Pattern Recognition Letters 12 (1991) 483-489
North-Holland

August 1991

Fast boundary extraction for industrial inspection

Q.M. Wu and M.G. Rodd

Department of Electrical & Electronic Engineering, University College of Swansea, Singleton Park, Swansea, SA2 8PP, United Kingdom

Received 7 January 1991
Revised 10 April 1991

Abstract

Wu, Q.M. and M.G. Rodd, Fast boundary extraction for industrial inspection, Pattern Recognition Letters 12 (1991) 483-489.

This paper presents an integrated and knowledge-driven boundary extractor, which can extract closed boundaries of touching objects directly from grey-level images in specific, but generic, real-world industrial environments.

Keywords. Boundary extraction, industrial inspection.

1. Introduction

As discussed in Wu (1990), the efficient extraction of object boundaries is naturally of great importance in industrial visual inspection systems. One of the classical approaches adopted for the extraction of the boundaries of objects in digital pictures is based on contour tracing (Yokio et al., 1975). This algorithm is capable of efficiently extracting boundaries of isolated objects in ideal backgrounds. By simply applying it to an industrial image, in which objects may be touching each other and in which there are 'false features' present (as shown in Figure 1), however, it is immediately clear that the desired boundaries cannot be extracted (Figure 2). In this paper, we demonstrate via a specific example that a boundary follower, which is efficient in terms of high speed requirement and has intelligence to resolve ambiguities caused by touching points etc., must integrate routines with implicit rules to do fast boundary tracing with an explicit rule-based system which 'knows' when a touching point is met. It should then be able to make decisions to tell the follower which way to go. To this end, a rule-based boundary extraction algorithm is presented in the next section. In Section 3 the performance of the algorithm is analysed in terms of processing speed and boundary extraction accuracy. Finally, in Section 4, conclusions are drawn.

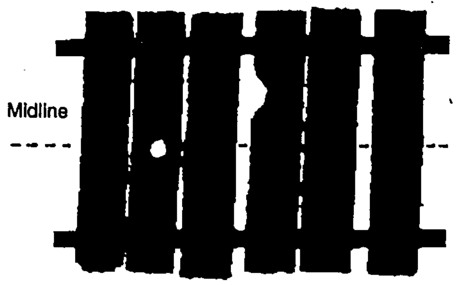


Figure 1. An image with non-ideal background and touching objects.

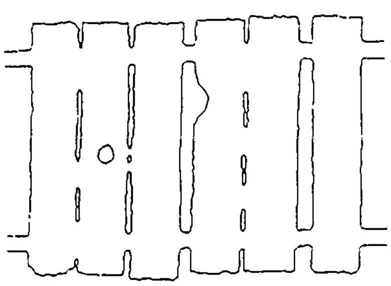


Figure 2. The boundaries extracted using a conventional boundary extractor.

2. A rule-based boundary extraction algorithm

The first step in extracting the boundaries of any objects in an image, using the contour follower described in Yokio et al. (1975), is to locate an object and find an edge point on each object from where to commence boundary tracing. In practical situations, the use of a priori knowledge about the environment can significantly speed-up such an object-location process. For example, in the application tackled in this work the objects to be inspected are supported by a double-band conveyor system (a set of two narrow supporting rubber belts), hence the process adopted for locating objects in an image can be reduced from scanning the whole image to just one line, say, the middle line of the image. This is feasible as the middle line crosses all the objects in the image. A procedure, referred to as *the midline scheme* can thus be introduced for locating the objects, and is described as follows:

The midline scheme

In an image $I(0 \leqslant x \leqslant L,\ 0 \leqslant y \leqslant W)$, we have a priori knowledge that the standard object width is B_w (in pixels) with a tolerance

$$[-B_t, +B_t], \quad \text{where } B_t < B_w/4.$$

A possible *trace-starting-point* $P_s(x_s, y_s)$ is found if

$$I(x_s - 1, y_s) > T,$$

where T is the threshold for binarization, and $I(x_s, y_s) \leqslant T$ is satisfied. To confirm the point P_s as a true starting-point, we need to find a successive point $P_e(x_e, y_e)$ on the midline satisfying

$$I(x_e, y_e) \leqslant T \quad \text{and} \quad I(x_e + 1, y_e) > T.$$

The distance between P_s and P_e is denoted as d_{se}. Because of the possible touching between different objects, the trace-starting-points for some objects may not be found. This can be seen from the value of d_{se}. In terms of this, d_{se} can have three cases—false; the trace-starting-point for a non-touching object, and the trace-starting-point for a touching object. These can be identified as

non-touching case,	if $B_w/2 < d_{se} < 3B_w/2$;
touching case,	if $d_{se} > 3B_w/2$; and
false case,	if $d_{se} < B_w/2$.

These three cases are, respectively, shown in Figure 3. In this way, all the trace-starting-points for the non-touching objects can be found. However, when several objects are touching each other, only the starting-point for the first object can be found. However, if the procedure can report *how many objects* are in contact, the remaining starting-points become progressively available, as the boundary of each object in the touching sequence is successfully extracted. The number of objects in the touching sequence can be found by examining the value of d_{se}, i.e., if

$$(n - (1/2))B_w < d_{se} < (n + (1/2))B_w,$$

there are n objects in the touching sequence. Note that 'touching' and 'non-touching' here only mean whether they are touching or not touching on the middle line. Thus if two objects are found, in this way, touching each other, they must be touching, but if two objects are not touching each other on the middle line, they may still touch each other

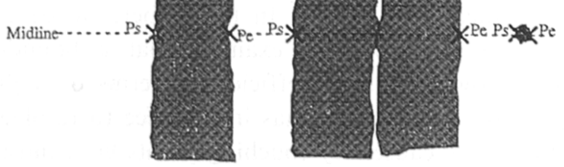


Figure 3. Starting-point location in three cases: non-touching, touching, and noise.

elsewhere. This will not cause problems, however, in extracting the boundary of each object—as will be seen later.

Having found the trace-starting-points, the boundary follower described in Yokio et al. (1975) can then be used to trace the boundary of an object. If the object is not touching others, the boundary-tracing operation would be the same as that described in Yokio et al. (1975). But if the object *is* touching, the contour-tracing is misleading because of the touching, as shown in Figure 4. To resolve this problem, we need to have rules which can determine when a touching point is met. However, we cannot use the same algorithm described in Yokio et al. (1975) to trace the piece of boundary in the touching area, since the searching rule for border points developed there cannot be used here—simply because there are no background pixels in the touching area.

What we can do now is to find, with the aid of a priori knowledge, a 'resuming point' from where the contour tracing can be *resumed* as normal. Then we can put an appropriate segment (e.g., a line or a curve) between the *touching-point* and the *resuming-point* as part of the contour. This is reasonable, since the touching cases occurring in many applications, including the one described later in this section, are likely to be points touching or small areas touching. Thus, the most efficient way for extracting a closed boundary for an object in contact with another, is to resume the normal tracing wherever it is not touching. However, if edges are totally in contact over long regions, the algorithm will not work.

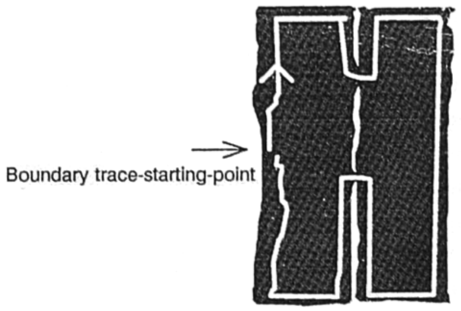


Figure 4. Illustration of boundary tracing for touching objects using a conventional boundary follower.

The thing to be noted in putting a line segment in between the first touching-point and the resuming-point, is to ensure that the touching area is separated in the image, so that there will be no touching problem in this area when tracing the next boundary. The rules suggested for finding such a touching-point and a suitable resuming-point are described below.

Touching-point finder

As shown in Figure 5(a), assume that the current count on the boundary point is i, and let x_i, y_i represent respectively the x and y coordinates of the boundary point P_i, and P_{i-s} and P_{i-t} respectively represent the points s and t points behind P_i. It can be seen from Figure 5(a) that because of the touching at the area around the point P_{i-t}, the boundary tracing is leading from point P_{i-2t} to point P_{i-t-s}, P_{i-t}, P_{i-t+s} and point P_i. The correct contour tracing should be leading from point

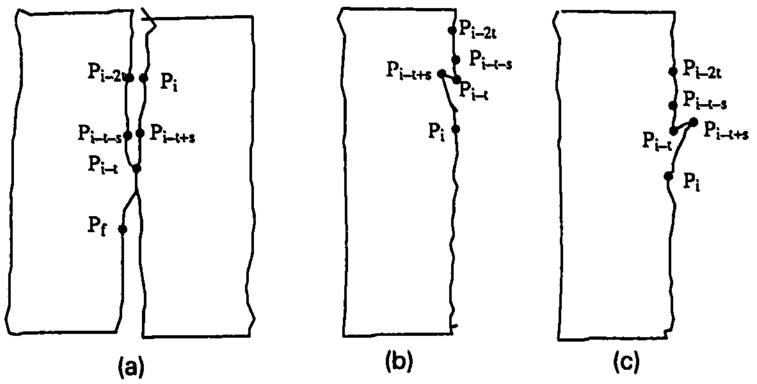


Figure 5. Touching-point detection. (a) Boundary touching each other. (b) A defective object. (c) Another defective object.

P_{i-2t} to P_{i-t-s}, then to P_{i-t} and P_f. Obviously misleading starts from point P_{i-t}.

To locate the touching area, it is important to find the major features illustrated on the contour near to the touching area. In analysing the desired boundaries extracted under ideal conditions for objects used in a specific application in the food industry, it is clear that v-shaped boundary structures should not exist on these boundaries. Therefore, if a v-shaped structure is produced in the contour-tracing process, it is appropriate to assume that touching between objects has occurred.

From Figure 5(a), it can be seen that when a v-shaped structure occurs, the following expressions exist. (Note: x is increasing rightwards, and y is increasing downwards.)

$$y_{i-t}-y_{i-t-s}>0, \tag{1}$$

$$y_{i-t}-y_{i-t+s}>0, \tag{2}$$

$$x_{i-t+s}-x_{i-t-s}>0, \tag{3}$$

$$x_i-x_{i-2t}<C \tag{4}$$

where C is a constant distance representing the 'closeness' of two points. The satisfaction of expressions (1) and (2) means occurrence of a v-shaped structure on a contour. Expression (3) and (4) are used to confirm that the satisfaction of expressions (1) and (2) has not resulted from the two possible defective cases shown in Figure 5(b) and (c). It can be seen from Figure 5(b) and (c) that both defective cases satisfy expressions (1) and (2), but the defective case shown in Figure 5(b) does not satisfy expression (3), and neither case satisfies expression (4). The appropriate values of constant C, and parameters s and t, are those which enable a v-shaped structure caused by objects touching to satisfy all the expressions (1)–(4), and that caused by a defective object to satisfy none, or only some expressions. The actual selection of this value is very much dependent on the specific application and the user's experience. Normally C ranges from 4 to 7 pixels, t from 13 to 15 and s from 7 to 10.

It should be pointed out that, for a v-shaped structure on the other side of an object, the inequalities in expressions (1), (2) and (3) should be reversed and the inequality in expression (4) is changed to

$$x_{i-2t}-x_i<C.$$

Having derived an algorithm for finding points situated in the touching area, we then need to find a point which is 'best' for use in the separation processes. It is obvious that the line segment which is used for segmenting the touching objects should be the one that is closest to the true boundaries of both objects. Therefore the touching-point, used as one of the two points forming the segmenting line, should be the *first touching-point* which is shared by the boundaries of both objects in contact. It will be the first point satisfying the condition

$$y_{\text{touch}}=\min\{y_{i-t-s}, y_{i-t+s}\}.$$

If there are only two points (or even just one) in the touching area, the rule for finding the first touching point will be still the same. In this case, the line segment would just consist of three points: the two touching points and the resuming point. As will be shown in detail in Section 3, the separation of the touching objects, using the above point, will indeed yield the least distortion to both objects in the contact.

Resuming-point finder

Let x_t, y_t represent the x and y coordinates of the first touching point P_t. The search for the resuming point takes place in a window $[x_t-W_t, x_t+W_t]$ as shown in Figure 6, where W_t is a constant, until the first point satisfying

$$I(x,y)<T \quad \text{and} \quad I(x+1,y)>T$$

is found, which is regarded as the resuming point.

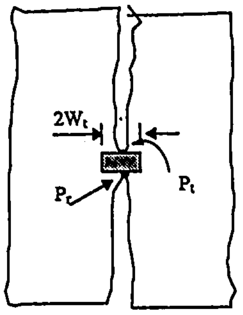


Figure 6. The window for resuming-point location.

The constant W_t is selected as a value less than B_w, often chosen as $B_w/4$, to ensure that the searching for background points is taking place within two objects.

The incorporation of these operations into the original boundary follower enables it to operate even when objects are touching, but so far we have not tackled the case when there are false features—such as background details arising from, say, a conveyor system—present in an image. As in the touching problem, false features mislead the contour tracing, as shown in Figure 7. To eliminate the effect of false features, such as a conveyor belt, on the boundary extraction, a procedure is proposed below.

False-feature elimination

The basic principle suggested for eliminating the false-feature effect, appropriate to the particular industrial applications tackled in this paper, is one of knowledge-organisation. Simply, before the inspection starts, a program is run to allow a frame of the image of the empty belt to be captured, so that the position of the belt and its features can be stored by the system. This information is subsequently regularly checked during on-line inspection. During the boundary-tracing phase, the tracer is given the intelligence to know if it hits, say, the belt's edge. If it does, it 'jumps over' the belt and resumes the tracing, and this part of the object boundary is replaced by a straight-line segment. This procedure is, of course, application-dependent, and is acceptable here in that the width of each band of the conveyor is far shorter than the

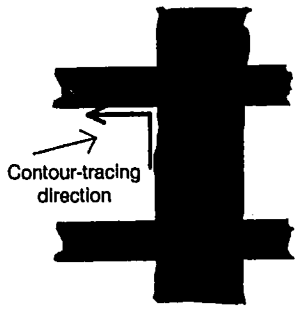


Figure 7. Illustration of contour tracing when a false feature is encountered using a conventional boundary follower.

object length, and the edges of the objects are straight lines.

The belt-elimination procedure is essentially self-learning. For example, as a boundary is traced, it can learn about the approximate orientation of the objects lying on the belt, and thus it goes to approximately the right position to look for boundary-trace resuming-points.

Thus, by the incorporation of explicit, environment knowledge, we have solved touching and false-feature problems. We have also speeded up the boundary-extraction process, eliminating lengthy operations by using a priori knowledge. In the next section, we shall examine the performance of the proposed rule-based boundary extractor.

3. Performance analysis

The proposed rule-based boundary extractor has been extensively used in applications in the food industry—in particular in a biscuit-inspection system (Wu, 1990). A typical frame of the image captured from a full-scale model of an actual production line is shown in Figure 1, and is used here as an example to show the performance of the rule-based boundary extractor.

The image resolution in this example is 280 by 384 with gray-levels ranging from 0 to 255. The middle line is, thus, 140. By activating the trace-starting-point finding module, 6 complete biscuits are found in the image, but with only 4 desired trace-starting-points (respectively for the first, the second, the fourth and the sixth biscuits) found in the first instance, due to touching. The program reports that the trace-starting points for the third and sixth biscuits (counted from the left) are to be obtained when tracing the boundaries of the second and fourth biscuits.

In tracing the boundaries of the first and the second biscuits, all the modules in the boundary extraction system are activated. In this example, the parameters chosen, by experience, were $t=15$, $s=5$. The first biscuit's boundary length was 355, and the time taken was 86 ms (executed on a single INMOS T800 transputer). The second biscuit's boundary length was 357, and the time taken was 86 ms. Note that the time taken for extraction of the boundary from a non-touching object is the total time for tracing a normal boundary and for eliminating the false features from, say, a conveyor system. The time taken for extraction of the boundary from an object touching others is the total time for tracing a normal boundary, eliminating false features, and separating touching areas.

In the course of separating the touching area between the second and the third biscuits, a trace-starting-point, which is the touching-point closest to the $y = 140$ line, is stored for use in tracing the boundary of the third biscuit. After the second biscuit boundary had been traced, the third one was found to be an ordinary non-touching one. Therefore, only the basic boundary follower is activated to trace the boundary of the third biscuit. Its boundary length was 357; the time taken was 82 ms. The processing of the rest is similar to that of the first three.

The extracted boundaries of the biscuits are shown in Figure 8; the total time taken for extracting all boundaries of the biscuits in the image (including the time for finding boundary-tracing starting-points) using just one INMOS T800 transputer was about 510 ms.

To quantitatively analyse the distortion caused by touching of the actual object boundaries, we need to have a detailed look at a touching area. In Figure 9, a, b, c, ..., i represent pixels on object 1 only, j, k, ..., r represent pixels on object 2 only, A, B, ..., E represent the common pixels shared by both objects, and empty grids are background. According to the algorithms described in Section 2, A is found as the first touching-point and h is the

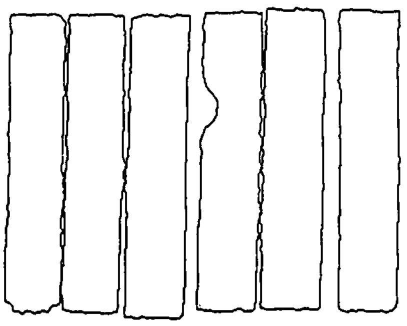
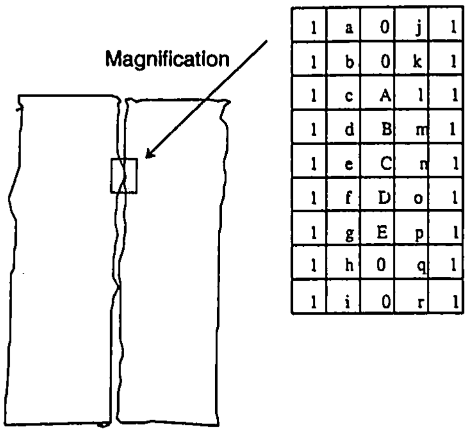


Figure 8. Boundaries extracted using the rule-based boundary extractor from the image shown in Figure 1.



1	a	0	j	1
1	b	0	k	1
1	c	A	l	1
1	d	B	m	1
1	e	C	n	1
1	f	D	o	1
1	g	E	p	1
1	h	0	q	1
1	i	0	r	1

Figure 9. A detailed look at a touching area.

resuming-point. By applying our boundary-extraction algorithms, A, B, C, f, g, h form part of the boundary at the touching area for object 1, and k, l, m, n, D, E, q form part of the boundary for object 2. Thus the maximum distortion to the boundaries of objects touching each other using the proposed boundary-extraction scheme is 1 pixel—which is acceptable in most applications, yielding a reasonably high precision. For example, in the biscuit inspection, this effect does not affect the decision whether to reject or accept a biscuit, since flaws such as nicks or bumps must have at least 6 pixels (3 mm, which is the required tolerance) deviation from the normal boundary width.

These results indicate that the proposed boundary-extraction scheme for touching objects is very promising. It cannot only provide least-distorted contours (even with some noise in the image), but can also execute at high speed. It is also able to distinguish objects from background details, such as a conveyor system.

4. Summary

This paper has emphasised the proposition that to develop a satisfactory boundary-extraction system, it is essential to incorporate the use of both domain-specific knowledge relating to the content of a scene, and general heuristics for edge detection which are independent of the class of scenes under analysis. To this end, we have discussed techniques for boundary extraction from images with non-ideal backgrounds and with touching objects, and concluded that a robust solution can be derived only with the aid of a priori knowledge of the scene.

To solve the major problems encountered in extracting desired boundaries of objects touching each other in a non-ideal background, a rule-based boundary extractor has been developed. Using a priori information about a scene, the knowledge-based algorithms are able to segment touching objects with acceptable distortion to the boundaries of both objects in contact, and are able to distinguish objects from a non-ideal background—demonstrated in the context of this present work by the conveyor system.

Acknowledgements

The authors are grateful to the referees for helping to improve the presentation of this paper. The authors also wish to acknowledge the support from ACME Directorate of the SERC of the UK.

References

Wu, Q.M. (1990). *Algorithms and Architectures for Industrial Visual Inspection*. PhD Dissertation, University of Wales, Department of Electrical and Electronic Engineering.

Yokio, S., J.I. Toriwaki and T. Fukumura (1975). An analysis of topological properties of digitized binary pictures using local features. *Computer Graphics and Image Processing* 4 (1), 63-73.