependent semantics, terms and associated rules are interpreted giving rise to valid visual concepts. Most of graphic documents consist in line drawing structures. Because of that, primitive extraction in graphics recognition usually involves a vectorization process. Taking vectorization as the core problem, this chapter will be focused on the specific techniques to recognize graphical terms, namely, text, vectorial primitives, and repetitive patterns. These units are the basis of the subsequent tasks of the graphics recognition pipeline: symbol recognition and graphical document interpretation, studied in the next chapters ▸Chaps. 16 (An Overview of Symbol Recognition) and ▸17 (Analysis and Interpretation of Graphical Documents).

## History and Importance

The recognition of technical documents of different areas of engineering (mechanical, electrical, civil, etc.) became very popular in the 1990s. Document analysis had been mainly addressed the text recognition problem. Optical character recognition was one of the typical pattern recognition problems. But there was a growing industrial need to convert raster images of scanned drawings and maps to formats compatible with Computer-Aided Design (CAD) software and Geographic Information Systems (GIS). With the advent of software platforms for creating and editing such types of documents, the collateral challenge was the ingest of the existing large collections of engineering drawings and maps in paper format. The need was to convert these documents to a format compatible with such platforms for a subsequent edition. Hence, vectorization was the seed of an emerging graphics recognition activity that for decades has been, along with text recognition and layout analysis, the third leg of the document analysis field.

In addition to line primitives, graphical documents also contain textual labels (e.g., dimension labels or toponymy). Such labels are present mixed to graphics, in different orientation and font sizes. Classical OCR approaches are not able to recognize them. It leads to refer to the problem of text-graphics separation.

Methodologically, vectorization and text-graphics separation are considered two different problems; however, from an applied point of view, both can be considered as complementary in the task of converting technical drawings to a CAD or GIS format. As indicated above, their origin came from an industrial need, and a research community grew in the 1990s around it. Nowadays, many mature commercial applications exist and the interest of the scientific community has evolved to focus on problems related to the recognition and interpretation of graphical documents. The problem of text-graphics separation has nowadays a certain resurgence. In business document applications (e.g., processing of forms and bank checks), it is still a need to separate printed or handwritten text from rule lines.

## Evolution of the Problem

Many solutions have been proposed, either from the academia or the industry, and it seems to be an already mature problem. The evolution, in terms of performance, has been slow. Vectorization improvements have consisted in approaches that obtain more accurate approximations of lines with the minimum number of primitives. The problem of text-graphics separation has evolved improving the difficulty of text touching graphics and with methods more invariant to scale and orientation. In both cases, the major concern has been to get rid of the sensitiveness to a number of parameters.

Table 15.1 shows an interesting observation on the activity of the research evolution on graphics recognition techniques. It shows the percentage of papers presented for different topics in the graphics recognition workshop (GREC) series

**Table 15.1** Evolution of the graphics recognition topics in GREC workshop series

| | GREC95 (%) | GREC97 (%) | GREC99 (%) | GREC01 (%) | GREC03 (%) | GREC05 (%) | GREC07 (%) | GREC09 (%) |
|---|---|---|---|---|---|---|---|---|
| Low-level processing | 16 | 7 | 0 | 10 | 6 | 6 | 0 | 0 |
| Vectorization and text extraction | 16 | 13 | 10 | 16 | 6 | 19 | 17 | 14 |
| Technical drawings maps | 21 | 30 | 29 | 19 | 18 | 0 | 8 | 3 |
| Layout analysis and diagrammatic notations | 16 | 13 | 6 | 13 | 3 | 8 | 3 | 8 |
| Applications, systems, and architectures | 0 | 13 | 10 | 13 | 12 | 6 | 0 | 3 |
| Symbol and shape recognition | 11 | 13 | 23 | 6 | 18 | 25 | 14 | 14 |
| Retrieval, indexing, and spotting | 5 | 0 | 6 | 10 | 15 | 11 | 14 | 11 |
| Sketching interfaces | 0 | 0 | 3 | 10 | 18 | 8 | 11 | 8 |
| Performance evaluation | 16 | 10 | 13 | 3 | 6 | 6 | 17 | 8 |
| Historical documents | 0 | 0 | 0 | 0 | 0 | 11 | 14 | 11 |

over the years.[1] The activity on graphics recognition techniques is quite stable. Although in general the major challenges of the graphics recognition community have evolved, there is still room for improvements and applied research on basic techniques.

## Applications

The graphics recognition techniques described in this chapter are applied to technical diagrams belonging to different engineering disciplines. The most relevant application domains are:

- **Electrical and logic circuit diagrams**. It is one of the early areas of graphics recognition. Electronic schematics have a standardized notation that makes the processing easier, with rectilinear structures representing wires, and loop-like structures corresponding to electrical components.
- **Engineering drawings**. They can be seen as the assembly of low-level primitives such as arcs and straight lines, dashed lines, crosshatched, and solid areas. The recognition of these low-level primitives combined with domain-dependent knowledge gives meaning to the document and allows them to be converted to a CAD format. Thus, a hatched area combined with a semicircle could represent a screw, but a similar hatched area in another part of the same document could represent a section of a solid part. The recognition of dimensioning annotation plays an important role in such documents.
- **Architectural drawings**. A growing number of methods have been proposed for recognizing architectural drawings for conversion to CAD environments and subsequent actions as edition or 3D visualization. A particular focus of attention has been made on hand-drawn sketches, where actions such as vectorization and text detection are more imperfect due to the natural deformation of strokes. As in engineering drawings, the recognition of dimensioning annotation is very frequent.
- **Maps**. Map-to-GIS conversion has been an active application field over the last decades. Maps are one of the most difficult types of technical diagrams. Different classes of maps exist. Cadastral maps consist of polygonal shapes and hatched patterns, representing parcels, with circumscribed text annotations. Utility maps consist of lines and small symbols composed of geometric basic primitives (squares, circles, arrowheads, etc.). Finally, geographic maps with line objects that usually represent isolines and roads and small symbols whose meaning is given by a legend. In this kind of maps, color information plays an important role in the segmentation. The detection of symbols is usually legend-driven, so text-graphics separation is very important to read the legend.

## Main Difficulties

Although stable and robust vectorization methods exist, it is not clear what vectorization should be or should output. Vectorization is not an isolated step and its accuracy depends on the goal of the whole system. In some cases, the best the primitives fit into lines, the better is vectorization; however, in other cases, the goal is the minimum number of primitives although the vectors roughly approximate the lines. Usually vectorization systems provide a set of parameters that have to be set by the user or customized in terms of document categories.

One of the worst difficulties of text-graphics separation algorithms is the case of text touching graphics. Although this configuration can be detected, in some cases the reconstruction of characters for the subsequent recognition is very difficult. As for vectorization, the avoidance of parameters, or their customization in terms of document properties, is another difficulty of text-graphics separation methods.

## Summary of the State of the Art

Many solutions have been proposed, either from academia or industry, for the graphics recognition techniques reviewed in this chapter. Vectorization, i.e., raster-to-vector conversion, is a mature problem; however, the community has not yet reached stable methods because of the lack of a consensus on what a vectorization system ought to output. Vectorization consists in a kind of polygonal approximation of the medial axis of line images [32, 36]. Skeleton-based approaches tend to displace the junction points and to be sensitive to image irregularities. Another family of methods is based on line following or line contour matching [24]. These methods are more precise in junction finding but their performance decreases in complex images. Sparse-pixels approaches are considered to be a third family of methods [22]. These approaches do not analyze all the image pixels but detect vectors by analyzing key points in terms of local neighboring configuration. Vectorization approaches often involve the detection of other primitives than straight segments, but curve ones, in particular circular arcs [21].

Another classical problem in graphics recognition is text-graphics separation. Graphical documents often contain text annotations (names of streets in maps, dimensions in engineering drawings, beat in musical scores) that must be segmented before being recognized by an OCR module. Text can be segmented looking for small connected components following a regular path. However, text is sometimes connected to graphics. A baseline reference to solve it is the algorithm of Fletcher and Kasturi [25] and subsequent improvements such as Tombre et al. [68]. Other classical methods work at a multiresolution representation [64] which allows to detect text of different sizes and orientations. A recent approach proposed by Hoang and Tabbone [29] consists in an elegant formulation inspired by techniques from the signal processing field.

## Outline of the Chapter

This chapter addresses the different tasks of graphics recognition that are applied when analyzing line drawings. First in section "Text Recognition in Graphical Documents," the main algorithms for separating text from graphics are described. Although there is no standard pipeline, text-graphics separation is usually performed first, so afterwards a raster-to-vector process is applied to the graphical entities, and an OCR to the textual ones. In section "Raster-to-vector Conversion," vectorization algorithms are reviewed. Section "Recognition of Dimensions in Technical Drawings" provides a short overview on the recognition of dimensions, a classical entity appearing in engineering drawings. Section "Extraction of Texture-Based Graphical Primitives" is addressed to review the main approaches for the recognition of textured patterns (one-dimensional and bidimensional) such as dotted lines, or hatched areas. Finally, section "Conclusion" draws the conclusions and perspectives.

## Text Recognition in Graphical Documents

Graphical documents usually contain text labels (e.g., dimensions in mechanical drawings, place or river names in maps, room names in architectural drawings, labels in flowcharts). This kind of textual terms cannot be recognized following classical OCR approaches that assume a regular layout organized in columns, paragraphs, and lines (Manhattan layout). Common problems of text analysis in graphical documents are touching to graphics, multioriented or even following curvilinear paths, reduced lexicons, etc. In this section, the state-of-the-art algorithms for detecting text strings in graphical documents will be described.

### Text-Graphics Separation

Graphical documents at large rely on a diagrammatic notation dependent of the domain. Each individual entity such as an arc, a dashed line, a hatched area, or a symbol, has a meaning in a given context and according to a syntax. Text labels are used to give attributes to the graphical entities. For example, a numerical string in an engineering drawing or an architectural plan informs the reader about the dimension of an element or a text label in an isoline of a map informs about the altitude in the geographical location represented by the isoline. A key step in the conversion or raster images to CAD formats requires the segmentation and recognition of text labels, and their interpretation jointly with the graphical parts.

The text-graphics separation process aims at segmenting the document into two layers: a layer assumed to contain text characters and annotations and a layer containing graphical objects. As introduced before, the recognition of text strings may be very relevant for the interpretation of the graphical artifacts. Most approaches use to perform text separation at early stages in the processing pipeline, usually using image processing techniques and limited knowledge about the configuration of the image in terms of higher-level objects.

As Lu suggests in [46], the differentiation between text and graphics in a graphical document is sometimes subjective if only the local distribution of pixels is considered, without taking into account the context. An isolated small line can be part of an "I" character but also part of a dashed line, the end of a dimension component, etc. Although there is some overlapping due to the mentioned higher-level interpretation, a common definition of both categories may be stated as follows:

- **Text**: symbols or strings for the interpretation of the document parts, including letters, words, digits, and/or special symbols
- **Graphics**: non-textual components having a domain-dependent meaning according to a diagrammatic notation, including all kinds of lines, curves, solid, or textured areas

The use of low-level information for discriminating text in graphical documents requires the consideration of a number of geometric and topological properties at

pixel or region level. These properties will drive the heuristics for the process. According to Lu [46], in technical drawings, some geometric features that differ between textual and graphical components can be observed:

- The size of text characters is often much smaller than that of graphics components. Changes in text size or shape are within a narrow range.
- Text characters usually appear in the form of short strings, having uniform size, inter-character distance and a "smooth path" (usually rectilinear and oriented horizontally, vertically, or slanted at an angle of 45°).
- The local stroke density of text regions is often much higher than that of graphics.
- The length of the linear components included in strokes of text strings is much shorter than that of graphics.

It is quite straightforward to implement algorithms that filter the image parts (connected components or segments) according to the above features. Figure 15.1 illustrates the expected output of this process. However, the detection of text components in graphical documents has a number of difficulties that require more sophisticated steps. The main difficulties are the following:

- Graphical components include lines of any length, thickness, orientation, and pattern (continuous, dashed, dotted). Closed curves of different order (circles, ellipses, etc.) or polylines can be unfilled, filled with solid areas, hatched, tiled.
- Text and graphic parts appear mixed, sometimes touching or overlapped. In some cases, there is no clear geometric difference between them (e.g., small components of dotted or dashed lines can be confused with characters or punctuation symbols).
- Text can be of any font, size, and orientation. The problem of multioriented text is an important difficulty that makes unusable traditional OCR techniques. Engineering drawings use to present text strings in vertical, horizontal, or slanted rectilinear orientation; however, in other documents like maps, text may appear in curvilinear orientations.

Tombre et al. in [68] proposed three families of text-graphics separation methods. Recently, Hoang and Tabbone [29] slightly reformulated this classification. According to the previous works, the following taxonomy of text-graphics separation approaches is proposed:

- Morphology analysis
- Connected component analysis
- Line (vector)-based segmentation
- Multiresolution analysis
- Signal processing
- Ad hoc methods (forms, music)

In the next subsections the different families will be further described.

### Methods Based on Morphology Analysis

A number of methods use morphological operators with the assumption of that the text is what remains after applying iterative openings to the original image with structuring elements designed to eliminate rectilinear objects. The method proposed by Wahl et al. [70] is one of the first methods based on morphological

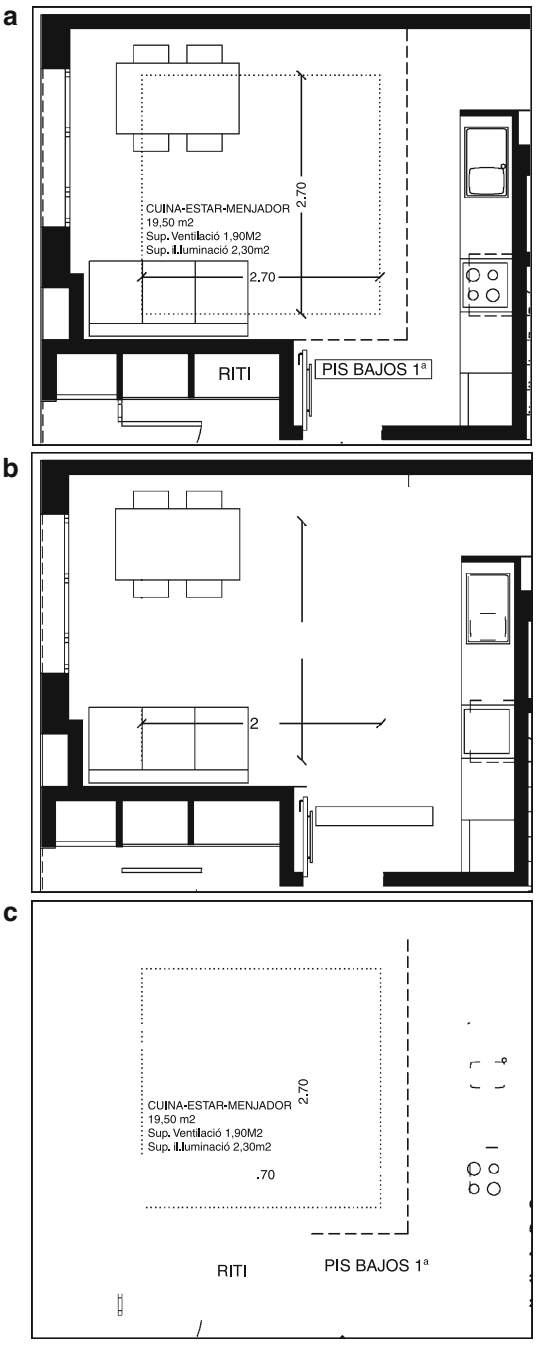


**Fig. 15.1** Text-graphics separation example. (**a**) Original image, (**b**) graphical layer, (**c**) textual layer

filtering. It uses Run-Length Smoothing Algorithm (RLSA) to detect vertical and horizontal text strings. RLSA can be seen as morphological closing (or opening) operations with vertical and horizontal structuring elements of length according to the text size and graphical lines width. The method of Lu [46] uses RLSA too. The main improvement of this work is that it allows to detect slanted lines by performing a stretching operation at different angles. The main drawback of these approaches is that they tend to wrongly label text as graphics. RLSA has proven to be efficient in separating rule lines in forms, but its use in graphics documents is less frequent.

### Methods Based on Connected Component Analysis

This family of methods is one of the most commonly used. A pioneer and well-know work was proposed by Fletcher and Kasturi [25]. The basic idea is to segment text based on basic perceptual grouping properties. Thus, simple heuristics on font size, inter-character, word and line spacing, and alignment are used. The method requires many thresholds, but the good point is that they are extracted from the image object properties and are not manually set a priori. The main steps of this method are:

1. Connected component generation
2. Filtering of connected components based on area and size
3. Connected component grouping in terms of area and size to cluster those that are likely to belong to the same font size, so are candidate to be in the same string
4. Hough Transform applied to the centroids of connected components (text strings are supposed to have a rectilinear arrangement)
5. Logical grouping of strings into words and phrases. This step intends to capture those components kept aside by the Hough step, but that fall into the potential text area (in terms of interline spacing, intercharacter gaps, etc.). For example, a period at the end of a string or an accent
6. Text string separation

This method combines simplicity with good performance and scalability to different text properties. This is probably the reason that most of the methods are based on the Fletcher and Kasturi one, with small variations and adaptations to different contexts. The weakness of this method is that it does not cope with text touching graphics. Tombre et al. [68] proposed an improved approach able to separate text touching to graphical parts. In addition, they introduced some more heuristics allowing to improve the performance. Many commercial systems nowadays use approaches inspired by the ones described above. Figure 15.2 illustrates the main steps of the methods based on the grouping of connected components.

### Methods Based on Vectorial Representations

There are some approaches where the segmentation of text is performed at vector level, i.e., after the image is vectorized (see section "Raster-to-Vector Conversion"), instead of pixel level [17, 18, 35]. The main idea consists in recursively grouping short line segments in the vectorial image. Thus, after a continuous short segment is selected as a seed, touching segments are iteratively merged. This procedure allows to obtain a coarse detection of character bounding boxes that are then grouped in

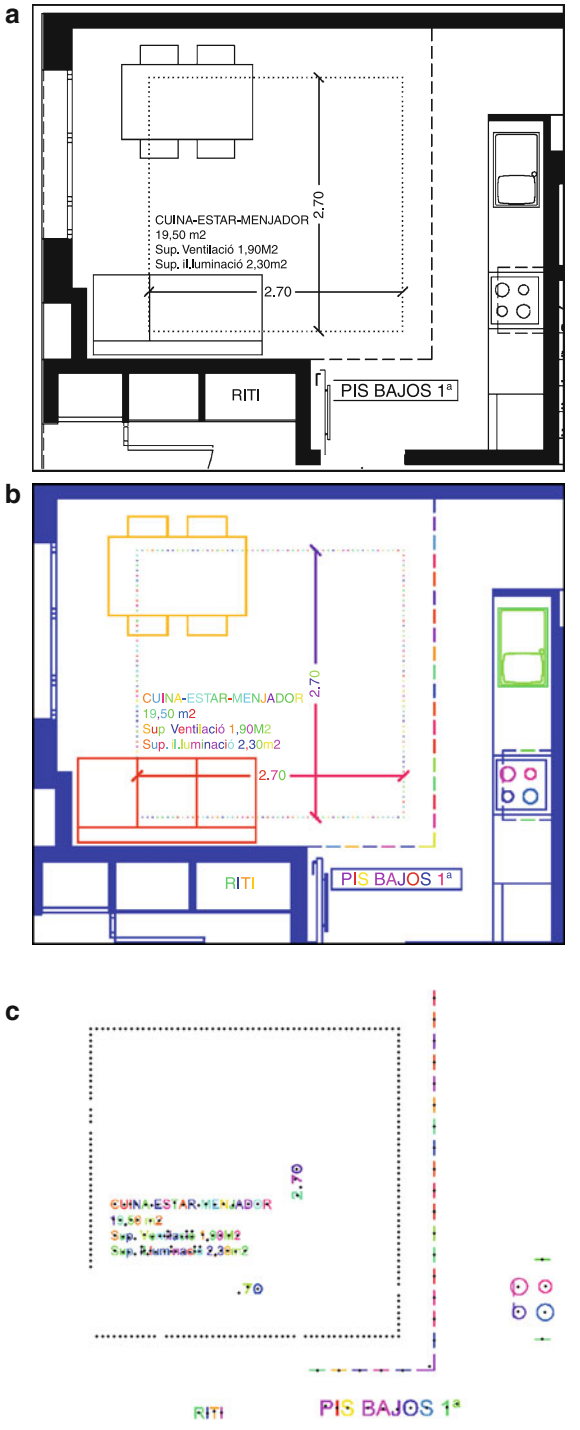


**Fig. 15.2** (continued)

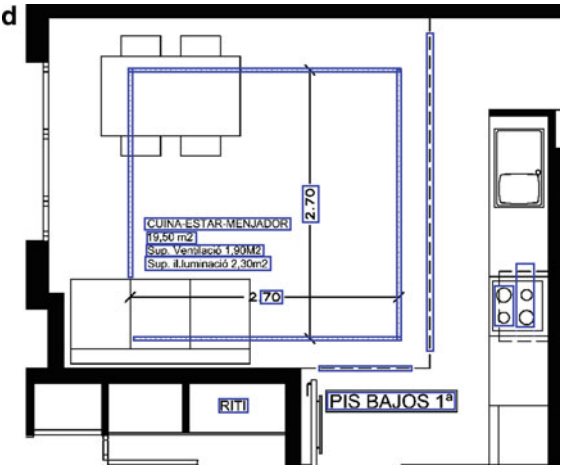


**Fig. 15.2** Text-graphics separation steps. (**a**) Original image, (**b**) connected components, (**c**) filtering of connected components, (**d**) result of string grouping

terms of their alignment. The basic process of this method does not differ from the methods based on connected component grouping. It is a bottom up process that instead of grouping pixels groups vectors.

### Methods Based on Multiresolution Representations

Multiresolution approaches are based on visual perception principles. Each resolution allows to highlight a specific category of information. Tan and Ng proposed a multiresolution approach in [64] based on the construction of a regular pyramid over the image. A pyramid representation is a well-known multi-scale signal representation in which an image is subject to repeated smoothing and subsampling. Historically, the pyramid representation is a predecessor to scale space representation and multiresolution analysis. The main assumption of this approach is that at a certain level of the image pyramid, a text line consists of a sequence of regularly placed components. However, the same text line at a higher (coarser) level looks like a long component. Hence, text is detected at the pyramid level where the granularity of the cells is "tuned" with the size of the text components. Loo and Tan in [45] proposed an improved approach based on irregular pyramids. The main advantage is that it does not require the detection of connected components. The main hypothesis of this approach is to view a text image as a combination of multiple irregular smaller regions. A word group is an aggregation of smaller regions holding fragments of a word and empty regions. The grouping process of lower-level regions into higher-level ones in the pyramid is determined by a concept of closeness. Thus, regions of the pyramid are merged in the next level if they contain word fragments with uniformity properties in area, mass, and density. The irregular pyramid structure contains the layout information of the entire document such that layout analysis is straightforward by navigating the pyramid levels. An advantage of this algorithm is that it is able to extract word groups with varying sizes, fonts,

alignments, and orientations. However, when text and graphics components touch, this approach induces wrong detection results.

### Methods Based on Signal Processing

Recently, Hoang and Tabbone [29] proposed a novel approach inspired in the techniques used in the signal processing field. In this approach, the image is seen as a bidimensional signal built as a mixture of two separated bidimensional signals of the same size (text and graphics) but containing morphologically different features. They propose a text-graphics separation method based on sparse representations. Sparse representations are compact representations that account for most or all information of a signal with a linear combination of a small number of elementary signals called atoms. Popular transforms are the discrete Fourier transform, the wavelet transform, and the singular value decomposition. The general problem of finding a representation with the smallest number of atoms from an arbitrary dictionary has been shown to be NP-hard.

In the approach presented in [29], the two signals, namely, text and non-text, are represented in terms of two discriminative dictionaries based on undecimated wavelet transform and curvelet transform. Morphological component analysis is employed for the separation of sparse representation of text and graphics components in these two dictionaries. Once text image is obtained, the final text string localization step follows a classical approach of connected component grouping in terms of heuristic spatial properties. The main advantage of this method is that it overcomes the problem of text touching graphics, showing an improved performance regarding the classical state-of-the-art approaches.

There exist some methods in the literature that use texture segmentation techniques. The basic assumption underlying these approaches is that the text and the graphics parts are considered as two different textured regions. Hence, the segmentation is seen as a binary text/non-text classification problem. The analysis is usually done in the frequential domain using filters that have maximum responses at some frequencies corresponding to text. Examples are Gabor filters [31], wavelet transform [2], or directional filters [34]. These approaches usually assume that documents are structured in a Manhattan layout, where text is organized in columns, paragraphs, and lines, and graphics are contained in figures inserted in the text parts. Since this is a layout segmentation problem rather than text detection in graphical documents, the reader is referred to Chaps. ▸5 (Page Segmentation Techniques in Document Analysis) and ▸8 (Text Segmentation for Document Recognition) for further details.

### Ad hoc Methods

There are some types of documents like forms, tables, or musical scores that due to their particular layout have some text separation rules specifically designed. The graphical parts usually consist in horizontal or vertical lines.

Common methods under this category are those for ruling-lines separation from handwritten text in forms [1, 6] or staff removal in musical scores [10]. These methods assume that lines have an horizontal configuration. The detection is

performed with projection profiles, RLSA, or line following algorithms. For further details, the reader is referred to Chaps. ▸19 (Recognition of Tables and Forms) and ▸22 (Analysis and Recognition of Music Scores).

## Summary

In the above subsections, the most relevant methods for the segmentation of text strings in graphical documents have been reviewed. A comparative summary of the different categories in terms of the main desired features is provided in Table 15.2. The categories of methods have been evaluated depending on the ability of segmenting text touching the graphical parts, if they are able to cope with different fonts and sizes, and the ability to detect multioriented text, even if it is not in a rectilinear layout.

The segmentation of text touching graphics is one of the main difficulties. In this classification, a method mets this feature if it is able to discriminate the text component at pixel level and is also able to reconstruct the segmented strokes. The baseline method based on grouping connected components [25] does not solve this problem. However, the improvement proposed by Tombre et al. [68] overcomes this problem proposing a method inspired by the vector-based approaches. These approaches achieve the best performances. The reason is that, since they work at skeleton level, they can reconstruct broken strokes from the bifurcation points of the skeleton. Since it is a problem that in the literature is addressed specifically, the solutions is further described in the next subsection.

The segmentation invariant to multiple fonts and sizes is solved working at different scales. It is implicitly done in methods that analyze pyramids of spatial bins at different scales (multiresolution methods) or methods that model the types of components in terms of frequencies (signal processing). The rest of the methods are able to detect text of different sizes but involving some thresholds defining the size of the structuring element, or the connected components or vectors that are grouped. Such thresholds are not necessarily set manually, but they can be inferred from the properties of the image being analyzed.

Concerning the property of multioriented text detection, the main difficulty is to cope with text that does not follow a rectilinear orientation but a curvilinear one. Typical examples are text strings appearing in maps that follow the orientation of an isoline, a river, or a road. Most of the methods cope with rectilinear orientations by ad hoc procedures like repeating the process at different orientations

**Table 15.2** Strong and weak points of text-graphics separation approaches

| | Text touching graphics | Multiple fonts and sizes | Multioriented text |
|---|---|---|---|
| Morphology analysis | − | + | −/+ |
| Connected components | + | + | −/+ |
| Vector-based segmentation | ++ | + | −/+ |
| Multiresolution analysis | − | ++ | + |
| Signal processing | + | ++ | + |

(morphology analysis) or using Hough-based methods when grouping components. The methods based on multiresolution or frequential analysis achieve in general better performance; however, they do not really define text strings as output artifacts. Roy et al. [55] recently proposed an approach addressing the problem of segmenting multioriented curvilinear text strings in graphical documents.

## The Problem of Text Touching Graphics

As stated in section "Text-Graphics Separation," one of the main difficulties of the separation of text and graphics is when text components are connected to graphical ones. Although some of the methods reviewed in the previous section can deal with text characters which touch the graphics, a better performance is achieved with specific methods. Such methods usually separate text from graphics by detecting touching lines.

When there is a priori knowledge about the structure of the graphical parts like long vertical and horizontal lines in forms, or the width of the lines, it can be modeled and ad hoc graphics detectors can be used. Otherwise, there are two main assumptions which are used. First, text characters connected to the graphics use to generate a local configuration of small strokes connected to a long line. Second, it is assumed that some isolated neighboring characters have already been detected and define a certain context.

Cao and Tan [5] base their method on the observation that the constituent strokes of characters are usually short segments in comparison with those of graphics. They work on the skeletons of large connected components. Using the properties of smooth continuation and line width, small strokes are grouped to reconstruct long segments underlying the region of intersection. Afterwards, the rest of the strokes connected to the reconstructed long line are grouped in candidate characters depending on their proximity and adjacency. Finally, character strings are extracted by merging connected components of a certain size likely to be isolated characters and the characters connected to graphics. Tombre et al. [68] proposed a similar approach. The main difference is that they assume that isolated characters have been previously segmented and grouped into strings. For each string, continuation zones are defined as prolongation of its ends. Then potential characters touching graphics are searched grouping connected short skeleton strokes included in these regions.

Song et al. [62] as part of a raster-to-vector conversion system proposed an improvement of the method of Dori and Liu [20]. The vectorization approach progressively simplifies the raster image by erasing recognized graphic objects to eliminate their interference with subsequent recognition. This iterative simplification allows to implicitly identify intersections between small vectors belonging to text characters and the erased graphical parts and gather nearby vectors in subsequent steps.

In the approach of Lu [46] mentioned in section "Methods Based on Morphology Analysis," the problem of text touching graphics is solved by an opening

morphological operation once text strings are detected in a given direction. It eliminates possibly existing connections but can provoke a side effect of breaking down connected components corresponding to single characters. A similar approach based on directional morphological filtering was proposed by Luo and Kasturi [47].

## Raster-to-Vector Conversion

Vectorization, also known as raster-to-vector conversion, has been one of the most studied problems since the beginnings of graphics recognition. The vectorization process can be defined as the automated conversion of scanned document images, having a pixel representation, into a set of vectors and other simple geometrical primitives. An example of the result of a raster-to-vector conversion algorithm is shown in Fig. 15.3. By representing graphic-rich documents, such as line drawings, in vectorial format its edition is easier and the storage much more efficient. In addition, vectorial data can be more suitable for further analysis than working directly with the raw pixel values in certain domains. Within the graphics recognition field, vectorization is a mature research topic. Many vectorization methods have been developed and implemented throughout the years and a number of software packages are available. Most of the methods provide good enough results to be used as input data for higher-level recognition and analysis methods. In that sense, raster-to-vector conversion seems to be an already solved problem. There are very few new scientific contributions to this specific topic, as vectorization methods have reached maturity and are assimilated by researchers as state of the art. However, all the methods present their specific weaknesses and, as stated by Tombre in [65], one cannot say that perfect raster-to-vector conversion is available. Several surveys that review the existing methods, presenting the strong and weak points of each of them, can be found in the literature, for example, Doermann's overview on the vectorization and segmentation problems presented in [14], Wenyin and Dori's survey of non-thinning vectorization methods [73], or Tombre et al.'s discussion on which are the most stable and robust vectorization methods presented in [67].

Focusing only on line features, the vectorization processes can be decomposed in three consecutive steps. The first step is to find the straight lines in the original raster image. The next step is to approximate these found lines into a set of vectors. After this approximation, a post-processing step is often required in order to merge some vectors, find more accurately the junction positions, etc. Each of these three steps will be overviewed in sections "Finding the Lines," "Polygonal Approximation," and "Post-processing and Contextual Information," respectively.

However, in order to be useful for higher-level interpretation phases, a raster-to-vector process should not be limited to the recognition of straight lines and should be able to deliver other geometric primitives. One of the most prolific research areas

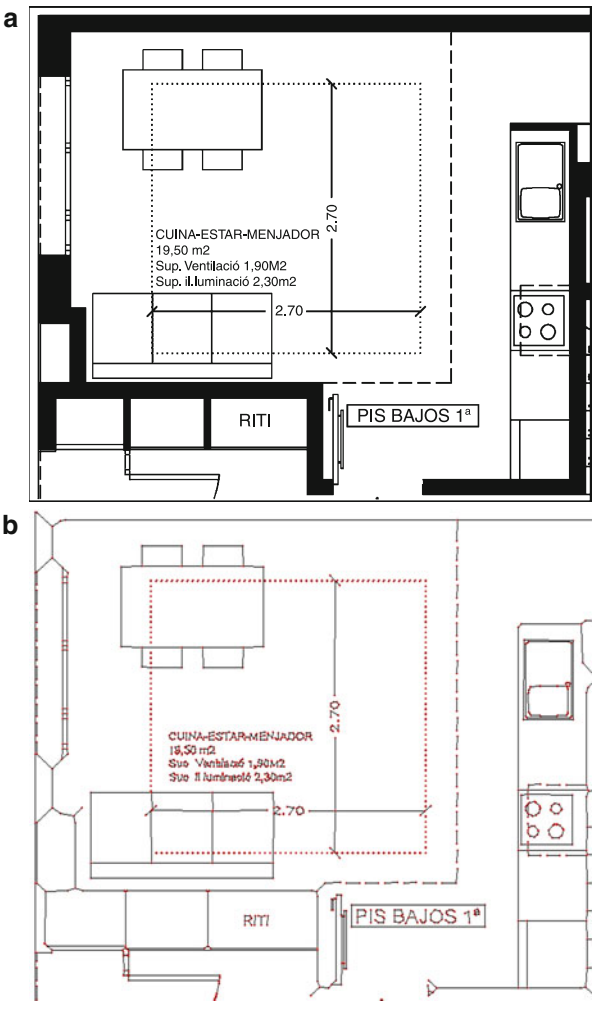


**Fig. 15.3** Raster-to-vector conversion example. (**a**) Original drawing, (**b**) vectorial representation

within the vectorization problem is the circular arc detection. The state of the art in arc detection will be overviewed in section "Arcs and Other Primitives."

## Finding the Lines

The first step in any raster-to-vector scheme is to extract a set of lines, i.e., chain of pixels, from the raster image to be processed.

Historically, the state-of-the-art methods have been categorized in several families depending on how they perform the line finding step. Roughly one can summarize these families into the following taxonomy:

- **Skeleton-based approaches** encode the shapes' topology by reducing them to their skeleton.
- **Contour matching methods** extract the vectors by tracking pairs of points from parallel edges.
- **Line-fitting methods** use a parametric line model to detect the lines that are present in the image.
- **Sparse-pixel approaches** avoid having to examine all the pixels in the image by using subsampling methods that give a broader view of the line.

### Skeleton-Based Methods

Computing the skeleton of the shapes from the raster image is the most common approach for vectorization [32, 36].

There are two well-known paradigms for extracting the skeleton from an image that are commonly used for vectorization purposes. The first one is based on an iterative thinning of the original image by a boundary erosion process [40]. The iterative process removes pixels until a unit-wide pixel chain remains. It is usually known as using a "peeling an onion" approach. On the other hand, the skeleton of an image can be computed by identifying the ridge lines formed by the centers of all maximal disks included in the original shape. For this second approach, usually some distance transform [51] is used for efficiency. As studied by Tombre and Tabbone in [66], iterative approaches need multiple passes before reaching the final result, so that computational time might be too high. In addition, they are sensitive to noise. On the other hand, distance-based approaches provide their results efficiently (only two passes are needed) and yield stable skeletons. It is therefore the preferred choice for the skeleton-based methods, used in many works as for instance in [28, 38].

Skeleton-based methods yield good precision regarding the positioning of the line. However, they tend to give lots of barbs when the image is irregular; therefore, post-processing steps to prune these barbs are needed. In addition, skeleton-based methods are weak with respect to the correct restitution of the junction and extremity points, compared with the position the draftsman wanted to be. This effect can be appreciated by looking at Fig. 15.4b. This drawback is due because skeletons follow the centers of maximal disks within the shape, whereas the correct junction or extremity position is not on those centers. Recovering the original line width is straightforward when using distance-based approaches, but it is not so when using iterative ones.

### Pairwise Contour Matching Methods

Contour matching approaches extract the vectors by tracking pairs of points from parallel edges. Once the contours are extracted from the raster image, parallel edges are matched together. From a pair of parallel contours, the medial line is generated and taken as the chain of pixels that represent the straight lines in the drawing. A final step that extends the medial lines until the intersections are found is performed in most of the cases. Several examples of raster-to-vector methods that follow this approach can be found in the literature [3, 24, 27].

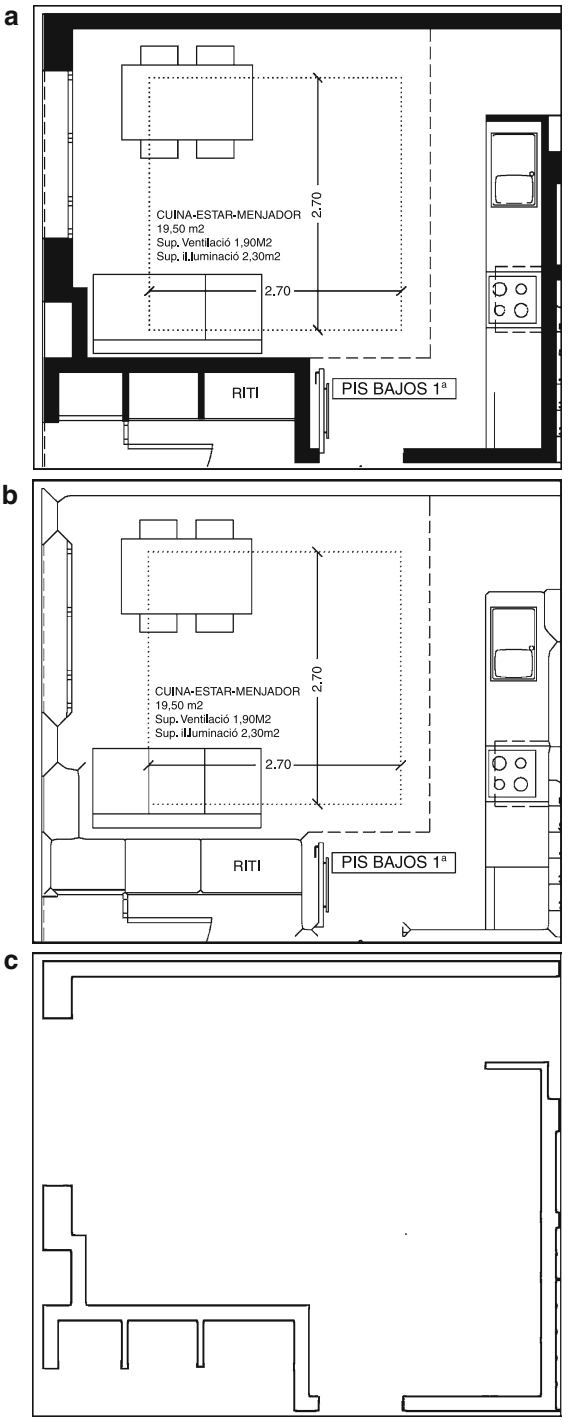


**Fig. 15.4** (continued)

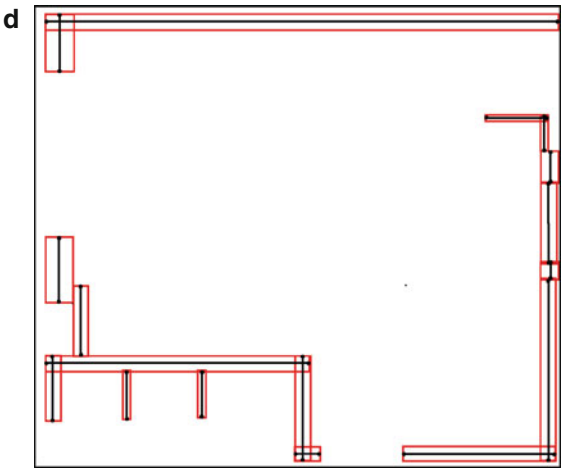


**Fig. 15.4** Line finding examples. (**a**) Original image, (**b**) its skeleton, (**c**) contours of thick elements, (**d**) result of the pairwise contour matching

The methods based on matching parallel contours are better at positioning the junction points than the skeleton-based ones. In addition, it is simple to add a line width recovery step, in order that the lines in the vectorial result have the original widths. However, the main drawback of these approaches comes from the difficulty to handle one-to-many and many-to-many contour matches when the original drawing presents complex patterns. See example in Fig. 15.4c,d.

In order to get the best of both approaches, some authors as Hori and Tanigawa [30] or Shimotsuji et al. [60] propose hybrid methods that use a combination of the skeleton and the contours in order to find the lines from the raster images.

### Line-Fitting Approaches

Line-fitting approaches consist in using a parametric line model to detect the lines present in the image. The most common and well-known technique to do so is the Hough transform. The main advantage of such methods is that the polygonal approximation step is avoided since the Hough transform already delivers vectorial data. However, very few attempts using the Hough transform have been proposed. Some examples are the following works [16, 61]. The main disadvantage of Hough-based methods is its memory inefficiency. In addition, since the parameter space is discretely sampled, the localization accuracy might be hindered.

### Sparse-Pixel Approaches

A number of sparse-pixel approaches have also been proposed. The general idea is to avoid having to examine all the pixels in the image, by using appropriate subsampling methods which give a broader view of the line. Examples are the orthogonal zigzag method presented by Dori and Wenyin in [22], the sparse-pixel tracking approach by Wenyin and Dori in [72], or the mesh-based approaches by Lin et al. [44] and by Vaxivière and Tombre [69]. The sparse-pixel approaches are

time-efficient due to the sparse sampling of the image data; however, their main limitation is that they are prone to double detections.

### Other Approaches

Other approaches that do not fall into any of the above families can also be found in the state of the art.

Some algorithms base the line detection on a run-based encoding of the raster image and then an analysis step of such runs. Examples of this technique are the following works [4, 13, 50]. The main problem of such techniques is again the inaccuracy to detect junction points, the sensitivity to noisy data, and the dependence on the scanning direction.

Other line extraction techniques are based on directly tracking lines in the original image itself. For example, Fukada presented in [26] a method using a line sensor that follows a line by keeping a tracking window perpendicular to the line direction. Or, in [59], Shih and Kasturi present a method which encodes the image in terms of maximum black squares. The lines are then extracted by tracking the centers of aligned maximum squares.

### Summary

Summarizing, skeleton-based methods are good at positioning the lines with minimum displacement from the original (localization) but are sensitive to noise (false-positive lines are detected) and are weak with respect to the correct restitution of junction and extremity points.

Contour matching methods perform well at both localizing and detecting the correct pixel chains but rely on complex matching schemes that can derive in detecting two lines where there is only one (multiple response). They perform well at the junction positioning and are able to recover the original line width.

Sparse-pixel approaches work well regarding localization but tend to miss small details and are prone to double detections. They also tend to provide junction points where there are none.

Finally, Hough-based methods are too dependent on the sampling of the parameter space provoking line merging and localization errors. They perform nonetheless quite good at detection of true lines and junction points.

All these strengths and weaknesses for each of the line finding families described above are presented in Table 15.3 (inspired by the summary by Tombre et al. [67]).

**Table 15.3** Strong and weak points of line-finding approaches

| | Skeleton | Contour | Sparse | Hough |
|---|---|---|---|---|
| Localization | ++ | + | + | +/− |
| Detection | −− | + | +/− | +/− |
| Single response | ++ | − | − | − |
| Junctions | − | + | − | + |
| Width | +/− | + | + | − |

## Polygonal Approximation

The chain of pixels extracted from the previous step have to be represented by a set of vectors. Many methods exist to perform this polygonal approximation step. The interested reader is referred to Rosin's paper on benchmarking of polygonal approximation algorithms [53].

Among the most common approaches that are used for the specific purposes to build a raster-to-vector conversion system, the method proposed by Rosin and West [54] has been often the preferred one [23, 67].

The method by Rosin and West [54] is based on a recursive split-and-merge technique. The principle is to recursively split the pixel chain into smaller and smaller segments, until the maximum deviation is zero or there are three or less points left. When performing this splitting phase, a tree structure is built. Then, a merging step traverses the segment tree upwards by keeping those segments that maximize a measure of significance. This measure of significance is defined as the ratio between the maximum deviation and the segment's length. Rosin and West's method tends to split up the lines around junction points into many small segments but has the advantage that is nonparametric and thus very generic.

Following the example of Fig. 15.5, a list $L_{ij}$ of skeleton pixels is hypothesized as being a straight line passing through its end points $P_i$ and $P_j$, the point $P_n$ corresponding at the point of maximum deviation $d_{ij}$ to the straight line segments the list $L_{ij}$ in two lists $L_{in}$ and $L_{nj}$, and the process is repeated recursively on each of the two lists. The recursive process is stopped when a line segment is smaller than four pixels long or the deviation is less than three pixels. The result of the recursive process is a multilevel tree where the description of the list of skeleton pixels at each level is a finer approximation of the level above. The tree is then traversed back up to the root. At each level, if any of the line segments passed up from the lower level are more significant than the line segment at the current level, they are retained and passed up to the next higher level as candidate line segments. If this is not the case, the line segment at the current level is passed up. In Fig. 15.5, an example of the first steps of the algorithm in approximating a curve by a set of straight lines is presented.

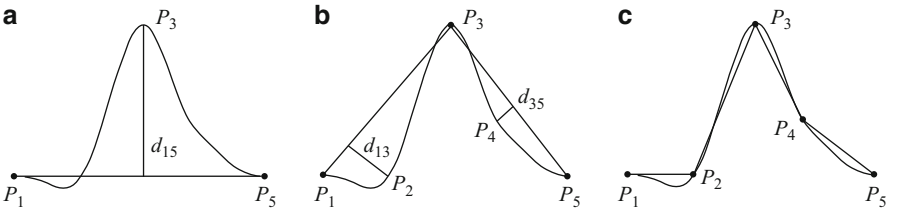


**Fig. 15.5** Three levels of the straight line approximation. (**a**) First iteration; (**b**) second iteration; (**c**) third iteration

### Post-processing and Contextual Information

After the polygonal approximation, a post-processing step is often required in order to merge some vectors, prune small barbs, find more accurately the junction positions, etc. Since these low-level post-processing steps might enhance significantly the quality of the vector description, almost all the authors propose their own post-processing. Some examples can be found in [7, 28, 32].

When the documents to vectorize come from a very specific domain, it makes sense to propose ad hoc post-processing steps that model the domain-specific constraints. Until now the presented steps of line finding and polygonal approximation do not use any specific knowledge about how vectors are supposed to be in the original drawings. The addition of contextual knowledge favors a given method for a specific application area whereas it hinders its generality. Some examples are the work of Chhabra et al. [8] aimed to process drawings from a telephone company, the system from Dosch et al. [23] processing architectural drawings, or the work of Lee et al. [42] where they use a knowledge base to refine the vectorization of cartographic maps.

### Arcs and Other Primitives

Until now the vectorization methods were just focused on dealing with straight lines; however, in order to be useful for higher-level interpretation phases, a raster-to-vector process should not be limited to the recognition of segments and should be able to deliver other geometric primitives.

In the literature, several raster-to-vector methods that deliver other geometric primitives such as dashed lines [7, 36] or halftone dots [37] can be found. However, the most prolific research area related to the vectorization problem is the circular arc detection.

According to Dori and Wenyin [21], the arc detection methods can be separated in two families. The first family is based on the circular Hough transform and directly works on the original pixels of the image. An example of such arc detection is presented in Fig. 15.6. Some examples are the works presented in [9, 12, 52]. The main difficulty with these approaches is that they often are of considerable complexity and quite sensitive to the image quality. On the other hand, the second family of methods work on the pixel chains or the resulting segments from the polygonal approximation in order to estimate their edge curvature and to fit an arc model to these digital curves. Some examples of this family are presented in [41,54,63]. The main inconvenient of such methods is that they rely on a previous step of line extraction and polygonal approximation instead of working with the original bitmap. Potential errors can be introduced in these preliminary stages and thus the accuracy and reliability of the system might be damaged.

Finally, it is worth to mention that for more than a decade, arc segmentation contests have been held in ICPR conferences and GREC workshops. The reports of

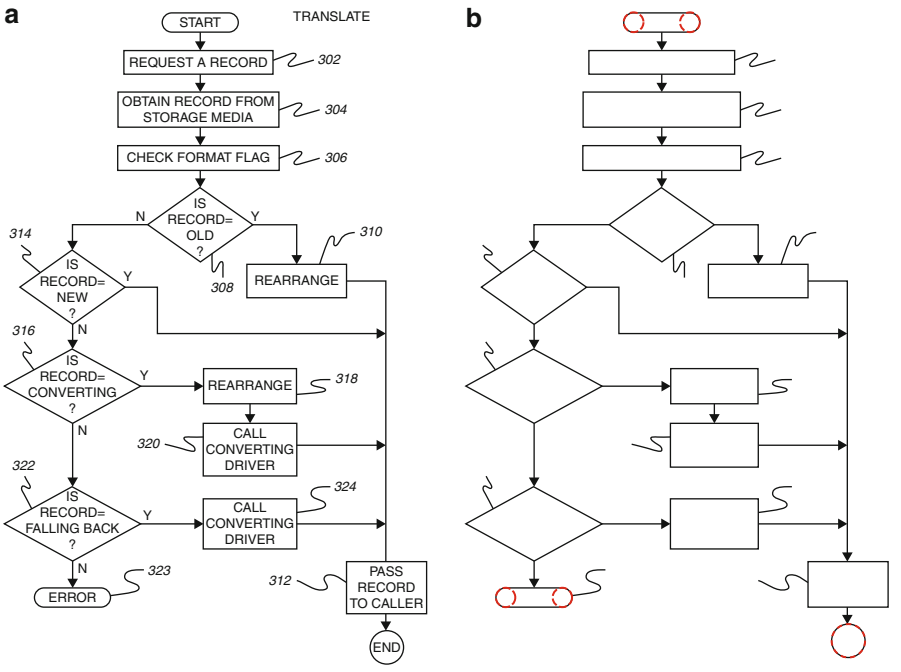


**Fig. 15.6** Arc detection example. (**a**) Original drawing, (**b**) arc detection by circular Hough transform after separating text from graphics and computing the skeleton of the original image

these contents (cf. [58] as an example) provide a clear insight on the evolution of the state of the art in arc segmentation and the remaining challenges.

## Recognition of Dimensions in Technical Drawings

Dimensions are graphical entities appearing in technical documents, usually engineering and architectural drawings. Since this kind of graphical documents is drawn to fit some ANSI or ISO drafting specifications, the dimension sets have a particular structure and follow standard rules. Dimension sets consist of several elements (see Fig. 15.7), namely, arrowheads and tails, witness lines, and text components. In the literature, several system examples dealing with the recognition of dimensioning sets [11, 15, 39] can be found. Usually, the systems dealing with dimensions recognition follow three main steps.

The first step deals with the separation of text from graphics. Usually, state-of-the-art techniques as the ones reviewed in section "Text-Graphics Separation" are used in this step. However, some works that propose text detection methods for the specific case of dimensions can be found, as the one presented by Dori and Velkovitch in [20].

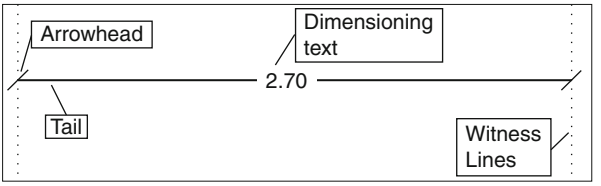


**Fig. 15.7** Dimensions example

Once the text and the graphic elements are separated, the next step is focused on the detection of the dimensioning lines. In order to properly detect these lines, several works that propose to first detect and recognize the arrowheads can be identified. The arrowheads are used as anchor points to then extract the complete witness lines. Instead of using an off-the-shelf symbol recognition method aimed at locating these arrowheads, usually an ad hoc designed arrowhead model is preferred. Some examples of the arrowhead recognition and localization are presented in [39,43,71]. A final association of the text with the extracted dimensioning lines is performed. This association is not always straightforward since for every consecutive pair of arrowheads several situations are possible, whether two arrowheads correspond to a dimension set or not and have or do not have some text lying in between.

Since the dimension sets follow some standard rules, a final syntactic validation step is often used. Bidimensional grammars are used to specify how all the possible dimension sets can be generated and describe all the various possible dimension types. Web grammars have been commonly used for this purpose by many authors [15,19,49].

## Extraction of Texture-Based Graphical Primitives

Some graphical documents contain primitives characterized by a repetitive pattern. For example, isolines representing points of equal value in geographical maps or some dimensions in technical drawings are drawn using dotted lines. On the other hand, technical drawings use to have regions filled by hatched patterns (e.g., walls in architectural drawings or parcels in cadastral maps).

In the recognition of repetitive patterns, Perceptual Organization (PO) plays an important role. PO operates at the intermediate vision level to identify some particular emergent patterns that are used in end tasks as indexing and recognition. Saund et al. [57] studied the application of the PO framework to the field of graphics recognition. In this work, they noticed that PO allows to extract salient patterns with weak prior models and also to represent them compactly making explicit many of their features. In graphical documents, signatures formulated in terms of *proximity*, *continuation*, and *parallelism* of primitives are the basis for the detection of repetitive patterns.

One-dimensional patterns usually consist of dashed or dashed-dotted lines. Human perception gives the ability to "fill the gaps" between the small segments and hence perceive the dashed line as a unique entity. Machines simulate this perceptual

cue by a sequential stepwise recovery of components that meet certain continuity conditions.

Bidimensional patterns consist in structured textures, like hatching or tiling, filling some areas of the drawing. A structured texture can be informally described as a set of primitives regularly arranged following some placement rules. Two major approaches can be used to recognize these structures, namely, Hough-based and syntactic methods.

Classical Hough transform maps each image point $(x, y)$ into a set of points $(\theta, \rho)$ that fulfill the equation $\rho = x \cos\theta + y \sin\theta$. Detection of peaks in the parameter space constitutes a powerful method to detect straight lines in the input image. A configuration of several straight segments results in a set of corresponding peaks in the Hough space. These peaks have a regular arrangement when the original segments form a hatched or tiled pattern. As can be seen in Fig. 15.8a, a hatched pattern generates a set of aligned peaks at regular step in the Hough space. It is straightforward to detect it with a double Hough transform, i.e., detecting straight lines in the Hough space. The principle that parallel lines in the image generate aligned peaks in the Hough space can be extended to tiled patterns. Fig. 15.8b,c show the Hough transform for two types of tiled patterns. In both cases, lines of peaks appear in the Hough space. In addition, since tiled patterns consist of regular polygons, the lines of peaks in the Hough space are parallel and with an interline distance corresponding to the inner angles of the polygons.

Grammars are a suitable tool to represent repetitive structures in graphic documents, either linear structures, as dashed lines [33], or bidimensional structures, as tiled patterns [48, 56]. Among the different types of syntactic models existing in the literature, graph grammars are the most common one used in graphic documents. Informally speaking, a graph grammar models patterns by a set of rules hierarchically describing patterns in terms of subpatterns. Primitive patterns, called terminal vocabulary in the terminology of grammars, are represented by graphs corresponding to the lines of the document. A grammar is useful to describe repetitive primitives, as hatching or tiling patterns, or syntactic rules that a diagrammatic notation involves, as for dimension symbols or musical scores. The recognition process is hence carried on by a parser. Grammars are complex models supported in a robust theory (syntactic pattern recognition). This is out of the scope of this chapter. The reader is referred to [56] for a further description.

## Executive Summary

Graphics recognition concerns on the analysis of the non-textual entities of document images (tables, symbols, line drawings, etc.). It has experienced a growing interest since the 1980s when there was an industrial need for converting raster images of engineering designs and maps to CAD and GIS platforms for subsequent edition, printing, and reflowing. Specific graphics recognition techniques were addressed for the low-mid level stages of the whole interpretation pipeline. In this chapter, the relevant approaches to convert raster images into basic primitives

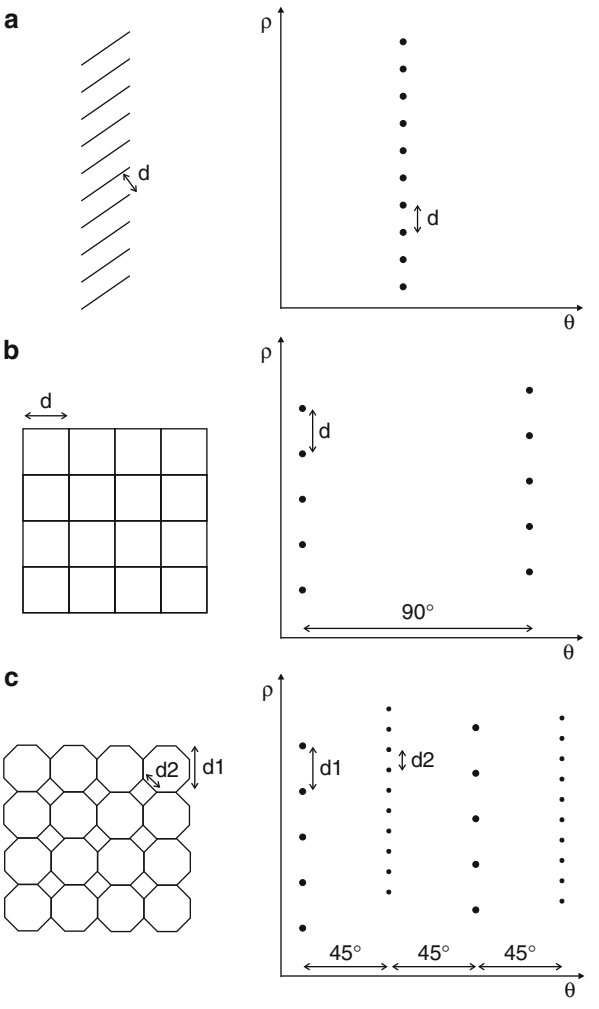


**Fig. 15.8** Hough space configuration for (**a**) hatched patterns, (**b**) (**c**) tiled patterns

that compound meaningful objects have been described. Such basic primitives are basically vectors (analytic descriptions of lines of pixels as segments, arcs or higher order curves), and text labels (usually associated to dimensions, object labels, georeferences or toponyms). Therefore, the two main operations that are the core of the graphics recognition techniques are text-graphics separation and vectorization.

Having a look to the literature, both text-graphics separation and vectorization can be considered mature techniques from a scientific point of view. Commercial software includes them at reasonable levels of performance. Hence, one can not expect groundbreaking scientific innovations in the future. The challenges have to be foreseen in a more global view, having vectorization and text-graphics separation

modules integrated in systems and platforms for massive processing and integral interpretation of documents.

## Conclusion

In this chapter, the state of the art in the techniques addressed to extract the compounding units of graphical documents has been reviewed. The needs of the market have been the leading force behind a huge amount of research and development in graphics recognition techniques. Hence, there was an industrial need of converting engineering drawings and maps to CAD and GIS platforms that originated a growing applied research activity in the 1990s on vectorization and text detection in graphical documents. A community on graphics recognition has been consolidating since then including other challenges such as symbol recognition and document interpretation, topics that are reviewed in the following chapters.

Vectorization, or raster-to-vector conversion, is the problem that has more mature solutions. There is still the open question of what vectorization has to be, but this might be a rhetoric question that will never be answered. Vectorization is not a stand-alone functionality, but it is integrated in bigger systems. Hence, the desired performance of a vectorization process depends on the purpose of the entire system. If vectorization is an intermediate step in the process of recognition, it is not needed that vectors perfectly fit into the lines of pixels, but it should be stable in terms of invariance to variations of shapes. On the other hand, when the purpose of the system is just converting to a DXF or SVG format compatible with a CAD system for reprinting or storing the document, then it is better a higher number of primitives to get a perfect fitting into the original lines. It leads to vectorization algorithms with different parameters where the user or the document conditions determine the settings. One of the current challenges of vectorization is to develop systems that learn the configuration of the parameters in terms of the user feedback in previous documents or from the features of the document contents.

Currently, engineering drawings are digitally born documents, so the classical need of vectorization for the sake of conversion from raster to CAD format is quite obsolete. However, there is a latent need for vectorization in novel applications appearing in the market. An example of this is CAD software for nonprofessional use where the user can scan a plan of his/her house to render a simple 3D view augmented with some furniture objects, changed colors of the walls, etc., for decoration purposes. Whatever is the application, vectorization is not usual to be a stand-alone process, but it is combined with the recognition of graphical entities.

The main objective of text-graphics separation is not only to detect the text strings in a graphical document, independently of the font size and the orientation, but to be able to reconstruct the characters touching graphical lines. The baseline methods are the ones of Fletcher and Kasturi [25] and Tombre et al. [68]. These methods are based on the iterative grouping of connected components having some spatial properties (similar size, alignment, regular inter-character distance).

The segmentation invariant to multiple fonts and sizes is solved working at different scales.

Nowadays, the problem of text-graphics separation is having a resurgence but in new applications as form processing in digital mail room platforms for the automation of incoming mail processing. Among the milliards of documents being processed, forms are a big subset. Forms have vertical and horizontal lines forming tables, text boxes, and rule lines. Classical text-graphics separation techniques cannot be applied to solve the segmentation of text in forms because it is usually handwritten. It opens new perspectives and reformulates to some extend the problem of text-graphics separation making closer the areas of graphics recognition and handwriting analysis.

Text-graphics separation is also of special relevance in maps. Not for the classical raster to GIS conversion, but for massive map processing, georeferencing, and retrieval maps using textual queries. In this case, although being a different application, the classical techniques reviewed in this chapter can be applied.

## Cross-References

▸Analysis and Interpretation of Graphical Documents
▸Analysis and Recognition of Music Scores
▸An Overview of Symbol Recognition
▸Page Segmentation Techniques in Document Analysis
▸Recognition of Tables and Forms
▸Text Segmentation for Document Recognition

## Notes

[1] GREC is the workshop of the technical committee on graphics recognition (TC10) of the International Association of Pattern Recognition (IAPR).
[2] http://www.qgar.org/

## References

1. Abd-Almageed W, Kumar J, Doermann D (2009) Page ruleline removal using linear subspaces in monochromatic handwritten arabic documents. In: Proceedings of the 12th international conference on document analysis and recognition, Barcelona, pp 768–772
2. Acharyya M, Kundu MK (2002) Document image segmentation using wavelet scale-space features. IEEE Trans Circuits Syst Video Technol 12(12):1117–1127
3. Antoine D, Collin S, Tombre K (1992) Analysis of technical documents: the REDRAW system. In: Structured document image analysis. Springer, Berlin, pp 385–402
4. Boatto L, Consorti V, Del Buono M, Di Zenzo S, Eramo V, Esposito A, Melcarne F, Meucci M, Morelli A, Mosciatti M, Scarci S, Tucci M (1992) An interpretation system for land register maps. Computer 25(7):25–33

5. Cao R, Tan CL (2002) Text/graphics separation in maps. In: Graphics recognition algorithms and applications. Lecture notes in computer science, vol 2390. Springer, Berlin/New York, pp 167–177
6. Chen J, Lopresti D, Kavallieratou E (2010) The impact of ruling lines on writer identification. In: Proceedings of the 2nd international conference on frontiers in handwriting recognition, Kolkata, pp 439–444
7. Chen Y, Langrana NA, Das AK (1996) Perfecting vectorized mechanical drawings. Comput Vis Image Underst 63(2):273–286
8. Chhabra AK, Misra V, Arias J (1996) Detection of horizontal lines in noisy run length encoded images: the FAST method. In: Graphics recognition methods and applications. Lecture notes in computer science, vol 1072. Springer, Berlin/Heidelberg, pp 35–48
9. Chiu SH, Liaw JJ (2005) An effective voting method for circle detection. Pattern Recognit Lett 26(2):121–133
10. Dalitz C, Droettboom M, Pranzas B, Fujinaga I (2008) A comparative study of staff removal algorithms. IEEE Trans Pattern Anal Mach Intell 30:753–766
11. Das AK, Langrana NA (1997) Recognition and integration of dimension sets in vectorized engineering drawings. Comput Vis Image Underst 68(1):90–108
12. Davies ER (1988) A modified Hough scheme for general circle location. Pattern Recognit Lett 7(1):37–43
13. Di Zenzo S, Cinque L, Levialdi S (1996) Run-based algorithms for binary image analysis and processing. IEEE Trans Pattern Anal Mach Intell 18(1):83–89
14. Doermann D (1998) An introduction to vectorization and segmentation. In: Graphics recognition algorithms and systems. Lecture notes in computer science, vol 1389. Springer, Berlin/New York, pp 1–8
15. Dori D (1992) Self-structural syntax-directed pattern recognition of dimensioning components in engineering drawings. In: Structured document image analysis. Springer, Berlin/New York, pp 359–384
16. Dori D (1997) Orthogonal zig-zag: an algorithm for vectorizing engineering drawings compared with Hough transform. Adv Eng Softw 28(1):11–24
17. Dori D, Liu W (1996) Vector-based segmentation of text connected to graphics in engineering drawings. In: Advances in structural and syntactical pattern recognition. Lecture notes in computer science, vol 1121. Springer, Berlin/New York, pp 322–331
18. Dori D, Liu W (1999) Automated CAD conversion with the machine drawing understanding system: concepts, algorithms, and performance. IEEE Trans Syst Man Cybern Part A: Syst Hum 29(4):411–416
19. Dori D, Pnuelli A (1988) The grammar of dimensions in machine drawings. Comput Vis Image Underst 42(1):1–18
20. Dori D, Velkovitch Y (1998) Segmentation and recognition of dimensioning text from engineering drawings. Comput Vis Image Underst 69(2):196–201
21. Dori D, Wenyin L (1998) Stepwise recovery of arc segmentation in complex line environments. Int J Doc Anal Recognit 1(1):62–71
22. Dori D, Wenyin L (1999) Sparse pixel vectorization, an algorithm and its performance evaluation. IEEE Trans Pattern Anal Mach Intell 21(3):202–215
23. Dosch P, Tombre K, Ah-Soon C, Masini G (2000) A complete system for the analysis of architectural drawings. Int J Doc Anal Recognit 3(2):102–116
24. Fan KC, Chen DF, Wen MG (1998) Skeletonization of binary images with nonuniform width via block decomposition and contour vector matching. Pattern Recognit 31(7): 823–838
25. Fletcher LA, Kasturi R (1988) A robust algorithm for text string separation from mixed text/graphics images. IEEE Trans Pattern Anal Mach Intell 10(6):910–918
26. Fukada Y (1984) A primary algorithm for the understanding of logic circuit diagrams. Pattern Recognit 17(1):125–134
27. Han CC, Fan KC (1994) Skeleton generation of engineering drawings via contour matching. Pattern Recognit 27(2):261–275

28. Hilaire X, Tombre K (2006) Robust and accurate vectorization of line drawings. IEEE Trans Pattern Anal Mach Intell 28(6):890–904
29. Hoang TV, Tabbone S (2010) Text extraction from graphical document images using sparse representation. In: Proceedings of the 9th IAPR international workshop on document analysis systems, Boston, pp 143–150
30. Hori O, Tanigawa S (1993) Raster-to-vector conversion by line fitting based on contours and skeletons. In: Proceedings of the 2nd international conference on document analysis and recognition, Tsukuba, pp 353–358
31. Jain A, Bhattacharjee S (1992) Text segmentation using gabor filters for automatic document processing. Mach Vis Appl 5(3):169–184
32. Janssen RDT, Vossepoel AM (1997) Adaptive vectorization of line drawing images. Comput Vis Image Underst 65(1):38–56
33. Jonk A, van den Boomgaard R, Smeulders A (1999) Grammatical inference of dashed lines. Comput Vis Image Underst 74(3):212–226
34. Journet N, Eglin V, Ramel JY, Mullot R (2005) Text/graphic labelling of ancient printed documents. In: Proceedings of the 8th international conference on document analysis and recognition, Seoul, pp 1010–1014
35. Kaneko T (1992) Line structure extraction from line-drawing images. Pattern Recognit 25(9):963–973
36. Kasturi R, Bow ST, El-Masri W, Shah J, Gattiker JR, Mokate UB (1990) A system for interpretation of line drawings. IEEE Trans Pattern Anal Mach Intell 12(10):978–992
37. Kawamura K, Watanabe H, Tominaga H (2004) Vector representation of binary images containing halftone dots. In: Proceedings of the IEEE international conference on multimedia and expo, Taipei, pp 335–338
38. Kolesnikov AN, Belekhov VV, Chalenko IO (1996) Vectorization of raster images. Pattern Recognit Image Anal 6(4):786–794
39. Lai CP, Kasturi R (1994) Detection of dimension sets in engineering drawings. IEEE Trans Pattern Anal Mach Intell 16(8):848–854
40. Lam L, Lee SW, Suen CY (1992) Thinning methodologies – a comprehensive survey. IEEE Trans Pattern Anal Mach Intell 14(9):869–885
41. Lamiroy B, Guebbas Y (2010) Robust and precise circular arc detection. In: Graphics recognition. Achievements, challenges and evolution. Lecture notes in computer science, vol 6020. Springer, Berlin/Heidelberg, pp 49–60
42. Lee KH, Cho SB, Choy YC (2000) Automated vectorization of cartographic maps by a knowledge-based system. Eng Appl Artif Intell 13(2):165–178
43. Lin SC, Ting CK (1997) A new approach for detection of dimensions set in mechanical drawings. Pattern Recognit Lett 18(4):367–373
44. Lin X, Shimotsuji S, Minoh M, Sakai T (1985) Efficient diagram understanding with characteristic pattern detection. Comput Vis Image Underst 30(1):84–106
45. Loo PK, Tan CL (2001) Detection of word groups based on irregular pyramid. In: Proceedings of the 6th international conference on document analysis and recognition, Seattle, pp 200–204
46. Lu Z (1998) Detection of text regions from digital engineering drawings. IEEE Trans Pattern Anal Mach Intell 20(4):431–439
47. Luo H, Kasturi R (1998) Improved directional morphological operations for separation of characters from maps/graphics. In: Graphics recognition algorithms and systems. Lecture notes in computer science, vol 1389. Springer, Berlin/New York, pp 35–47
48. Matsuyama T, Saburi K, Nagao M (1982) A structural analyzer for regularly arranged textures. Comput Graph Image Process 18:259–278
49. Min W, Tang Z, Tang L (1993) Using web grammar to recognize dimensions in engineering drawings. Pattern Recognit 26(9):1407–1416
50. Monagan G, Roosli M (1993) Appropriate base representation using a run graph. In: Proceedings of the 2nd international conference on document analysis and recognition, Tsukuba, pp 623–626
51. Niblack CW, Gibbons PB, Capson DW (1992) Generating skeletons and centerlines from the distance transform. CVGIP: Graph Models Image Process 54(5):420–437

52. Olson CF (1999) Constrained Hough transforms for curve detection. Comput Vis Image Underst 73(3):329–345
53. Rosin PL (2003) Assessing the behaviour of polygonal approximation algorithms. Pattern Recognit 36(2):505–518
54. Rosin PL, West GA (1989) Segmentation of edges into lines and arcs. Image Vis Comput 7(2):109–114
55. Roy PP, Pal U, Lladós J (2012) Text line extraction in graphical documents using background and foreground information. Int J Doc Anal Recognit 15(3):227–241
56. Sánchez G, Lladós J (2004) Syntactic models to represent perceptually regular repetitive patterns in graphic documents. In: Graphics recognition. Recent advances and perspectives. Lecture notes in computer science, vol 3088. Springer, Berlin/New York, pp 166–175
57. Saund E, Mahoney J, Fleet D, Larner D (2002) Perceptual organization as a foundation for graphics recognition. In: Graphics recognition: algorithms and applications. Springer, Berlin/New York, pp 139–147
58. Shafait F, Keysers D, Breuel TM (2008) GREC 2007 arc segmentation contest: evaluation of four participating algorithms. In: Graphics recognition. Recent advances and new opportunities. Lecture notes in computer science, vol 5046. Springer, Berlin/New York, pp 310–320
59. Shih CC, Kasturi R (1989) Extraction of graphic primitives from images of paper based line drawings. Mach Vis Appl 2(2):103–113
60. Shimotsuji S, Hori O, Asano M, Suzuki K, Hoshino F, Ishii T (1992) A robust recognition system for a drawing superimposed on a map. Computer 25(7):56–59
61. Song J, Lyu MR (2005) A Hough transform based line recognition method utilizing both parameter space and image space. Pattern Recognit 38(4):539–552
62. Song J, Su F, Tai CL, Cai S (2002) An object-oriented progressive-simplification-based vectorization system for engineering drawings: model, algorithm, and performance. IEEE Trans Pattern Anal Mach Intell 24:1048–1060
63. Song J, Lyu MR, Cai S (2004) Effective multiresolution arc segmentation: algorithms and performance evaluation. IEEE Trans Pattern Anal Mach Intell 26(11):1491–1506
64. Tan CL, Ng PO (1998) Text extraction using pyramid. Pattern Recognit 31(1):63–72
65. Tombre K (1998) Analysis of engineering drawings: state of the art and challenges. In: Graphics recognition algorithms and systems. Lecture notes in computer science, vol 1389. Springer, Berlin/New York, pp 257–264
66. Tombre K, Tabbone S (2000) Vectorization in graphics recognition: to thin or not to thin. In: Proceedings of the 15th international conference on pattern recognition, Barcelona, pp 91–96
67. Tombre K, Ah-Soon C, Dosch P, Massini G, Tabbone S (2000) Stable and robust vectorization: how to make the right choices. In: Graphics recognition recent advances. Lecture notes in computer science, vol 1941. Springer, Berlin/New York, pp 3–18
68. Tombre K, Tabbone S, Pelissier L, Lamiroy B, Dosch P (2002) Text/graphics separation revisited. In: Document analysis systems V. Lecture notes in computer science, vol 2423. Springer, Berlin/New York, pp 615–620
69. Vaxivière P, Tombre K (1994) Subsampling: a structural approach to technical document vectorization. In: Structure and pattern recognition. Proceedings of the IAPR Workshop on syntactic and structural pattern recognition, Haifa, Israel, pp 323–332
70. Wahl F, Wong K, Casey R (1982) Block segmentation and text extraction in mixed text/image documents. Comput Graph Image Process 20(4):375–390
71. Wendling L, Tabbone S (2004) A new way to detect arrows in line drawings. IEEE Trans Pattern Anal Mach Intell 26(7):935–941
72. Wenyin L, Dori D (1996) Sparse pixel tracking: a fast vectorization algorithm applied to engineering drawings. In: Proceedings of the 13th international conference on pattern recognition, Vienna, pp 808–812
73. Wenyin L, Dori D (1998) A survey of non-thinning based vectorization methods. In: Advances in pattern recognition. Lecture notes in computer science, vol 1451. Springer, Berlin/New York, pp 230–241

## Further Reading

For further reading, the reader is addressed to the classic literature contributions of the field. The key references in text-graphics separation are the work of Fletcher and Kasturi [25] and the subsequent improvements proposed by Tombre et al. [68]. Concerning the problem of raster-to-vector conversion, recommended readings are the surveys and analysis papers of the QGAR group led by K. Tombre [66, 67]. Practitioners interested in the implementation of the methods can download the open source QGAR software library.[2] Finally, to be kept informed of the progress on graphics recognition techniques and the area at large, the reader is referred to the publications arising from the IAPR graphics recognition workshop (GREC series), held every two years since 1995, and the post workshop book with selected papers that is published by *Springer* in the *Lecture Notes in Computer Science* series.