Czech Technical University, Faculty of Electrical Engineering
Center for Machine Perception, Computer Vision Laboratory
121 35 Praha 2, Karlovo náměstí 13, Czech Republic

Research Report

No. CTU-CMP-1996-02

Classification of lines and areas in documents – thesis proposal

Jan Müller

September 5, 1996

Contents

1 Introduction 2
- 1.1 What is Document Understanding about 2
- 1.2 Our aim 3

2 Motivation 6

3 Basic Concepts 8
- 3.1 Skeleton 8
- 3.2 Boundary 9
- 3.3 Grammar 10
- 3.4 Cellular complex 14
- 3.5 Voronoi diagram 17

4 Formulation of the research goal 20
- 4.1 What is to be done? 20
- 4.2 Low level processing 20
- 4.3 How to do it? 21
- 4.4 Example of description 23

5 Related work 24
- 5.1 The theoretical background 24
  - 5.1.1 Observation 24
  - 5.1.2 Mathematical morphology 24
  - 5.1.3 Relaxation and clustering 25
  - 5.1.4 Knowledge based approaches 26
  - 5.1.5 Grammars 27
- 5.2 Use of knowledge 27
- 5.3 Interpretation strategy 28
  - 5.3.1 Top down 28
  - 5.3.2 Bottom up 28
  - 5.3.3 Combined 28

6 Thesis plan 29

Chapter 1

Introduction

The automatic classification of lines and areas in scanned documents might be an important step towards automatic document interpretation. My research tackles this problem. This research proposal identifies my research topic and relates it to work of others. A short introduction to the problems covered by the *document understanding* research is presented in Chapter 1. The motivation for further research in document understanding is presented in Chapter 2. Concepts are defined in Chapter 3 and a new approach is described in detail in Chapter 4. Existing works about line and area recognition are presented in Chapter 5. The proposed approach is summarized, some expected results are described and a plan for a dissertation thesis is outlined in Chapter 6.

1.1 What is Document Understanding about

Document understanding is a part of image analysis research. The technology aims to automate the acquisition of data from scanned paper documents. The field of interest can be divided into two classes: (1) processing of text documents and (2) processing of graphical documents. The boundary between these classes is not sharp. Most documents contain a mixture of text and graphics.

Text documents, books, articles, newspapers, typed sheets and handwritten documents have been analysed for a very long time. The main interest is the recognition of handwritten addresses on parcels and postcards, and the recognition of printed articles in books and newspapers. The main field is *optical character recognition* (OCR). This is recognition of printed and handwritten letters. Many works have been published about this subject [MSY92, RP94, Nis95]. Other fields are the extraction of logical flow structure and separation of text from graphics in mixed documents [YYSH94, Men91].

Graphical documents, engineering drawings and maps of different scales are in focus today. Graphical documents are more complex than text documents and their processing is more difficult. There are also special symbols that are present in one kind of graphical document and not in the others, due to different semantics [AK94].

There are names of parts, annotations, and dimension sets in CAD engineering drawings. There are names of cities, towns, and rivers as well as many signs in the maps. For example, dimension sets in CAD documents consist of lines, arrows, special marks (i.e. diameter ø, angle ∠, square □) and numerical values. The separation of dimension sets and their recognition is very important for further interpretation and the correction of badly recognized CAD drawings. There are many publications on this subject [LK94, CC94]. But dimension sets are not the only important components in graphical documents. Text and other symbols are separated from the graphical document. These are recognized by a separate OCR program. The result is provided for further interpretation [Men91, Be92].

Next special objects are dotted, dashed, dashdotted, and many other structured lines created by repetition of some basic element [CLD96, WB86]. These elements are even more complex in maps. There are hatched areas in engineering drawings and maps. There are only a few simple hatching patterns (vertical, horizontal, cross lines) used in the CAD drawings. More complex patterns (trees, bushes, circles) are used in maps and their recognition is also more complicated.

1.2 Our aim

We propose an approach that can describe symbols and their distribution along a general line or grid. Our method might be able to describe and classify a wide variety of structured lines and hatching patterns (see figures 1.1 and 1.2).

We will be able to describe both connected and unconnected entities. Basic elements will be described in terms of skeletal and boundary parts. An attribute context-free grammar will describe the relation between the elements. A Voronoi diagram will be constructed to describe the unconnected objects. The Voronoi diagram will be described in terms of cellular complex. Acceptable relations between regions, edges and vertices will be described by a context-free attribute grammar. The parser (procedure that evaluates the grammar) will be common for both the description of the elements and the description of the distribution of the elements.

We expect that this description will be general enough to describe all patterns in graphical documents.

Figure 1.1: Examples of lines on the maps.

Figure 1.2: Examples of hatching on the map and engineering drawing.

Chapter 2

Motivation

The reasons that led us to work on the subject are sketched in this chapter.

Many people think that document understanding is going out of business as computers are penetrating offices and design facilities. Editors and CAD programs are often used to create documents. The reality is totally different at present. The majority of information is still in the form of paper documents. Paper is often used as an universal interface between different systems. Automated transfer of data to the computer can bring more power to the users in work with the documents. A brief summary of how many engineering documents are kept on paper and in electronic form can be found in [KHD95]. A short review follows.

- There are approximately 3.5 billion engineering drawings of various types in the USA and Canada, with about 26 million new ones added each year.
- Only 13% of the existing, active drawings are available in CAD form. In 1990, about 20% of the drawings were created in CAD and about 25% were CAD revisions of older CAD drawings. The remaining 55% were done on the drawing board, using traditional pen-and-pencil drafting techniques.
- Once a drawing is in CAD format, all the advantages of database storage, retrieval and query become available.

A similar or even worse situation is in the processing of cartographic, cadastral and utility maps. The automated transfer of the paper maps into Geographical Information Systems (GIS) can bring the opportunity for better planning.

There are only a few articles about the recognition of structured lines and hatch patterns in the literature. The articles present only special approaches for the recognition of one kind of line or hatching in one kind of document. For example, there are articles about the recognition of straight dashed or center lines in CAD drawings or articles about the recognition of line-hatched regions in maps [SEB91].

The proposed approach will describe both lines of any kind (straight or curved) and hatched patterns of any distribution and hatching pattern by a single representation of the input data. The proposed algorithm for line and area classification is general, easy and easily extendable.

Lectures "Theoretical aspects of Pattern Recognition" [Sch96] held by professor M. Schlezinger in the summer term 1996 had a big influence on the selecting of the method. Talks with him about the recognition of documents were very useful for finding the theme of my thesis and possible ways of solution.

Surprisingly, the algorithm that checks the correspondence of the input data to the grammar rules is computationally simpler in the 2D than in the 1D case (see page 12).

And the last but not the least reason why I have chosen this topic is that I love maps.

Chapter 3

Basic Concepts

Concepts such as skeleton, contour, Voronoi diagram, 2D grammar and cellular complex are defined in this chapter. These concepts are used for the description of the algorithms and theories in the rest of the paper.

3.1 Skeleton

Definition 3.1.1 *Let the skeleton $S(Y)$ of the connected set Y be the union of the centers x of the circles $D(x)$, where $D(x) \subset Y$ and the circle touches the boundary of the set Y in at least two points [vHB93].*

Definition 3.1.2 *Let K denote the basic elementary structuring element with respect to which we will determine the skeleton. K morphologically homothetic shapes of size n are easily generated by dilating $\{0\}$ with K n times: $\oplus_n K$. The only requirements for K is that K contains the origin and is bounded. The skeleton is denoted by $MA(A, K)$ and is defined by*

$$MA(A,K) = \{x | x \in K_y \ \& \ (\oplus_{d[A,K](y)-1} K)_y \supseteq (\oplus_{d[A,K](x)-1} K)_x \Rightarrow x = y\}$$

where $d[A, K]$ is the generalized distance transform of A with respect to K [HS92].

$$\begin{aligned} d[A,K] &= max\{m | x \in A \ominus_{m-1} K\} \\ &= max\{m | x \in A \ominus (\oplus_{m-1} K)\} \\ &= max\{m | (\oplus_{m-1} K)_x \subseteq A\} \end{aligned}$$

In other words, the skeleton of a set A with respect to a structuring element K contains the centers of all K-homothetic sets that are maximal with respect to the property of containment.

The skeleton has following properties:

- preserves the connectivity of the objects,

- is uniformly 1 pixel thick,
- represents the medial axis of the region.

It is a good representation of the shape of elongated objects that are relatively thin.

There are many algorithms that can create skeletons according to different definitions. We will describe only a few for illustration of different approaches.

The most popular approach is *morphological thinning.* The process is called ultimate erosion. Erosion by the given structuring element is performed if the region connectivity is preserved. If not the operation is not performed. The finite erosion step is repeated until the image has not changed.

There is a family of similar approaches that compares a neighbourhood of the pixel with predefined set of masks. If the neighbourhood of the pixel is the same as any of the masks the point is not removed. Otherwise it is removed [DP94].

The next popular approach uses a *distance transform.* Each object pixel is labeled with its distance to the nearest background pixel (different weights for horizontal and vertical directions can be specified). The points with highest values are selected. The next points have to be selected to preserve the connectivity of the region [SdBT94].

Another approach to skeleton extraction is proposed in [HF94]. The contour of the image is extracted in the first step. Then the contour is vectorized by a raster-to-vector convertor. The set of contours is searched for the pairs of lines and pairs of arcs that correspond to the same parts of the object. The compared properties of the lines and arcs are shape, distance and overlapping of the parts. Pairs of contours are matched in the last step and the skeleton is created as a medial axis of the matching contours.

3.2 Boundary

There are three different definitions of the boundary of an object: the inner, outer and crack boundary.

Definition 3.2.1 *The inner boundary (contour) is defined as a set of object pixels each having at least one non-object pixel as its 4-neighbour (see figure: 3.1) [DP94].*

Definition 3.2.2 *The outer boundary is defined as a set of background pixels each having at least one object pixel as its 8-neighbour (see figure: 3.1).*

Note: The crack boundary is given in definition 3.4.9.

There are many ways how to extract a boundary. For example we may use morphological erosion and difference, $b = i - (i \ominus e)$ where b is an image with

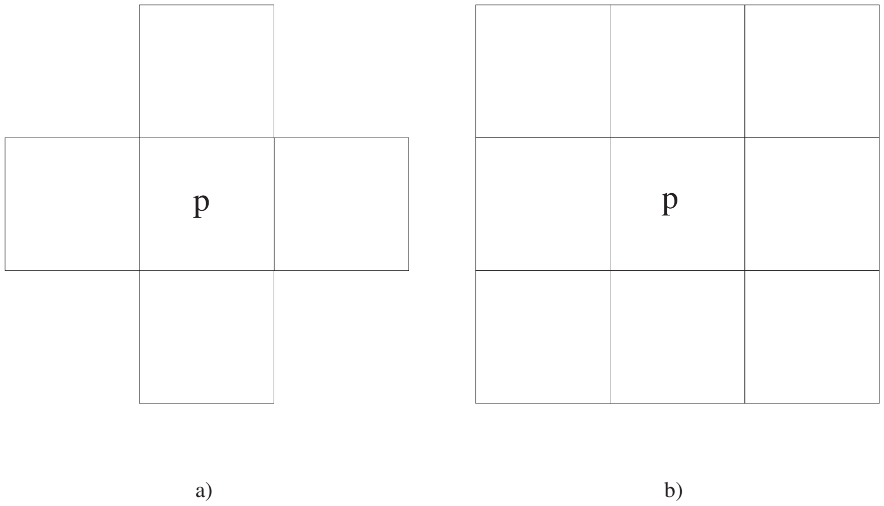

Figure 3.1: 4-neighbours (a), 8-neighbours (b).

boundaries, i is the original image and e is a structuring element. Or we may use distance transform and select pixels with distance 1 to the nearest background pixel.

Edge detectors are also often used. There are many mentioned in literature [HS92, vHB93]. The approximation of the first derivative (gradient) of the image data is computed by convolution with a predefined mask by Roberts (1965), Prewitt (1970), Sobel (1970), Kirsch (1971), Robinson (1977) and others. The approximation of the second derivative (Laplacian) is computed by *zero-crossing* edge detectors by Canny (1986).

3.3 Grammar

Definition 3.3.1 *A grammar is a 4-tuple* $G = (V_T, V_N, P, \sigma)$*, where*

- V_T *is a finite set of terminal symbols,*
- V_N *is a finite set of nonterminal symbols,*
- P *is a finite set of rewriting (production) rules,*
- σ *is an axiom (called the start symbol)* $\sigma \subset V_N$*[CC94].*

Our use of grammars is based on the idea that terminals of the grammar correspond to graphic primitives which can be directly extracted from an input image. The terminal symbols are combined to more complex structures by the production rules. The structures are further combined until all terminals create an acceptable image. The acceptable image is represented by the axiom. This task is called the *exact matching problem.* There is another task, called the *best matching problem*, that looks for the acceptable model closest to the original image.

Definition 3.3.2 *Let $G = (V_T, V_N, P, \sigma)$ be a grammar, $L(G)$ the language generated by the grammar G and $\bar{v}$ a sentence. The exact matching problem aims to determine whether the sentence $\bar{v} \in L(G)$ [Sch96].*

Definition 3.3.3 *Let $G = (V_T, V_N, P, \sigma)$ be a grammar, $L(G)$ the language generated by the grammar and $\bar{v}$ a sentence. The best matching problem is to find a sentence $\bar{u}$ such that $\bar{u} \in L(G)$ and $D(\bar{u}, \bar{v}) \rightarrow \min$, where $D(\bar{u}, \bar{v})$ denotes the distance between two sentences [Sch96].*

There are algorithms that can solve both the exact matching problem and the best matching problem. The algorithm by Kok, Janger and Kassami [Sch96] solves both problems. The algorithm will be introduced in the form for the exact matching problem and the necessary modifications for the best matching problem will be described later.

Let $\bar{v} = (v(i,j), 1 \leq i \leq m, 1 \leq j \leq n)$ be an image of size m rows and n columns. Let $\bar{v}(i_1, i_2, j_1, j_2) = (v(i,j), i_1 \leq i \leq i_2, j_1 \leq j \leq j_2)$ be a subimage. Let f be a five-dimension matrix, $f(s, i_1, i_2, j_1, j_2) \in \{0, 1\}$, where $s \in V_N$ and $i_1, i_2 \in \langle 1, m \rangle$, $j_1, j_2 \in \langle 1, n \rangle$. Let $P = \{P_T, P_R, P_H, P_V\}$ be the sets of rules:

- $P_T : V_N \times V_T \rightarrow \{0, 1\}$ terminating,
- $P_R : V_N \times V_N \rightarrow \{0, 1\}$ renaming,
- $P_H : V_N \times V_N \times V_N \rightarrow \{0, 1\}$ horizontal,
- $P_V : V_N \times V_N \times V_N \rightarrow \{0, 1\}$ vertical.

The algorithm works in the following steps:

1. $f(s, i, i, j, j) = P_T(s, \bar{v}(i, j))$.
2. $f(s, i, i, j, j) = f(s, i, i, j, j) \vee \Big[\bigvee_{s'} \Big(f(s', i, i, j, j) \& P_R(s, s')\Big)\Big]$.
3. $$\begin{aligned} f(s, i_1, i_2, j_1, j_2) = \; & \Big[\bigvee_{i, i_1 \leq i \leq i_2} \bigvee_{s_t, s_d} \Big(f(s_t, i_1, i, j_1, j_2) \; \& \; f(s_d, i+1, i_2, j_1, j_2) \; \& \; P_V(s, s_t, s_d)\Big)\Big] \vee \\ & \Big[\bigvee_{j, j_1 \leq j \leq j_2} \bigvee_{s_l, s_r} \Big(f(s_l, i_1, i_2, j_1, j) \; \& \; f(s_r, i_1, i_2, j+1, j_2) \; \& \; P_H(s, s_l, s_r)\Big)\Big]. \end{aligned}$$

	exact matching	best matching
P_V, P_H, P_R, P_T	$\{0,1\}$	$\{0,\infty\}$
$f(...)$	$\{0,1\}$	R_0^+
$\oplus$	$\vee$	min
$\odot$	$\&$	$+$

Table 3.1: Changes of the Kok, Janger, Kasami algorithm to solve the best matching problem

4. $f(s,i_1,i_2,j_1,j_2) = f(s,i_1,i_2,j_1,j_2) \vee \left[\bigvee_{s'}\Big(f(s',i_1,i_2,j_1,j_2)\&P_R(s,s')\Big)\right].$

The initialization from the input data and all possible renaming of the input data is found in steps 1 and 2. Steps 3 and 4 are repeated as the used cells are filled with values until the last cell $f(\sigma,1,m,1,n)$ is evaluated.

In the cell $f(\sigma,1,m,1,n)$ is the result of the exact matching problem.

To solve the best matching problem the definitions have to be changed according to the table 3.1. In the cell $f(\sigma,1,m,1,n)$ is the distance to the closest image from $L(G)$.

The computational complexity of the algorithm described above is $\mathcal{O}(kmn)$ for step 1, $\mathcal{O}(k^2mn)$ for steps 2 and 4 and $\mathcal{O}(m^2n^2(m+n))$ for step 3, where k is the number of nonterminal symbols and m,n are the numbers of rows and columns, respectively. Because $k \ll n$ and $k \ll m$, the complexity of the whole algorithm is $\mathcal{O}(m^2n^2(m+n))$.

Let us substitute the image by the 1D sequence of the same size. The complexity will be

$$m' = mn, n' = 1$$

and

$$\mathcal{O}(m'^2n'^2(m'+n')) = m^2n^2(mn+1) = m^3n^3$$

and that is much larger than $m^2n^2(m+n)$. So it is easier to solve the exact and best matching problems of images than of sequences.

Not only the structure but also the qualitative properties of the entities have to be represented. The structure is determined by the production rules of the grammar and the absolute or relative distances and sizes which can be represented by *attributes.*

We have to extend the definition of grammars to accommodate the attributes.

We denote a context-free grammar by $G = (V_T, V_N, P, \sigma)$. A production $p \in P$ is denoted by $p:\ X_0 \to w_0X_1w_1\ldots w_{n-1}X_nw_n$, where $n \geq 0, X_j \in V_N$ and $w_j \in V_T^*$ for $j, 0 \leq j \leq n$. V_T^* is a set of finite sequences $v_1, v_2, \ldots, v_n$ where $v_i \in V_T$.

An attribute grammar is defined over a certain *semantic domain*. A semantic domain D is a pair (V, F), where V is a class of attribute values and F is a set of functions of the form $f: V_1 \times V_2 \times \cdots \times V_m \to V_0$ with $m \geq 0$ and $V_i \in V$ for all $i, 0 \leq i \leq m$. A function in F is called a *semantic function*. In the case $m = 0$, f is an element of V_0.

Definition 3.3.4 *An* attribute grammar *is a quadruple $G_A = (G, A, R, D)$ that satisfies the following conditions:*

1. *$G = (V_T, V_N, P, \sigma)$ is a context-free grammar, called the* underlying grammar *of G_A. $D = (V, F)$ is a semantic domain over which G_A is defined.*
2. *Each nonterminal $X \in V_N$ has two disjoint finite sets $I(X)$ and $S(X)$ of* inherited *and* synthesized attributes*, respectively. We denote the set of all attributes of a nonterminal X by $A(X) = I(X) \cup S(X)$, and the set of all attributes of G_A by $A = \cup_{X \in V_N} A(X)$.*
3. *Each production $p: \; X_0 \to w_0 X_1 w_1 \ldots w_{n-1} X_n w_n$ in P has a finite set R_p of* semantic rules. *We denote the set of all semantic rules of G by $R = \cup_{p \in P} R_p$. For each production $p \in P$ and an attribute $a \in A(X_i)$ for $i, 0 \leq i \leq n$, we introduce a symbol $[a, i]$ called an* occurrence *of a in p. The set of all such symbols is denoted by $Occ(p)$. $Occ(p)$ is partitioned into the set of* input occurrences *defined as follows:*

   $$In(p) = \{[a, i] | a \in I(X_0) \quad or \quad a \in S(X_i), i \geq 1\}$$

   and the set of output occurrences *defined as follows:*

   $$Out(p) = \{[a, i] | a \in S(X_0) \quad or \quad a \in I(X_i), i \geq 1\}.$$

   Clearly

   $$Occ(p) = In(p) \cup Out(p), \quad In(p) \cap Out(p) = \emptyset.$$

   R_p includes the definitions of all output occurrences of p. A semantic rule defining an output occurrence $[a, k], 0 \leq k \leq n$ has the form

   $$[a, k] = f([a_1, i_1], \ldots [a_m, i_m]),$$

   where $0 \leq i_j \leq n$, $1 \leq j \leq m$, and $f \in F$ is a function of the form $f: V_1 \times V_2 \times \ldots \times V_m \to V_0$ with $V_0 = V([a, k])$ and $V_j = V([a_j, i_j])$ for all $j, 1 \leq j \leq m$.

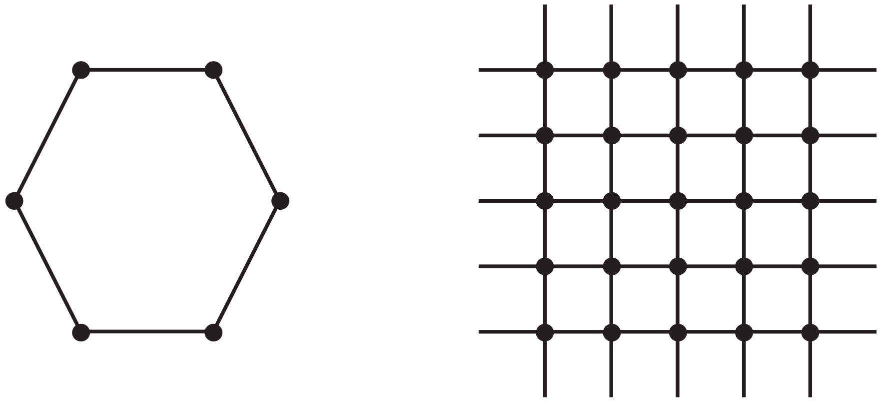

Figure 3.2: Examples of cell complexes.

3.4 Cellular complex

In this section, we define *digital geometry.* Digital geometry gives better results on the data structured into a general grid composed of cells [Kov93].

Definition 3.4.1 *An abstract cell complex (ACC)* $C = (E, B, dim)$ *is a set* E *of abstract elements provided with an antisymmetric, irreflexive, and transitive binary relation* $B \subset E \times E$ *called the bounding relation, and with a dimension function* $dim : E \to I$ *from* E *to the set* I *of non-negative integers such that* $dim(e') < dim(e'')$ *for all pairs* $(e', e'') \subset B$.

Elements of E are called abstract cells. If the dimension $dim(e')$ of a cell e' is equal to d then e' is called a d-dimensional cell or d-cells. If $(e', e'') \in B$ then e' is said to bound e''. Examples of ACCs are shown in figure 3.2, 0-cells are denoted by small circles representing points, 1-cells are denoted by line segments and 2-cells by the interior of polygons. The bounding relation is defined in a natural way: a 1-cell represented by a line segment in the figure is bounded by the 0-cells represented by its end points and 2-cell represented by the interior of a polygon is bounded by the 0- and 1-cells composing its boundary.

Definition 3.4.2 *A subset* S *of* E *is called* open *in* C *if for every element* e' *of* S*, all elements of* C *which are bounded by* e' *are also in* S.

Definition 3.4.3 *A subcomplex* $S = (E', B', dim')$ *of a given ACC* $C = (E, B, dim)$ *is an ACC whose set* E' *and the relation* B' *is an intersection of* B *with* $E' \times E'$. *The dimension* dim' *is equal to dim for all cells of* E'.

This definition makes clear that to define a subcomplex S of $C = (E, B, dim)$ it suffices to define a subset E' of the elements of E.

The definition 3.4.2 defining the notion of open subsets simultaneously defines the open subcomplexes of a given ACC. According to the axioms of topology any intersection of a finite number of open subsets is open. In a finite space there is only a finite number of subsets. Therefore in a finite ACC the intersection of all open subcomplexes containing a given cell c is an open subcomplex. It is called the *smallest open neighbourhood of* c (SON(c)). SON(c) consists of the cell c itself and of all cells of C bounded by c.

Definition 3.4.4 *Two cells* e' *and* e'' *of an ACC* C *are called* incident with each other in C *iff either* $e' = e''$, *or* e' *bounds* e'', *or* e'' *bounds* e'.

Definition 3.4.5 *Two cells* e' *and* e'' *of an ACC* C *are called* connected with each other in C *iff either* e' *is incident with* e'', *or there exists in* C *a cell* c *which is connected to both* e' *and* e''.

The connectivity relation is an equivalence relation (reflexive, symmetric and transitive). Thus it defines a partition of an ACC C into equivalence classes called the *components of* C.

Definition 3.4.6 *An ACC* C *consisting of a single component is called* connected.

Definition 3.4.7 *A sequence of cells of an ACC* C *beginning with* c' *and finishing with* c'' *is called a* path in C *from* c' *to* c'' *if every two cells which are adjacent in the sequence are incident.*

Definition 3.4.8 *An ACC* C *is called* path connected *if for any two cells* c' *and* c'' *of* C *there exists a path in* C *from* c' *to* c''.

Kong et al. have shown that definitions 3.4.5 and 3.4.8 are equivalent. Kovalevsky has shown that these definitions are in full accordance with classical topology and free of paradoxes.

The theory of ACCs leads to a topologically consistent definition of the boundary of a subset of an image. The notion of a boundary remains the same as in general topology.

Definition 3.4.9 *The boundary (frontier) of a subcomplex* S *of an ACC* C *relative to* C *is the subcomplex* $Fr(S, C)$ *consisting of all cells* c *of* C *such that the smallest open neighbourhood of* c *contains cells both of* S *and of its complement* $C - S$. *(See figure 3.3.)*

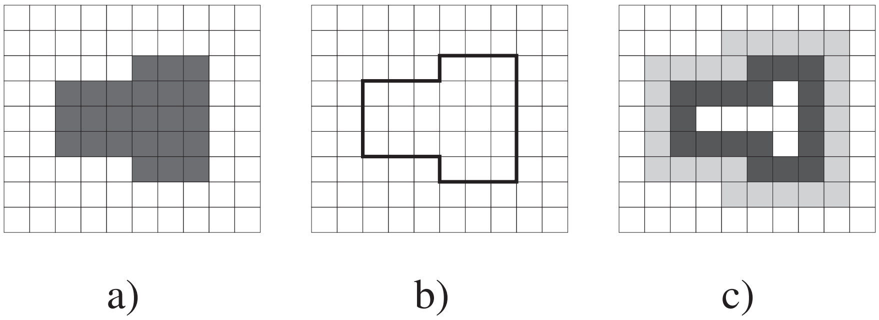

Figure 3.3: A subset (a), its boundary according to definition 3.4.9 (b) and its inner and outer boundaries (c).

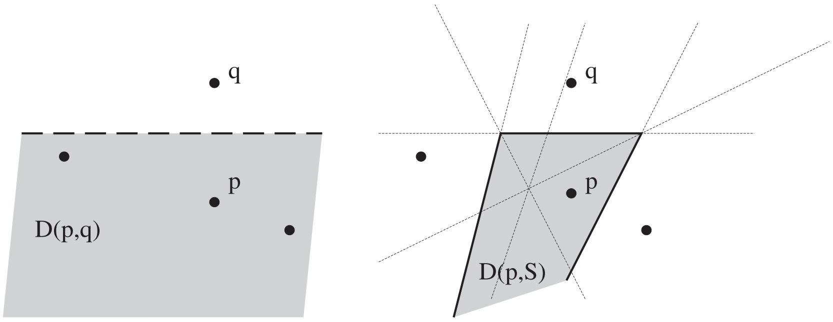

Figure 3.4: A definition of Voronoi region D(p,S).

3.5 Voronoi diagram

Definition 3.5.1 *If a finite set S of objects p_i is given in a space M, computing the Voronoi diagram of S means to partition the space into regions, $R(p_i, S)$, in such a way that $R(p_i, S)$ contains all points of Mthat are closer to p_i than to any other object p_j in S. [Kle89]*

Definition 3.5.2 *Let $S = \{p_1, ..., p_n\}$ be a set of n different points in the plane. For $p, q \in S, p \neq q$, let*

$$B(p, q) = \{z \in R^2; |p - z| = |q - z|\}$$
$$D(p, q) = \{z \in R^2; |p - z| < |q - z|\}$$

$B(p, q)$ is the perpendicular bisector of the line segment joining p with q; of the two half-planes separated by $B(p, q)$, $D(p, q)$ is the one that contains p. (See figure 3.4).

Definition 3.5.3 *The set*

$$D(p, S) = \bigcap_{q \in S, q \neq p} D(p, q)$$

of all points z that are closer to P than to any other element of S is called the (open) Voronoi region of p with respect to S. *(See figure 3.4).*

$D(p, S)$ is the interior of a convex , possibly unbounded polygon. The points on the contour of $D(p, S)$ are those that have more than one nearest neighbor in S, one of which is p.

Definition 3.5.4 *The union*

$$V(S) = \bigcup_{p \in S} \partial D(p, S)$$

of all region boundaries $\partial D(p, S)$ is called the Voronoi diagram *of S. (See figure 3.5)*

The common boundary of two Voronoi regions is a *Voronoi edge*. Edges meet at a *Voronoi vertex*, such a point has three or more nearest neighbors in the set S.

A Voronoi diagram divides an image into cells. Each cell contains one connected component of the image. The edge common to two cells represent the neighbourhood relationship of the two components. If no four points of S are

Figure 3.5: An example of Voronoi diagram created as a skeleton of the background.

cocircular then the planar dual of $V(S)$, if realized by straight lines, is a triangulation of S. This *Delunay triangulation* is uniquely determined by the fact that the circle circumscribing a triangle does not contain a point of S in its interior. This, in turn, is equivalent to saying that the minimum angle of triangles is a maximum over all triangulations.

Three algorithms have been introduced that allow the Euclidean Voronoi diagram of n points in the plane to be constructed within $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ storage, in the worst case.

The first optimal algorithm by Shamos and Hoey, made a divide-and-conquer approach. Here the problem of computing $V(S)$ is split into two subproblems, namely to compute $V(L)$ and $V(R)$, where $S = L \cup R$ is a partition into subsets of about equal size induced by a vertical (or horizontal) separating line. $V(L)$ and $V(R)$ are computed recursively. The main task is in merging the subdiagrams into $V(S)$; this step requires the computation of the bisector

$$B(L, R) = \{z \in R^2; \min_{p \in L} |p - z| = \min_{q \in R} |q - z|\}$$

of the sets L and R.

A simpler description of $B(L, R)$ is that it consists of precisely those edges of $V(S)$ that are common boundary of both an L-region and R-region.

The next optimal algorithm was introduced by Brown. Brown reduced the problem of constructing the Voronoi diagram of n points in Euclidean d-space to the construction of the *convex hull* of n points in $d+1$-space. For $d = 2$, the points of S are mapped onto a sphere by stereographic projection. Three non-colinear points, p, q and r, of S give rise to a Voronoi vertex (with adjacent regions of p, q, r) iff the circle, C, passing through p, q and r contains no further points of S in its interior. Since the convex hull of n points in 3-space was already known to be computable in time $\mathcal{O}(n \log n)$, this result allowed the Voronoi vertices to be determined and, thus, the 2-dimensional Voronoi diagram to be computed, within the same optimal time.

The third algorithm was presented by Fortune. It uses a geometrical transform to bring the Voronoi diagram in such a form that a *sweep-line* technique can be employed in its computation. In the sweep-line approach, a horizontal line l is moved upward across the scene, and at each stage the set of "active" Voronoi edges currently intersected by l is maintained, in their left-to-right order. Whenever this set changes, the sweep-line must "halt", and the information about active edges must be updated.

Chapter 4

Formulation of the research goal

Tasks we want to solve in the proposed research are specified in this chapter. The way towards the solution is specified in more detail.

4.1 What is to be done?

The goal of our research is to find a general description of lines and hatching in images. We want to be able to describe both connected and unconnected structured lines as well as a wide variety of hatching. The classification will be done according to the building elements and the distribution patterns.

We will test the proposed description on data synthesized by computer (no noise, no distortions) and on real data (scanned CAD drawings and maps).

We will find the limitations of the usability of the proposed description and specify the domain of documents that can be described.

4.2 Low level processing

We assume that the following processing was done in the preprocessing step and we will not describe it in detail.

- Noise removing.
- Segmentation of the image, description of segmented parts by sets of attributes and classification of the parts.
- Segmentation of the background of the original image or of an image of items classified into one class by a Voronoi diagram.
- Creating the structure of cells, edges and vertices of the Voronoi diagram.

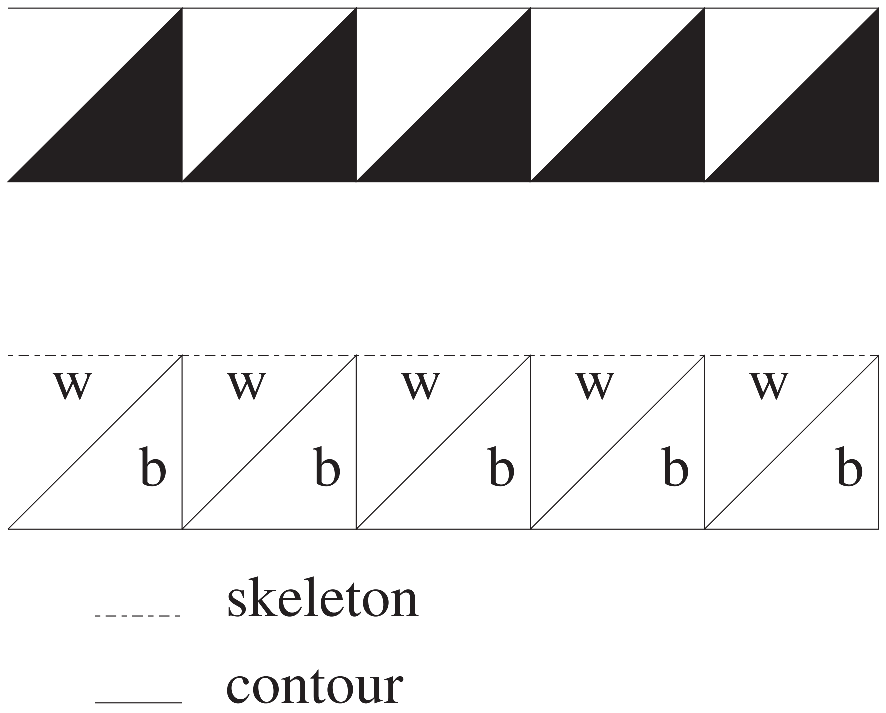

Figure 4.1: An example of the description of connected elements. The original line (top) and the description of the line (bottom) in terms of black and white regions, skeleton and contour lines.

4.3 How to do it?

The solution of the tasks mentioned in section 4.1 is proposed in this section. The basic elements for the description of the image data and for the description of classes have to be selected.

We propose to use skeletons and contours for the description of connected components. Skeletons will be used for the description of line-like parts of components. Skeletons are good representations for the shape and size of these parts. Contours will be used for the description of large areas of foreground pixels. Description by contour is more precise for large regions. The difference between regions described by skeleton and by contour is determined by a threshold computed from the resolution used by scanning and the real size of regions.

Skeleton and contour branches partition the area occupied by the component into cells. There are cells corresponding to the black and to the white areas of the component, edges that correspond to the contours and skeletons, and vertices. The bounding relation can be defined on this partitioning in terms of cellular complexes. For an example of the description of the connected element see figure 4.1.

For the description of unconnected components we will use a Voronoi diagram.

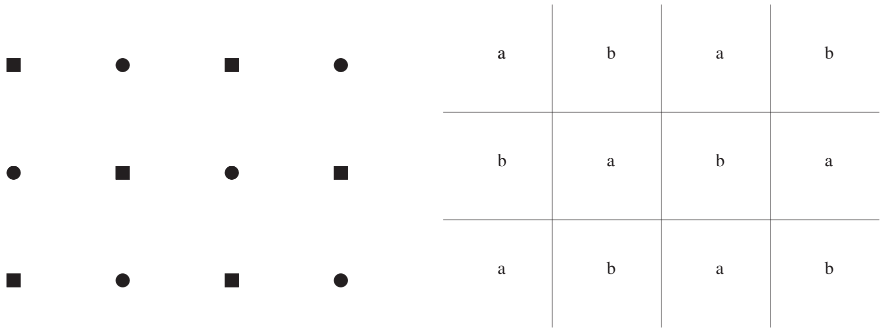

Figure 4.2: An example of the description of an unconnected element. The original pattern (left) and the description (right) by Voronoi diagram, *a* and *b* are classified elements.

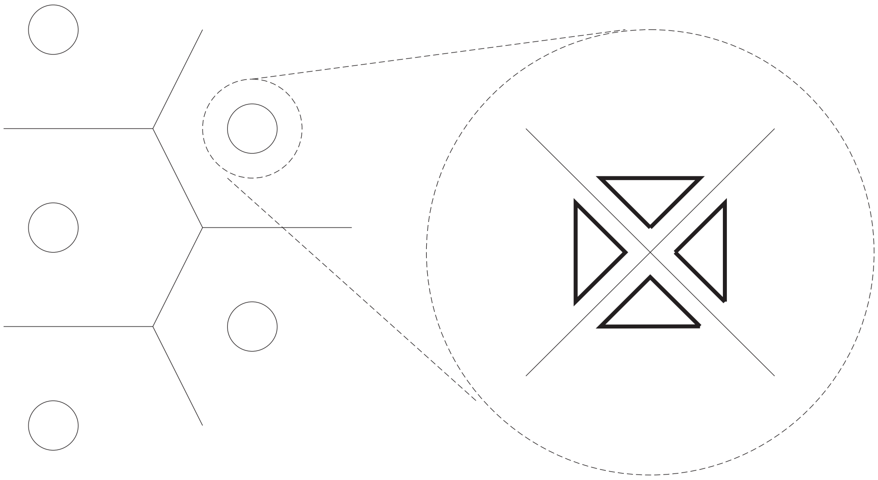

Figure 4.3: An example of hierarchical processing. The structure (on right) creates a more complex structure (on left).

The Voronoi diagram partitions the image into a complex of cells. There is one connected component in each cell. Actually the structure can be also inside a cell. We will describe this case when we discuss the hierarchical structures later in this section. The edges connect closest components of the structure. An example of the description of an unconnected structure is in figure 4.2.

We will talk about cells, edges and vertices. We have shown that patterns can be described by cells, whether they are connected or not. We believe that by laying the constraints on the complex of cells, edges and vertices we will be able to describe and classify general line and hatching.

We want to introduce a hierarchy into the processing of complex patterns. Hierarchical structuring will also reduce the number of production rules used in the grammar. Consider a complex structure as in figure 4.3. The description of the distribution of elements in a triangular grid is easy. The description of each

Figure 4.4: Line recognized as an example in section 4.4.

building element is also simple. We propose to check the existence of the building elements first and to check the triangular pattern in the next step.

4.4 Example of description

One example of the description of a pattern will be shown in this section. Consider a line (see figure 4.4) and its description after preprocessing. Let G be a grammar with the set of terminal symbols $V_{TC} = \{BLACK, WHITE, EXTERIOR\}$, $V_{TE} = \{CONTOUR, SKELETON\}$, $V_{TV} = \{any\}$ for cells, edges and vertices respectively. Let $V_N = \{bw\_line, bound\_line, bound\_element, vertex, inside\_edge, b\_bound, w\_bound\}$ be the set of nonterminal symbols. The nonterminal symbol *bw_line* is an axiom of the grammar. The production rules are then as follows:

1. $bw\_line \rightarrow bound\_line|bound\_line$
2. $bound\_line \rightarrow bound\_line|bound\_element$ or $bound\_element$
3. $bound\_element \rightarrow w\_bound|vertex|b\_bound$
4. $vertex \rightarrow b\_bound|inside\_edge|w\_bound$
5. $inside\_edge \rightarrow WHITE|CONTOUR|BLACK$
6. $b\_bound \rightarrow BLACK|CONTOUR|EXTERIOR$
7. $w\_bound \rightarrow WHITE|SKELETON|EXTERIOR$

This example shows that the description of structured lines is clear and easy. The axiom of the description can be a terminal of another grammar so the description by a hierarchy of grammars is possible too.

Chapter 5

Related work

Existing approaches to the recognition of lines and hatched regions are classified from different points of view in this chapter. They are described briefly and compared with each other.

5.1 The theoretical background

We compare algorithms according to their mathematical background. We found five basic approaches in the literature.

5.1.1 Observation

Hatching recognition in graphical documents. An approach to recognition of hatched regions in maps is presented in [SEB91]. The image is thresholded and the skeletons are created. The short branches are removed. The image is searched by a star like window. The number of passes from the background to the foreground and vice versa is counted in each of eight directions. The second method for finding the areas of interest is to count the number of pixel pairs (0-0, 0-255, 255-0, 255-255) in vertical and horizontal directions. If for a neighborhood of a pixel the number counted by one of the methods above is greater than a certain threshold the pixel is unchanged; otherwise the pixel is removed. Branches with dead ends are removed because they cannot form a hatching. Closed parts that contain no hatching are also removed. Finally the direction and size of the hatching structure is determined.

5.1.2 Mathematical morphology

Automatic line detection in document images using recursive morphological transforms. A system that detects lines of various types, e.g. solid lines and dotted lines is presented in [KCH95]. For faster processing of large data, the image is subsampled in such a way that lines remain preserved. Then solid lines are detected. Noise is removed and gaps are filled; then the image is opened with line-like and box-like structuring elements. The resulting images are subtracted. Connected component analysis and line fitting is used to find the end points. Finally the lines are checked back in the input image. Next dotted lines are detected. This is done in two steps. First dots are found using the hit-miss transform and the dots are grouped into lines. The distance between dots in each line is important for setting the thresholds on the transformed image. The thresholds can be automatically determined from the histogram of the transformed image.

Advantages. The transformation can be performed in constant time per pixel independently of the size of the structuring element. The algorithm offers the possibility of selecting the proper size of the structuring element after the whole image was examined.

Disadvantages. Detects only vertical or horizontal lines. That is good for tables and forms but not sufficient for general description.

Recognition of elevation value in topographic maps by multi-angled parallelism. The *directional field* is introduced in [YYSH94]. The image pixels are classified into 8 groups according to the direction of an edge. The classification is done by morphological operations. The direction erosion, direction dilation and other operations are defined on directional field. The defined operators can be used for the extraction of buildings, linear features, dots and numbers. The next processing creates the elevation number from the components already found. Next parts are searched to both sides of the already recognized components.

5.1.3 Relaxation and clustering

Perfecting of vectorized images. The algorithm [CLD96] works on vector data created by any general raster to vector convertor. In the input data dashed lines and dashed circles are represented as isolated small line segments, but lines and circles should rather be stored as lines or circles with an attribute "dashed". The data is searched for groups of objects of specified properties that could form more complex entity. Next properties are checked to verify that these parts really form the expected entity.

The recreation of arcs from short line segments, recognition of dashed and center entities and global alignment are implemented and described in the paper.

Advantages. Global properties are used for the processing of the data.

Disadvantages. The thresholds in the algorithm are determined by statistical methods and represent only a small subset of documents. Only four kinds of circle arcs are defined and the circular parts of the image are converted into the closest of the four.

Geometric grouping applied to straight lines is introduced in [WB86]. A zero crossing edge detector is used to find the edges in an image. The edges are further processed by the hierarchical linking and merging algorithm. The pair of lines is linked together in the linking step. The lines have to be close to each other, colinear and may not overlap much. Line sequences are approximated by single lines in the merging step. If the approximation is good enough each sequence is replaced by a single line. Hierarchical processing is introduced to find lines at different scales. The lines are divided into classes according to their length. The linking and merging steps are done for each class separately.

Line structure extraction from line-drawing images is described in [Kan92]. Line cues are detected by measuring the length of a pattern in four specific directions. Lines are fitted into detected cues. The fitting produces unlinked approximated lines. The vectorization converts these lines into refined and linked vectors. This approach can link overlapped and crossed lines as well as fill gaps.

Advantages: This approach detects lines of any direction. The algorithm is not sensitive to the presence of the signs and letters.

Disadvantages: Only solid lines are detected.

5.1.4 Knowledge based approaches

Knowledge-directed interpretation of mechanical engineering drawings. A new methodology for the interpretation of CAD drawings is presented in [JP92]. It is based on a combination of low-level image analysis routines, a set of explicit control rules and a set of classes describing prototypical drawing constructs. (Solid and dashed lines, solid and dashed curves, etc.) Each located drawing entity is represented by an instance of a particular class. Schemata form nodes of a network in which arcs correspond to subparts and subclass relations. Image analysis is carried out under the control of a given schema in the context of particular hypothesis regarding the local content of the drawing. So unnecessary low level processing is reduced. Control structure is modeled as the *cycle of perception*. It is a continuous loop in which a changing world model directs exploration, determines how its findings are to be interpreted and is modified as a result. Control system consist of strategy rules written if the form of grammar. The grammatical rules are not used to define acceptable engineering drawings but they specify strategies by which the various components of a drawing might be recognized.

Knowledge-based interpretation of utility maps. An initial approach toward an application independent interpretation mechanism is proposed in [HKG96]. The image is decomposed into primitives. Each primitive is stored in a separate bitmap and its features are measured. The mechanism that generates search actions is based on relation between object types. Detection of an object generates expectations about other objects in its neighborhood, which generate new search goals. Each search action is distributed to a bottom-up process, which tries to detect new objects. The object types and their spatial relationships are represented by a semantic network.

5.1.5 Grammars

Syntactic recognition of dimension sets using grammar is presented in [CC94]. The plex-grammar is defined and used to represent general dimensioning according to the ISO standard (after small changes to the grammar ANSI dimensioning can be recognized). *Graphics primitives* are recognized in the image in the low-level processing step as well as some numeric attributes (position, size, orientation). *Assembly operators* are used to define whether a pair of objects can be connected and to define the topological relationship of two objects (above, below, by site of). The *grammar* is used to control the building of the dimension set from the primitives. Thresholds and heuristics are used in the preprocessing step rather than in the grammar parser. The processing of the grammar is left more general and easy to extend for different tasks. The processing starts from the well-recognized (starting) primitive; and applies a bottom-up approach that determines the objects that can surround the starting primitive. The surrounding primitives are determined by the top-down part of the algorithm. The presence of the expected terminal primitives is checked in the input image.

A stochastic attribute grammar model of document production and its use in document image decoding is described in [CK95]. A theoretical background for recognition of text documents is laid in the framework of information theory. The recognition is described as an process inverse to document production and its transition through the noisy channel (the paper, printing and scanning). The attribute grammar is proposed to be used for representation of characters and larger pieces of text.

5.2 Use of knowledge

Approaches are compared according the use of explicit knowledge in this section. There are different levels of use of knowledge about the processed document. The use of knowledge has great influence on the useability of the algorithm for different documents and documents of various quality.

Knowledge can be implicit and hidden in the source code as is in [LK94, SEB91, KCH95, Kan92, WB86, CLD96]. The systems are then for one narrow set of applications only and are difficult to adjust. Knowledge can be made more explicit using rules [CK95, CC94, JP92] and the system can be adapted just by changing the set of rules, the rest will remain unaffected.

5.3 Interpretation strategy

In this part the approaches are compared according to the processing strategy. Some of them combine primitives found in the image and compare them with predefined constraints; they are called bottom-up. Others build an model of the recognized document and update the model according the data found in the image; these are called top-down. The rest combines both approaches.

5.3.1 Top down

There are no approaches using a pure top-down strategy that we know about.

5.3.2 Bottom up

There are approaches that use a bottom-up strategy to process data; for example [Kan92, CLD96]. This strategy is natural for building the interpreted drawing from pieces. There is an disadvantage of the bottom-up approach. This approach cannot correct mistakes made on the lower level of processing. Control of the interpretation process is difficult.

5.3.3 Combined

A combination of the bottom-up and top-down approaches is described in [CC94]. The starting symbol is found in the image and by a bottom-up strategy the surrounding objects are determined. For example suppose the object F was found in the image, and that there is a rule $D \rightarrow \alpha F \beta$ in the grammar. The rule means that an object D consists from three objects α, F and β. The top-down strategy is applied to find objects $\alpha and \beta$. So a kind of "growing" was created by the combination of two strategies.

There are other combined methods like [HKG96] that use a top-down strategy to create hypotheses about the drawing and a bottom-up strategy to extract primitives from the image data.

Chapter 6

Thesis plan

We will focus the dissertation research on the following topics.

- The description of structured lines by attribute grammars, its generality, efficiency and limitations.
- The description of hatching patterns by attribute grammars, its generality, efficiency and limitations.
- The description of the hierarchies of patterns.

In the theoretical part we expect to evaluate the properties of structured lines and hatching represented by attribute grammars. This description is new. We have not found any articles about it in the literature.

In the experimental part all tests will be done on artificial noise-free data and real images, scanned engineering drawings and various maps. The patterns will be tested both, by themselves and in the context of surrounding patterns. The results will be evaluated according to the specification in [KHD95].

The experimental environment will be created and the algorithms will be implemented. The created programs will be useable as a prototype of the module for the document interpretation system.

Bibliography

[AK94] J.F. Arias and R. Kasturi. Recognition of graphical objects for intelligent interpretation of line drawings. In C. Arcelli, L.P. Cordella, and G. Santini Di Baja, editors, *Aspects of Visual Form Processing*, pages 11–31. World Scientific, Singapore, 1994.

[Be92] L. Boatto and et al. Detection and separation of symbols connected to graphics in line drawings. In *Proceedings of the 11th International Conference on Pattern Recognition*, volume II., pages 545–548, Los Alamitos, California, September 1992. IEEE Computer Society Press.

[CC94] S. Collin and D. Colnet. Syntactic analysis of technical drawing dimensions. *International Journal of Pattern Recognition and Artificial Intelligence*, 8(5):1131–1148, 1994.

[CK95] P.A. Chou and G.E. Kopec. A stochastic attribute grammar model of document production and its use in document image decoding. *Proceedings of the SPIE - The International Society for Optical Engineering*, 2422:66–73, 1995.

[CLD96] Y. Chen, N.A. Langrana, and A.K. Das. Perfecting vectorized mechanical drawings. *Computer Vision and Image Understanding*, 63(2):273–286, March 1996.

[DP94] A. Datta and S. K. Parui. A robust parallel thinning algorithm for binary images. *Pattern Recognition*, 27(9):1181–1192, 1994.

[HF94] C. C. Han and K. C. Fan. Skeleton generation of engineering drawings via contour matching. *Pattern Recognition*, 27(2):261–275, 1994.

[HKG96] J. E. D. Hartog, T. K. T. Kate, and J. J. Gerbands. Knowledge-based interpretation of utility maps. *Computer Vision and Image Understanding*, 63(1):105–117, January 1996.

[HS92] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision, Volume I.* Addison Wesley, Reading, Ma, 1992.

[JP92] S.H. Joseph and T.P. Pridmore. Knowledge-directed interpretation of mechanical engineering drawings. *IEEE Pattern Analysis and Machine Intelligence*, 14(9):928–940, September 1992.

[Kan92] T. Kaneko. Line structure extraction from line-drawing images. *Pattern Recognition*, 25(9):963–973, 1992.

[KHD95] T. Kanugo, R. M. Haralic, and D. Dori. Understanding engineering drawings: A survey. Technical report, ISL University of Washington, 1995.

[KCH95] B. Kong, S. Chen, and R. M. Haralic. Automatic line detection in document images using recursive morphological transforms. Technical report, Department of Electrical Engineering, University of Washington, 1995.

[Kle89] R. Klein. *Concrete and Abstract Voronoi Diagrams.* Lecture Notes in Computer Science. Springer-Verlag, 1989.

[Kov93] V. Kovalevsky. Digital geometry based on the topology of abstract cell complexes. In *?? Conference Somewhere in France*, pages 259–283. ?? France, 1993.

[LK94] C. P. Lai and R. Kasturi. Detection of dimension sets in engeneering drawings. *IEEE Pattern Analysis and Machine Intelligence*, 16(8):848–855, August 1994.

[Men91] L. Meng. Toward the automatic digitization of map text. In B. Radig, editor, *13. DAGM-Symposium*, pages 361–366, Berlin, October 1991. Springer Verlag.

[MSY92] S. Mori, C.Y. Suen, and K. Yamamoto. Historical review of ocr research and development. *Proceedings of the IEEE*, 80(7):1029–1058, july 1992.

[Nis95] H. Nishida. Model-based shape matching with structural feature grouping. *IEEE Pattern Analysis and Machine Intelligence*, 17(3):315–320, March 1995.

[RP94] J. Rocha and T. Pavlidis. A shape analysis model with applications to a character recognition systems. *IEEE Pattern Analysis and Machine Intelligence*, 16(4):393–404, April 1994.

[SdBT94] G. Sanniti di Baja and E. Thiel. (3,4) weighted skeleton decomposition for pattern representation and description. *Pattern Recognition*, 27(8):1039–1049, 1994.

[SEB91] W. Shen, N. Ebi, and Ph. Besslich. Schraffurerkennung in graphischen vorlagen. In B. Radig, editor, *13. DAGM-Symposium*, pages 407–412, Berlin, October 1991. Springer Verlag.

[Sch96] M. Schlesinger. Theoretical aspects of pattern recognition. The lectures on CTU Prague, Faculty of Electrical Engineering, March-Jun 1996.

[vHB93] M. Šonka, V. Hlaváč, and R.D. Boyle. *Image Processing, Analysis and Machine Vision.* Chapman & Hall, London, U.K., first edition, 1993.

[WB86] R. Weiss and M. Boldt. Geometric grouping applied to straight lines. *CVPR??*, pages 489–495, 1986.

[YYSH94] H. Yamada, K. Yamamoto, T. Saito, and K. Hosokawa. Recognition of elevation value in topographic maps by multi-angled parallelism. *International Journal of Pattern Recognition and Artificial Intelligence*, 8(5):1149–1170, 1994.