

Philips Research Laboratory
Ave. Em. van Becelaere 2, Box 8
B-1170 Brussels, Belgium

Report R.469

**REAL-TIME DETECTION
OF CONNECTED COMPONENTS
IN BINARY PICTURES**

Christian Ronse and Pierre A. Devijver

February 1983

Abstract: Separation of objects from their background is a major problem in pattern recognition and scene analysis. It finds applications in many fields ranging from optical character recognition to biomedical engineering. It is often associated with other processes like thinning, vectorization, etc.

Besides the question of their mathematical validity, algorithms for extracting connected components in real-time are subjected to implementation constraints resulting from the actual state of technology, namely,

- ▶ the limited size of the memory: if we wanted to store, say, a whole A4 sheet of paper digitized at facsimile resolution in computer memory, we would need storage space for about 4×10^6 bits, which is frequently much too large with regards to the amount of meaningful information contained in it;
- ▶ the need for real-time processing: in a number of applications, features (e.g., connected components) should be extracted at the time they are encountered in order to avoid memory congestion and permit a smooth cooperation of cascaded processes.

In virtually every application, it is normally expected that the detection of any connected component be accompanied by some concise description of that component. We start from the premise that one should not overlook the amount of information which is made available at the output: For pattern recognition purposes, it is preferable to receive a minute description of the component structure rather than, say, the sole chain-code of its borders.

In this report, we outline an algorithm—up to the code level—which satisfies these requirements to a great extent.

In terms of memory requirements, we assume that the input image consists of an arbitrary number of rows of given length, and is read in a single raster scan. The program maintains essentially two types of memories: a working memory whose size is linear in the size of the rows, and an object memory containing *i*) partial descriptions of connected components under detection, and *ii*) surrounding relations between them.

The algorithm detects the completion of the scan of any connected component at the time the last pixel of that component is being read. Concomitantly, the description of that component is transferred to an output buffer in a time linear in the size of that component. Thus, the algorithm operates in *real-time* at the component level.

For what concerns component descriptions, the algorithm performs a decomposition of components in 2-dimensional structural elements called *blocks* and *hinges*. Geometrical information about components is encoded in the geometry of these elements as well as adjacency relations between them. Topological information is encoded in terms of surrounding relations between *outer-edges* of connected components of both the figure and the background.

The program offers a number of options, e.g., 4- or 8-connectivity. The

two major options concern the level of detail of the geometrical and topological descriptions desired in the output. They are called *full* or *restricted adjacency*, and *full* or *restricted surrounding* respectively. They can be activated interactively and are fully orthogonal in the sense that any combination of both is feasible. Thanks to these options, our approach is quite versatile, and the algorithm can be readily adapted to the needs of a wide variety of possible applications.

The algorithm is presented in successive stages through the chapters of this report. Every attempt was made to keep the presentation self-contained. The theoretical background is recalled whenever needed, and the discussion is carried up to the darkest corners of the implementation. The complete program can be found in the appendices. The algorithm was implemented in *Pascal*, and was tested on the VAX 11/780 computer at the Philips Research Laboratory Brussels.

Two major factors concern the level of detail of the geometrical and topological descriptions desired in the output. They are called full or reduced or summary and full or restricted surrounding respectively. They can be selected interactively and are fully orthogonal in the sense that any combination of both is possible. Unlike in these systems, our algorithm is quite versatile and the algorithm can be modified to suit the needs of a wide variety of possible applications.

The algorithm is presented in a series of steps through the steps of this report. Every step was made to keep the presentation self-contained. The theoretical background is recalled whenever needed, and the discussion is carried up to the latest context of the implementation. The complete program can be found in the appendix. The algorithm was implemented in Pascal and was tested on the VAX-11/750 computer at the Philips Research Laboratory Eindhoven.

Contents

1	INTRODUCTION	1
1.1	The Problem and Possible Approaches	1
1.2	Topological and Geometrical Information	2
1.3	Program Options	3
1.4	Summary of the Report	4
2	REAL-TIME SEQUENTIAL PROCESSING OF BINARY PICTURES	7
2.1	The Fundamental Constraints	7
2.1.1	Sequential processing	8
2.1.2	Limited memory	9
2.2	Some Definitions and Notation	11
3	HORIZONTAL BLACK RUNS AND THEIR PARAMETERS	13
3.1	Definitions and Some Properties of Runs	13
3.2	The Parameters of a Run	15
3.3	The Computation of Run Parameters	17

4	DECOMPOSITION OF A FIGURE INTO BLOCKS AND HINGES	21
4.1	Blocks, Hinges, d-Blocks and Block-Continuations	21
4.2	Basic Properties of Blocks and d-Blocks	24
4.3	Adjacency Relations Between Objects	25
4.4	Representation of Objects and Adjacency Relations	28
4.5	Real-Time Construction of Object Records	32
4.5.1	Data structure and basic processing steps	33
4.5.2	Creating and updating object records	36
4.5.3	Comments	39
5	THE DETECTION OF CONNECTED COMPONENTS	41
5.1	Preliminaries	41
5.2	Edge-Following in a Figure	43
5.2.1	Notation and theoretical background	43
5.2.2	Full adjacency	47
5.2.3	Restricted adjacency	53
5.2.4	Implementation	54
5.2.5	Comments	56
5.3	The Neighborhood-Tree and its Representation	57
5.3.1	The neighborhood-tree of a figure	58
5.3.2	The vertex-string and the edge-string	61
5.3.3	Full surrounding and restricted surrounding	65
5.3.4	The representation of cycles and edge-strings	68
5.4	The Construction of Cycles and Edge-Strings	70
5.4.1	Introduction	70
5.4.2	Relations between chains and cycles	71
5.4.3	Representation of chain extremities and their properties	75
5.4.4	Implementation	80
5.5	Comments	89
6	OUTPUT PROCEDURES AND SAMPLE OUTPUT	91
6.1	Output Procedures	91
6.2	Sample Output	95

6.2.1	Restricted adjacency and full surrounding	95
6.2.2	Full adjacency and restricted surrounding	100
7	CONCLUDING COMMENTS	103
7.1	A Guided Tour of the Program	103
7.2	Some Possible Extensions	105
Appendix A: Main Program and Initialization		109
A.1	Main Program	109
	A.1.1 Comments	112
A.2	Initialization	113
Appendix B: The Processing of Rows		115
B.1	Procedure Process on Row	115
B.2	Procedure Transition to the Next Row	116
B.3	Procedure Window	
Appendix C: Procedure Allocate		119
C.1	Allocate	119
C.2	Comments	121
Appendix D: The Processing of Objects		123
D.1	Procedure Endof	123
D.2	Procedure Conbelow	124
D.3	Procedure Newobject	125
D.4	Procedure Thisrowobjty0	128
D.5	Procedure Thisrowobjty1	129
D.6	Procedure Blockenlarge	129
D.7	Procedure Continuationenlarge	129
D.8	Procedure Newhinge	130
D.9	Procedure Newblock	130

D.10	Procedure Newcontinuation	130
------	---------------------------	-----

Appendix E: The Processing of Cycles	131
---	------------

E.1	Procedure Concatenate1	131
E.2	Procedure Concatenate2	132
E.3	Procedure Enclose1	133
E.4	Procedure Enclose2	134
E.5	Procedure Extendchain	134
E.6	Procedure Newchain	135
E.7	Procedure Mergechain	136
E.8	Procedure Closechain	136

Appendix F: Output Procedures	139
--------------------------------------	------------

F.1	Procedures Outcy	139
	F.1.1 Procedure Outcy1	139
	F.1.2 Procedure Outcy2	141
F.2	Procedures Outobj	143
	F.2.1 Procedure Outobj1	143
	F.2.2 Procedure Outobj2	145
F.3	Procedures Idobj	146
	F.3.1 Procedure Idobj1	146
	F.3.2 Procedure Idobj2	147
F.4	Procedure Outxy	147
F.5	Procedure Interncy	149