Intelligent Automation and Soft Computing, Vol. 8, No. 3, pp. 217-234, 2002

A NEW EFFICIENT EDGE EXTRACTION ALGORITHM FOR IMAGES USING DIRECTIONAL TRACING TECHNIQUES

S. S. IYENGAR
Louisiana State University
Dept. of Computer Science
Baton Rouge, LA 70803, USA

VIR.V. PHOHA
Louisiana Tech University
Computer Science
Ruston, LA 71272, USA

Y. WU
Louisiana State University
Dept. of Computer Science
Baton Rouge, LA 70803, USA

HLAMIN
Louisiana State University
Dept. of Computer Science
Baton Rouge, LA 70803, USA

ABSTRACT—The ability to recognize edges of an object is of fundamental importance in computer vision and image processing. In this paper, we present a fast, efficient method of edge extraction of images using directional tracing algorithm. Our work builds on the work of R. Nevatia and K.R. Babu [5], who use linear feature extraction to present algorithms for edge extraction. By employing an intuitive principle that an edge pixel should possess local maximum gradient, but having no more domain information or knowledge, we propose an effective, robust strategy to extract edges that integrates directional tracing and feature extraction. We discuss the computational efficiency of our approach and present a schematic for real time implementation of the algorithm. Application of our algorithm on different images shows excellent results. The main advantages of our technique include: (i) it can be used for a wide variety of images; (ii) it is simple and easy to implement; (iii) it is fast; and (iv) the algorithm is flexible to provide performance control.

Key Words: Edge Detection, Edge Extraction, Low Level Vision Processing, Image Segmentation

1. INTRODUCTION

Edge segmentation is a fundamental technique in image processing and computer vision. Research and development in this area has increased dramatically over the last twenty years. Even though the literature in this area is large, it is still an active and important research topic primarily because efficient edge segmentation is a key to many applications in computer vision, robotics, automatic manufacturing, and in many other fields.

Edge segmentation includes two basic tasks: *Edge Detection* and *Edge Extraction.* Edge detection means measuring the local 'edgeness' of each pixel in an image; then, based on the detected result, edge extraction is used to locate the possible edge pixels and link them together to form edges. Edges in an image are the contours that are recognized by the abrupt changes in grey values. Various computational

approaches for edge detection have been proposed [1-4]. But no matter what computation scheme is employed, we can generally consider that the result is some kind of a *gradient map* for describing the intensity-changing trend on the pixels of an image. We call the gradient map a *rough edge map* in this paper. Once a rough edge map is available, the important problem becomes how to extract the edge pixels out from the map and group them to get the edges. The existing techniques mainly include the following issues:

- *Thresholding*: In a real image, the great majority of non-edge pixels have non-zero intensity gradients, but their magnitudes usually are very small. Thresholding involves using a suitable selected threshold value [2, 12] for suppressing the noise effect of non-zero intensity gradients of non-edge pixels. It is a very practical strategy for separating edge pixels from a rough edge map, and it is used in many edge segmentation schemes.
- *Directional Local Maximum*: When a rough edge is produced by directional edge masks [1,2], we naturally assume that an edge pixel in the map is a local maximum along the gradient direction of this pixel, and an edge pixel can then be picked up by this maximum. Numerous approaches are derived from this intuitive idea [2, 4, 5].
- *Zero-Crossing*: For a step edge model, the second order derivative of an edge pixel along its gradient direction should be zero. For this reason, an edge point is called a zero-crossing point. If a rough edge map is generated by some second order difference operators, edges pixels can be isolated by zero-crossing criterion [3, 9].
- *Optimal Method*: This method concentrates on the optimal characterizations of edge detecting operators [4, 14, 15], so edge pixels are located under certain optimality in the presence of local noise.
- *Relaxation*: Starting from a rough edge map, a relaxation process [8, 13] uses iterated operation to reduce and eliminate the ambiguity of the 'edgeness' for the pixels by applying some knowledge rules and statistical inferences.

An evaluation and comparison on some existing edge extraction schemes is given in Table 1.

Table 1. A comparison on the selection of the edge extraction schemes.

Schemes	Prior Knowledge	Comments
Thresholding	Small intensity gradients on pixels are usually resulted by noise effect	An effective method for finding the candidates of edge pixels
Directional Local Maximum	An edge pixel possesses local maximal gradient	An intuitive and basic idea for edge extraction
Zero Crossing	Edges are the zero-crossing contours in a rough edge map	Closed edges are available; but their correctness and accuracy are relatively poor.
Optimal Method	A parameterized edge model in the presence of local noise	Good noise immunity; but the optimality depends on the domain assumptions
Relaxation	Some edge label structures to pairs of adjacent pixels	Capable of using sophisticated edge structures in the presence of noise; but the algorithm is time-consuming in execution

A brief review of additional edge detection and enhancement techniques follows. Jeong and Kim [16] propose a minimizing function over the entire image and use the relaxation method to solve a nonconvex optimization problem to estimate a unique scale for edge detection at each point in the image. Their method is adaptive and is constrained by the complicated shape of the objective function and the resulting sensitivity of the selected scale to the initial guess. Perona and Malik [20] propose an anisotropic diffusion network to enhance edges. Here space and time varying conduction coefficients, which are a decreasing function of the estimated gradient magnitude of the luminance function, determine the rate of diffusion at that point. Lindeberg [17] [18] proposed methods for selecting a local scale for edge detection based on maximizing a heuristic measure of edge strength.

Phoha and Oldham use the principle of competitive learning to develop an iterative algorithm [21] [22] for image recovery and segmentation. In this study, within the framework of Markov Random Fields (MRF), the image recovery problem is transformed to the problem of minimization of an energy function. A local update rule for each pixel point is then developed and shown to be a gradient descent rule for an associated global energy function. Simulation experiments using this algorithm on real and synthetic images show promising results in smoothing within regions and also in enhancing the boundaries. Elder and Zucker [19] use the knowledge of sensor properties and operator norms to minimize the reliable scale for local estimation at each point in an image. They apply this knowledge to detect and localize edges in images with shallow depth of field and shadows.

For edge extraction, the concept of *Directional Local Maximum* seems to be the most basic, intuitive idea. A similar role for edge detection is played by the concept of *Mask Matching*. We make a claim that different approaches for edge detection and extraction are just different formulations based on these two concepts. Also there is evidence that the mammalian visual system responds to edges through special low-level template-matching edge detectors [2].

In this paper, we describe an effective edge extraction approach using directional tracing. The development of our technique arose from the need of a fast edge-extracting algorithm for real-time stereo analysis. The zero-crossing method and the relaxation scheme are not good candidates for our requirements because edges derived from the former often suffer location errors, and the latter usually is time consuming. As for the optimal method, we know that generally, the analysis is in the continuous domain, and the optimality of a solution is based on the assumed conditions and criterion. Therefore, we try to find a better solution by using the basic approach of Directional Local Maximum.

When taking the edge as an image primitive for stereo analysis or any other use, we assume that the images possess rather sharp contour features. Intuitively, for edge masks the execution can be realized in real time by hardware [7]. If the edges are confined to straight lines, the technique presented by Nevatia and Babu [5] gives a simple, effective solution for our need in principle. But for the general situation, the various techniques summarized in [2] are not good enough for different reasons. In this paper, we extend Nevatia and Babu's approach [5] to a general situation.

The key problem in the extension is to find a way to effectively handle direction information. Nevatia and Babu [5] did not investigate whether their technique was useful for locating the position of an edge and used six 5×5 directional edge masks as edge detectors. However, we noticed that the masks of size 5×5 sometimes cause an edge to depart one pixel from its best position, even for straight lines. The 3×3 masks are a better choice computationally, and we adopted several measures to achieve comparably better and satisfactory results using 3×3 masks.

This paper is organized as follows: section 2 describes our algorithms in detail; section 3 contains experimental results and comparisons; and section 4 contains conclusions and future work.

2. A NEW TECHNIQUE OF DIRECTIONAL EDGE TRACING

Our edge extraction is a three-stage process:

(1) Production of a rough edge map;
(2) Determination of candidate edge pixels;
(3) Obtaining final edges.

A detailed description for each of the three stages follows in the below sub-sections.

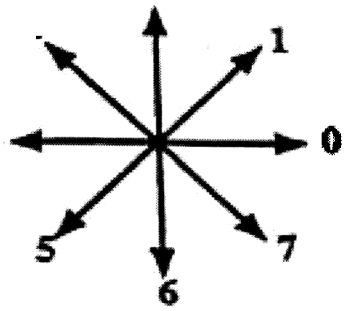
2.1 *Production of a Rough Edge Map*

In our technique, a rough edge map is produced by convolving an image with a group of 8 directional edge masks. For example, consider the following 3×3 Kirtch operators.

$$k_0 = \begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix} \quad k_1 = \begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix} \quad k_2 = \begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix} \quad k_3 = \begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix}$$

$$k_4 = \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix} \quad k_5 = \begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix} \quad k_6 = \begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix} \quad k_7 = \begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix}$$

$$d_0 = \begin{bmatrix} 1 & & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad d_1 = \begin{bmatrix} & 1 & \\ & 0 & \\ -1 & -1 & -1 \end{bmatrix} \quad d_2 = \begin{bmatrix} 1 & 0 & \\ 1 & 0 & \\ 1 & 0 & \end{bmatrix} \quad d_3 = \begin{bmatrix} 0 & & -1 \\ 1 & & -1 \\ 1 & & 0 \end{bmatrix}$$



by the symmetry, $d_{i+4} = -d_i$ for i = 0, 1, 2, 3.

There are different ways for assigning a direction feature to a pixel, both in continuous and discrete domains [2, 4, 5]. We have selected 8 directional masks since they naturally fit the popular eight-neighbor format of a digital image.

The proposed directional-tracing algorithm does not depend on any special edge detectors. It works based on a gradient map where each pixel has two interrelated components—its magnitude and direction. However, the selections of detectors do affect the tracing process in certain ways. Some of the effects are well known, such as the smaller masks are sensitive to noise whereas the larger masks cannot resolve fine details, and the bigger the size of a mask, the higher the cost of the convoluting computation. Since we consider edge points as those which possess directional local maximum in a rough edge map, it is important to know what are the effects on the position of the edge pixels by using different detectors. For instance, consider the ideal step edge pattern in Figure 2:

The edge on the pattern should be a subpixel line (the bold line on the figure) located between the two regions with the values 10 and 20 respectively. At pixel level, however, all the pixels along the ideal

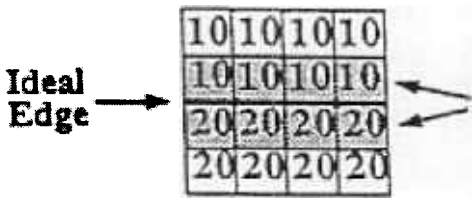


Figure 2. An ideal digital step edge pattern.

edge are the candidates of edge pixels in pixel level. More concretely, observe the convolution results in Table II by using Kirtch operators. We see that edge pixels in value 10 and 20 have their edgeness measure, (150,4) and (-150,0) respectively. So we get a two-pixel wide edge in pixel level. We can choose either of them to obtain a one-pixel wide edge with a deviation of half pixel from the ideal edge. If the above 3×3 difference operators are employed, only four rounds of convolving calculations are needed by the symmetry of the difference operators, but we can no longer distinguish the two types of edge pixels by direction information.

Table II. The convolution results by Kirtch operators for the edge pixels of Figure 2

	Convolution Value							
Kirtch Operators	k_0	k_1	k_2	k_3	k_4	k_5	k_6	k_7
Edge Pixel in Value 10	-90	-90	-10	70	150	70	-10	-90
Edge Pixel in Value 20	-150	-70	10	90	90	90	10	-70

On the other hand, the size of an edge detector also affects the position of edge pixels. More detailed discussion on this aspect will be presented in the following sections.

2.2 Determination of Candidate Edge Pixels

As presented in [5], for the straight lines in aerial images, an edge pixel should satisfy the following three criteria using the principle of local directional maximum:

1. The magnitude of the gradient at a pixel exceeds a given thresholding value T.
2. The edge magnitude at the pixel is larger than the edge magnitudes of its two neighboring pixels in the direction normal to the direction of this pixel.
3. The direction features of the two neighboring pixels are in the interval $[d - \pi/6, d + \pi/6]$.

Then, the pixel is considered a candidate of edge pixel; and the two neighboring pixels along the normal direction of this pixel are disqualified from being candidates of edge pixels.

In the general situation, however, we may miss some edge pixels by simply using these three criteria. For example, the two-pixel wide edge in Figure 2 cannot be identified by the criteria; on the other hand, because of the noise effects, different edge masks may have some convolution output on a pixel. If this output happens to be the magnitude of the pixel, there is no determined way to assign the pixel a correct gradient direction. Edge pixels, derived by the principle of local directional maximum, will be further judged in the following stage of directional tracing. It is usually easier to remove a poor candidate than to recover a missing edge pixel in the processing of directional tracing. We prefer to reserve as many candidates of edge pixels as we can in the stage of edge pixel extraction. Hence, we reformulate the above conditions 2 and 3 into 2' and 3' respectively as follows:

2′. Let m be the edge magnitude at the current pixel p; In the two normal directions relative to the directional feature of p, let the edge magnitudes of the two successive neighboring pixels be m_{11} and m_{12} along one direction, and m_{21}, m_{22} along the another. Then, p is a possible edge pixel if one of following conditions is held:

(a) $m > m_{11}$ and $m > m_{21}$;

(b) $m = m_{11}, m > m_{12}$, and $m > m_{21}$;

(c) $m = m_{11}, m = m_{21}, m > m_{12}$, and $m > m_{22}$.

3′. One of the direction features of the two neighboring pixels is not normal to d.

The following algorithm presents the process in detail.

Algorithm 1: *Edge_Pixel_Extraction*

Input : The rough edge map and the threshold value T. The rough edge map is presented by a magnitude array $M(u, v)$ and a direction array $D(u, v)$ with dimension $n_1 \times n_2$.

Output : A binary $n_1 \times n_2$ Edge map $E(u, v)$: where, if an entry $E[u, v]$ is an edge point, $E[u, v] = 1$, otherwise, $E[u, v] = 0$.

Procedure *Edge_Pixel_Extraction* (T, M, D, E, n_1, n_2)
begin

for $v \leftarrow 1$ **to** n_2
for $u \leftarrow 1$ **to** n_1
begin

$E[u, v] \leftarrow 0$; $m \leftarrow M[u, v]$; $d \leftarrow D[u, v]$;

if ($p \geq T$) **then case** ($D(u, v)$ **mod** 4 **of**

0 : $d_1 \leftarrow D[u, v-1]$; $d_2 \leftarrow D[u, v+1]$;
$m_{11} \leftarrow M[u, v-1]$; $m_{21} \leftarrow M[u, v+1]$;
$m_{12} \leftarrow M[u, v-2]$; $m_{22} \leftarrow M[u, v+2]$;

1 : $d_1 \leftarrow D[u+1, v-1]$; $d_2 \leftarrow D[u-1, v+1]$;
$m_{11} \leftarrow M[u+1, v-1]$; $m_{21} \leftarrow M[u-1, v+1]$;
$m_{12} \leftarrow M[u+2, v-2]$; $m_{22} \leftarrow M[u-2, v+2]$;

2 : $d_1 \leftarrow D[u-1, v]$; $d_2 \leftarrow D[u+1, v]$;
$m_{11} \leftarrow M[u-1, v]$; $m_{21} \leftarrow M[u+1, v]$;
$m_{12} \leftarrow M[u-2, v]$; $m_{22} \leftarrow M[u+2, v]$;

3 : $d_1 \leftarrow D[u-1, v-1]$; $d_2 \leftarrow D[u+1, v+1]$;
$m_{11} \leftarrow M[u-1, v-1]$; $m_{21} \leftarrow M[u+1, v+1]$;
$m_{12} \leftarrow M[u-2, v-2]$; $m_{22} \leftarrow M[u+2, v+2]$;

end (case)

if (d_1 or d_2 is not normal to d) **then**
begin
if ($m > m_{11}$ and $m > m_{21}$) $E[u, v] \leftarrow 1$;
if ($m = m_{11}$ and $m > m_{12}$ and $m > m_{21}$) $E[u, v] \leftarrow 1$;
if ($m = m_{11}$ and $m = m_{21}$ and $m > m_{12}$ and $m > m_{22}$) $E[u, v] \leftarrow 1$;
end

end (for)
end (*Procedure*)

2.3 Obtaining Final Edges

Directional tracing uses the relationship among edge pixels to combine them together. This method has been successfully applied for extracting straight lines in aerial images [4, 5]. We extend the technique to a more general situation.

There are two sets of functions in the edge tracing process. One is to confirm the good edge points and remove the poor ones in *E (u,v)* derived by Algorithm 1. Another is to label the related edge points to form an integrated edge. In this paper, we only consider the first function. The approach proposed by [5] fulfills the two functions in one execution for straight lines. However, if the edges are not straight lines, we have sufficient reasons to separate the two functions in two executions step by step. For example, by the first function, we need to remove some poor edge segments; by the second function, we need to connect some separate edge segments together. If the edges in consideration possess different shapes and not just straight lines, then we may get confused in some edge segments due to the two conflicting needs.

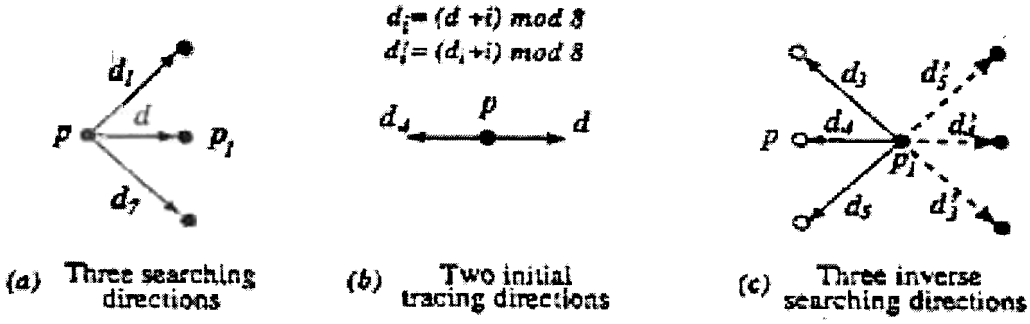


Figure 3. Illustrations on tracing directions.

Our tracing strategy is intuitive and simple. Suppose a non-isolated edge point $\vec{p}_s$ in *E(u,v)* is selected as the starting point. Let $\vec{p} \leftarrow \vec{p}_s$ and d be the direction of $\vec{p}$. Then, in the searching order of d, $d_1 = (d + 1) \bmod 8$ and $d_7 = (d + 7) \bmod 8$ as shown in Figure 3-(a), if there is a neighboring edge point $\vec{p}_1$ of $\vec{p}$ in one of the three directions, link $\vec{p}_1$ to $\vec{p}$; then, let $\vec{p} \leftarrow \vec{p}_1$ and repeat the same procedure for the new $\vec{p}$. When the first tracing procedure terminates, as illustrated in Figure 3-(b), another tracing process will be started from $\vec{p}_s$ along the direction, which is opposite to the original direction of $\vec{p}_s$. The edge segment passing through $\vec{p}_s$ is then the combination of the two tracing results.

In the presence of noise, we should notice that, when $\vec{p}_1$ is linked to $\vec{p}$, the direction feature of $\vec{p}_1$ may point back to $\vec{p}$ as shown in Figure 3-(c), where the direction feature d of $\vec{p}$ comes from $\vec{p}$ to $\vec{p}_1$, but the direction d' of $\vec{p}_1$ may take the values of $d_3 = (d + 3) \bmod 8$, $d_4 = (d + 4) \bmod 8$, or $d_5 = (d + 5) \bmod 8$. For these cases, the tracing direction should be reversed to the opposite direction $d'_i = (d_i + 4) \bmod 8$, $(i = 3, 4, 5)$.

Algorithm 2 is the formal description of this tracing strategy.

Algorithm 2: *Directional_Tracing*

Input : The edge map $E(u, v)$ produced by Algorithm 1 and the direction array $D(u, v)$ of the rough edge map, where the dimensions of $E(u, v)$ and $D(u, v)$ are $n_1 \times n_2$. The starting point $\vec{p}_s = (us, vs)$ is a known edge point in $E(u, v)$.

Output : Two chain codes *cc*1 and *cc*2 for recording the tracing results.

```
Procedure Directional_Tracing ( E, D, LT1, us, vs, cc1, cc2 )
begin
    u ← us; v ← vs; length ← 0; d ← d1 ← D[us, vs]; status ← tracing;
    while( length < LT1 )
    begin
       if ( ((d1 + 3) mod 8)=d ) d ← (d + 4) mod 8;
       if ( ((d1 + 4) mod 8)=d ) d ← (d + 4) mod 8;
       if ( ((d1 + 5) mod 8)=d ) d ← (d + 4) mod 8;

       Starting from the current point [ u, v ], assign p1, p2 and p3 the next values of
       initial edge map E along the directions d, d1 and d7 respectively

       if ( p1=0 ) then if ( p2>0 ) d ← (d + 1) mod 8;
                        else if ( p3>0 ) d ← (d + 7) mod 8;
                        else status ← terminal;

       if ( status = terminal ) then
       begin
          if ( it is in the second tracing process ) then Return;
          if ( it is in the first tracing process ) then
          begin
             u ← us; v ← vs; length ← 0;
             d ← d1 ← D[us, vs]+4 mod 8; status ← tracing;
          end
       end
       else
       begin
          move the point [ u, v ] forward a step along the direction of d

          if ( it is in the first tracing process ) then cc1[length] = d;
                                                   else cc2[length] = d;

          length ← length + 1; d1 ← d; d ← D[u, v]
       end
    end ( while )
end ( Procedure )
```

The parameter LT_1 for the procedure *Directional_Tracing* ( ) has a control threshold to interrupt the tracing process from falling into a dead loop. Depending on the size of an edge map, an arbitrary large number can be applied for this purpose. In fact, the chances that the tracing process falls into a dead loop

must be very few, and so we keep LT_1 there just as a preventive measure. However, to judge a segment returned by the procedure, we need another control threshold LT_2. If the length of the returned segment is shorter than LT_2, we will refuse to accept it as an edge segment. This measure is based on the observation that a noise segment usually is much shorter than an edge segment.

The whole tracing process for refining an edge map E is presented in the procedure *Tracing* ( ) as below.

```
Procedure Tracing ( E, E', D, LT1, LT2, n1, n2 )
Declaration :
        E[u, v] is a n1 × n2 edge map generated by algorithm 1
        E'[u, v] is a n1 × n2 array for storing the refined edge map
        D[u, v] is the n1 × n2 directional feature matrix.
        LT1 , LT2 are two control parameters
begin
    for v ← 1 to n2
    for u ← 1 to n1
    begin
        if ( E[ u, v ] is a non-isolated edge point in E(u, v) )
        begin
            Directional_Tracing (D, E, u, v, LT1,cc1, cc2);
            If ( the length sum of cc1 and cc2 is larger than LT2 )
            begin
                Record cc1 and cc2 into E' and remove them from E
            end
        end
    end (for)
end (Tracing).
```

2.4 *The Computational Efficiency of Our Technique*

The effectiveness of our method on edge extraction is presented by the experimental investigation in section 3. In this section, we discuss the computational efficiency of our technique.

In principle, the time complexity of an edge extraction algorithm is $O(n_1 \times n_2 \times n_3)$ for a digital image with size $n_1 \times n_2$ is included because the algorithm must scan every pixel in an image to find its edgeness; factor n_3 corresponds to the complexity of the processing procedure on a pixel. So, ideally we hope to achieve $n_3 = 1$. In other words, we should simplify the processing procedure on a pixel a much as possible.

For the aforementioned edge extraction schemes in section 1, the approaches of zero-crossing and optimal method need to employ edge detecting operators in various scales for edgeness analysis; relaxation and many previous directional tracing algorithms require certain complicated searching procedures in finding an edge pixel. These methods are more expensive than Nevatia and Babu's method [5] from the viewpoint of computational efficiency. But the application of the latter is confined to straight lines.

As a general extension of [5], the algorithms presented in the above sections maintain the simplicity by their straightforward and determined style. In algorithms 1 and 2, we intentionally ignore carrying out a complex searching procedure in some big and complete search space but specify the searching and comparison in a limited and determined range to decide whether an image pixel is an edge pixel or not. The

algorithm is fast in execution and the experimental results show that our idea works well. Notice that for the procedure *Tracing* ( ) in section 2.3, if a traced segment is not judged long enough by the control parameter LT_2, we keep the segment in E (u, v). This will cause a noise segment being traced up to the times of its length, but the measure can prevent removing an edge segment when a tracing procedure begins at an unsuitable starting point. There is no significant effect on the algorithm execution of this measure, and it is worth paying the price for our main purpose.

Our method can be implemented in hardware for real-time performance; parallel-processing techniques can be applied to improve performance in software. Figure 4 presents one possible implementation. In Figure 4, and in the following sections, we name *Initial Edge Map* and *Final Edge Map* for the results obtained by successive processes of thresholding and tracing on a rough edge map respectively.

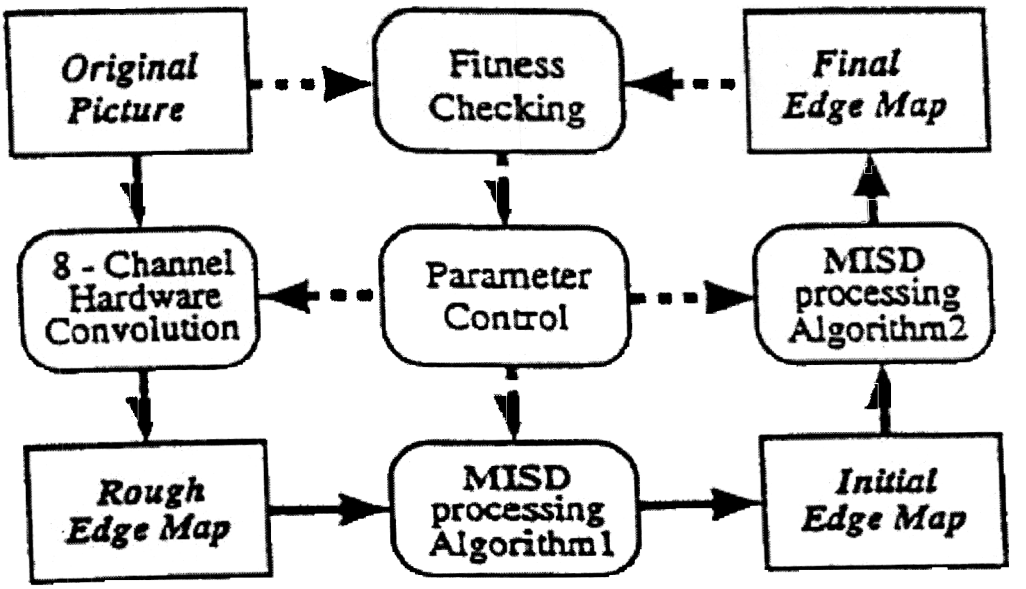


Figure 4. A real time implementation scheme for our method.

3. EXPERIMENTAL RESULTS

3.1 Background

In the evaluation of the techniques on edge detection and extraction, two common criteria are used: the first requirement is that of a high rate in finding a real edge and low rate in producing a fault edge; the second is the correctness in the position of the found edges. However, there is no formal way to asses the effectiveness of the two criteria, except perhaps by a visual inspection, and so we present results of our approach using a visual inspection.

Figure 5 illustrates the process of our methods. In (b), a rough edge map is generated by 3x3 difference masks on the original picture (a). After the thresholding process using a threshold value of T = 8, we got (c) from (b); then (c) was refined by the directional tracing to (d). Finally, we superimposed (d) on (a) and checked if the obtained edges fit the original picture correctly.

Three factors that affect the performance of our approach are the edge detectors, the threshold value T of algorithm 1, and the length control parameter LT_2 of algorithm 3. We discuss their effect in the following three sections.

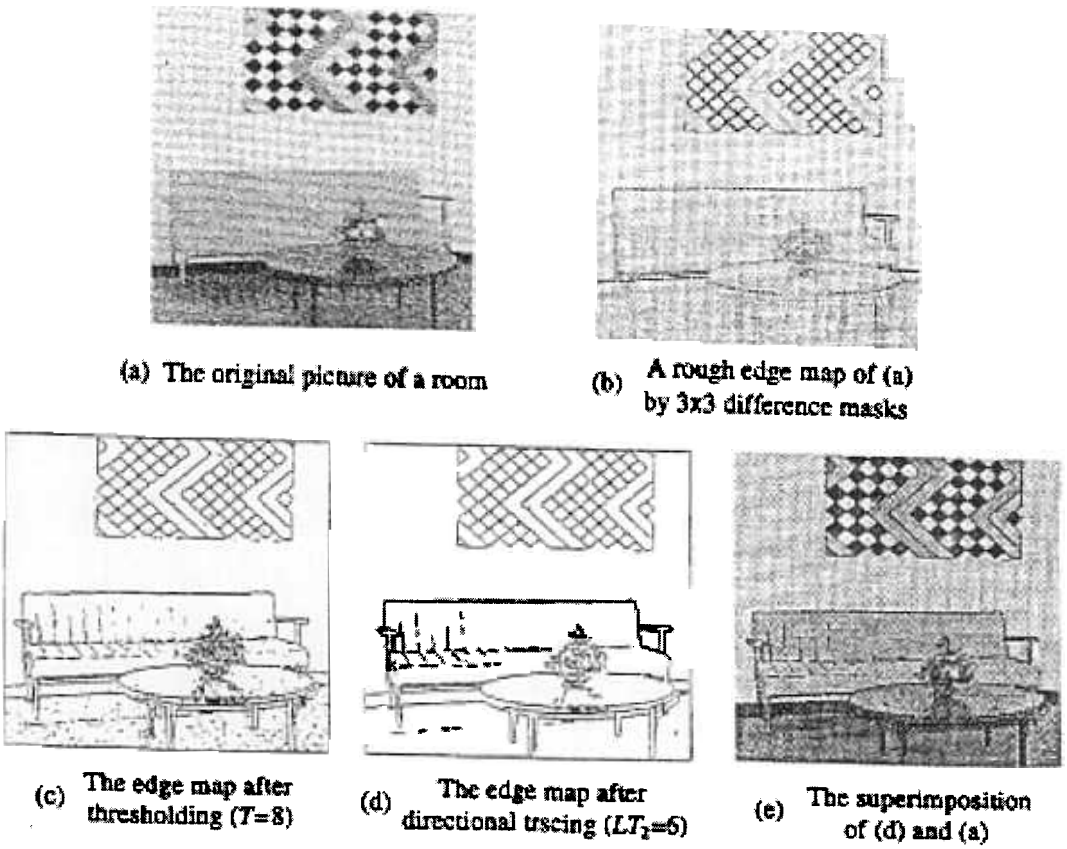


Figure 5. The process of edge detection, extraction and evaluation.

3.2 Mask Effects

Edge detecting masks vary in their patterns and sizes. Although different masks certainly produce different rough edge maps, we use only one intuitive rule—the principle of local directional maximum—for generating an initial edge map from a rough edge map. Whether the principle is efficient enough for picking edge pixels up is the key to the effectiveness of our technique. Our experiments on this problem are shown in Figures 6 and 7.

The original picture is shown in Figure 6-(a). We used four different mask sets to detect its edges. Two sets of masks are Kirtch operators and difference operators as described in section 2.1. One of the other two sets of masks is 3×3 Robert operators, another is the extension of the Robert Operators to a 5×5 version. They can be derived by rotating r_0 and R_0 defined respectively as follows:

$$r_0 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad R_0 = \begin{bmatrix} 1 & 2 & 2 & 2 & 1 \\ 1 & 2 & 2 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -2 & -2 & -2 & -1 \\ -1 & -2 & -2 & -2 & -1 \end{bmatrix}$$

We can see from (d) to (i) in Figure 6 that different 3×3 masks produce almost the same results. In fact, when the masks have the same size, we can expect that, if a pixel is a local directional maximum under one of the three 3×3 mask sets, it should also be the local directional maximum to the other two mask sets. However, because different mask patterns give out different magnitude output, some pixels will

be removed by thresholding under this mask set but not by the others. At this time, differences may occur. For example, the results for Difference and Robert operators in Figure 6 are obtained by threshold value 8, but those for the Kirtch operator are obtained by threshold value 12. The result 6-(c) comes from the 5×5 masks. It seems to have little noise.

For checking the fitness of the edge map with the original picture, we superimpose the edge maps 6-(i) and 6-(c) on 6-(a), and the composed pictures are displayed in Figure 7. We see that the two edge maps fit the picture nicely. Notice that in Figure 7, the lady's necklace and dimples near her mouth look wider in (b) than in (a), and the same thing can also be observed in Figure 6. By more precise investigation in pixel level, we can say that an edge being drawn by a human being on a digital picture is more like the one derived by 3×3 masks but not that by 5×5 masks. Generally, this kind of location deviation is 1-pixel distance, and it occurs when some edge segments are close or near.

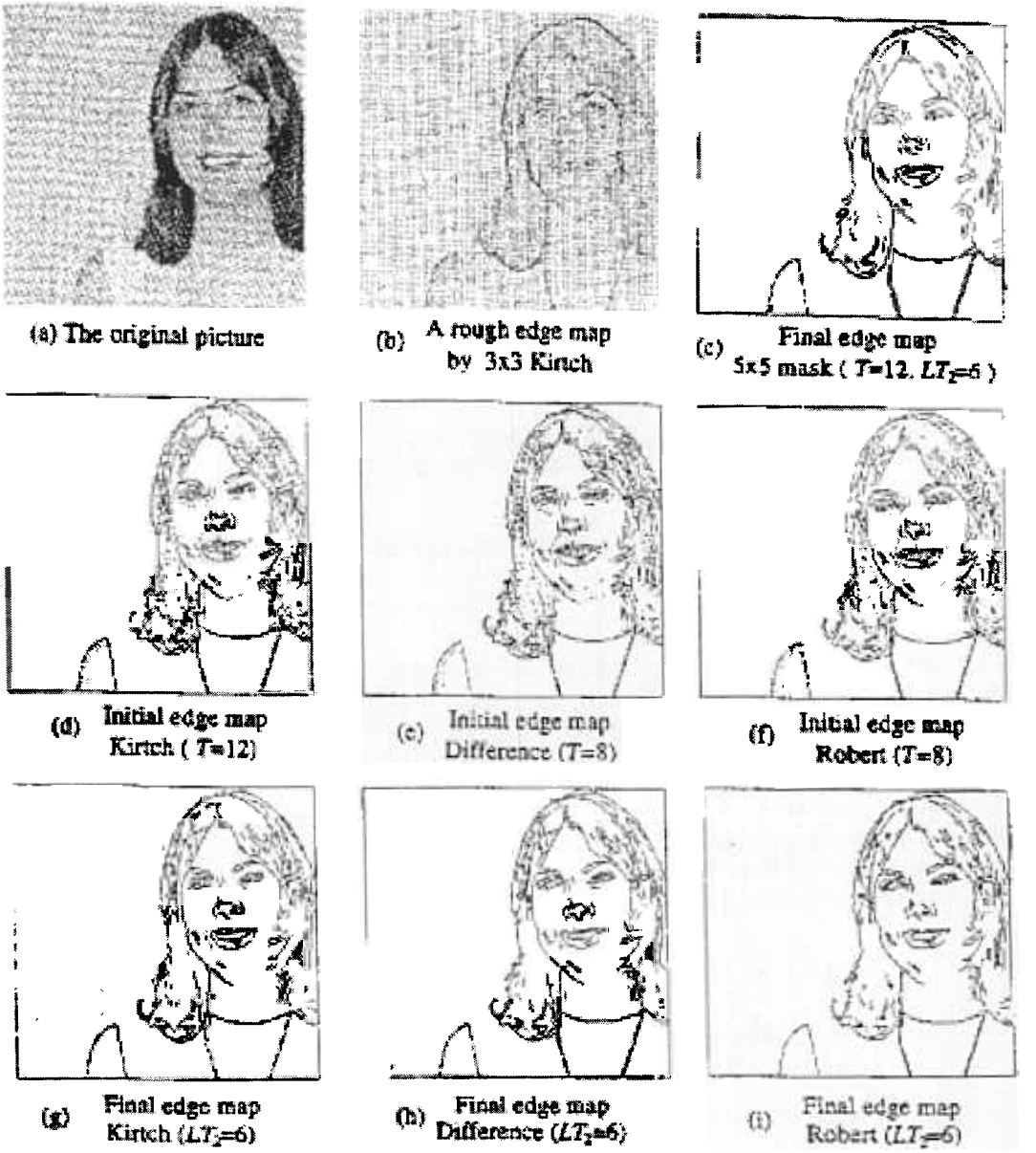
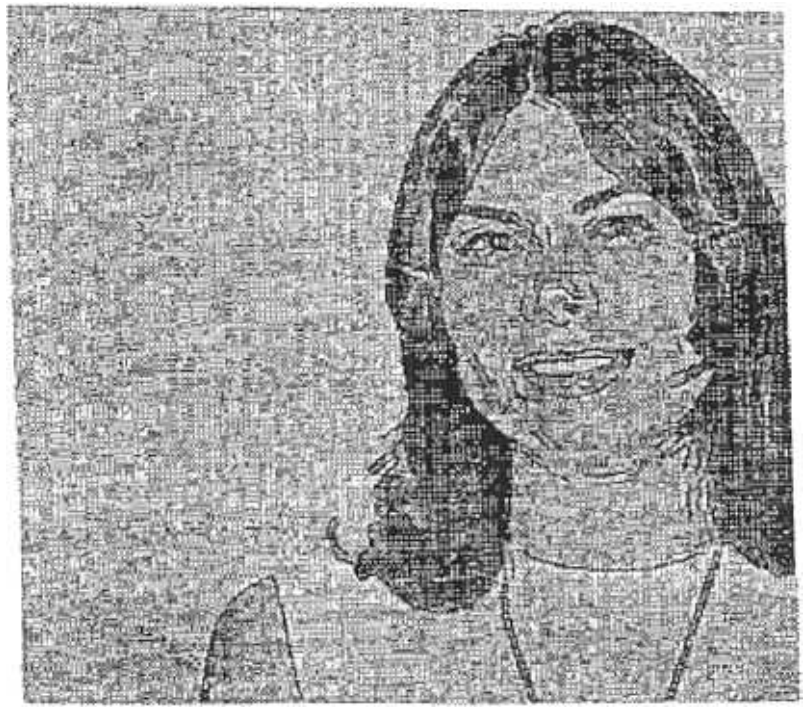


Figure 6. A Comparison on the different edge detecting masks (T is the threshold value, LT_2 is the tracing control parameter).

(a) Superimposition of 6-(i) on 6-(a)



(b) Superimposition of 6-(c) on 6-(a)

Figure 7. The edge fitness of 3x3 mask vs. 5x5 mask.

From this observation, we can conclude that the bigger the size of masks, the bigger the chance of error on the edge location. An edge location can be obtained in pixel level by using 3×3 masks. The edge location is very good in pixel level. Also, our algorithm is mask insensitive when a suitable threshold value is selected. On the other hand, we noticed that when 3×3 operators missed a edge segment, which occurred on a man-made edge map, in most of the cases, if we isolated that area from the picture and asked another person coming to judge the area, the person would think that there is no edge. For example, the part of the lady's right shoulder in Figure 6 is such a case. Therefore, we can simply depend on the edge-finding rate of the edge detectors. But the final selection of an edge pixel is also decided by the two control parameters T and LT_2 in algorithms 1 and 2.

3.3 Threshold Control and Tracing Control

The performance control of our technique is determined by the threshold T in algorithm 1 and the parameter LT_2 in algorithm 3. Because we try to let our method be applied to a wide variety of images, there is no determined way, or it is impossible sometimes, to have a way for finding a uniform T or LT_2 for our algorithms. However, when using the algorithms to some specific or for some special purpose, we can adjust the two parameters for achieving better performance. In Figures 5 and 6, we see that the two control factors are very efficient for separating the edge pixels and the random noises. Because of their combined effect, the two parameters can be set in small values that work well. Figure 8 illustrates the effects of the parameters T and LT_2 in detail. In the picture (a) of Figure 8, the background behind the car is grass. The grass may be considered as noise. However it is difficult to remove this kind of irregular texture noise. For our algorithm, we show in Figure 8 that either of the two parameters can serve for simplifying an edge map, and their combination would be more powerful.

Instead of directly removing an edge segment upon the threshold of the two control parameters, a noticeable idea is to label an edge segment with these two parameters. For our original interest of stereo analysis, suppose we need to keep as many candidate edge segments as we can, and we also need to classify edge segment's feature in some priority for processing, we can simply give an order to the nine situations of (d) to (s). Then, each edge segment can be assigned to an order number. The assignment processing do not need extra time but only work in passing during our tracing period. Then, stereo analysis can go on in order.

3.4 More Examples

More application examples of our algorithms are given in Figure 9. Each of the three results is generated by 3×3 masks and difference operators T =8 and LT_2 = 6. We see that, for different kinds of images, when the contour features are formed by sharp changes of intensity, the results of our algorithm are satisfactory. However, if the edges themselves are not sharp enough, our method does not work well. This is the typical limitation of our method.

In this situation, increasing the size of edge detectors may lead to better results, but we still have no information for locating the correct position of an edge.

Note that we usually expect that an edge or edge segment is single-pixel wide. For our tracing strategy, the situation described in Figure 10 will result in double-pixel wide edges. In the figure, all the darkened pixel positions will be determined as edge pixels by our algorithms. There is no problem in changing such an edge to a single-pixel wide one, but this will take some execution time. It could not be figured out if their presence leads to a negative effect, hence it has been included.

4. CONCLUSIONS AND FUTURE WORK

We have presented a simple and efficient technique of edge extraction. Our approach depends on the intuitive and basic ideas of edge detection and extraction. Application of our technique on several images and our experiments show excellent results. The main advantages of our method are summarized as follows:

- Applicable to a wide variety of images
- Computationally easy
- Fast execution
- Flexible performance control

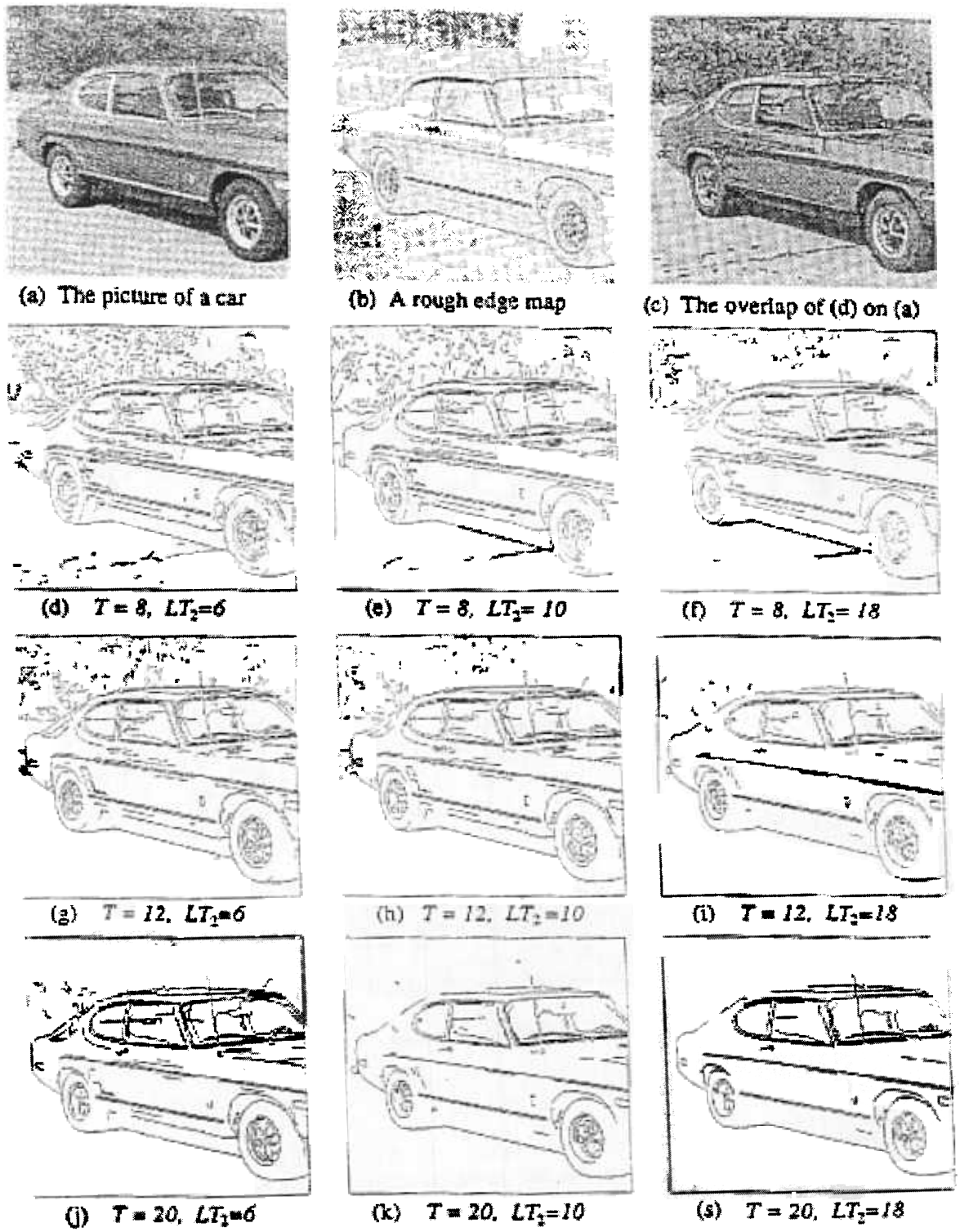


Figure 8. The effects of the parameters T and LT_2.

Our technique can be implemented in parallel and can be implemented in hardware for real time performance.

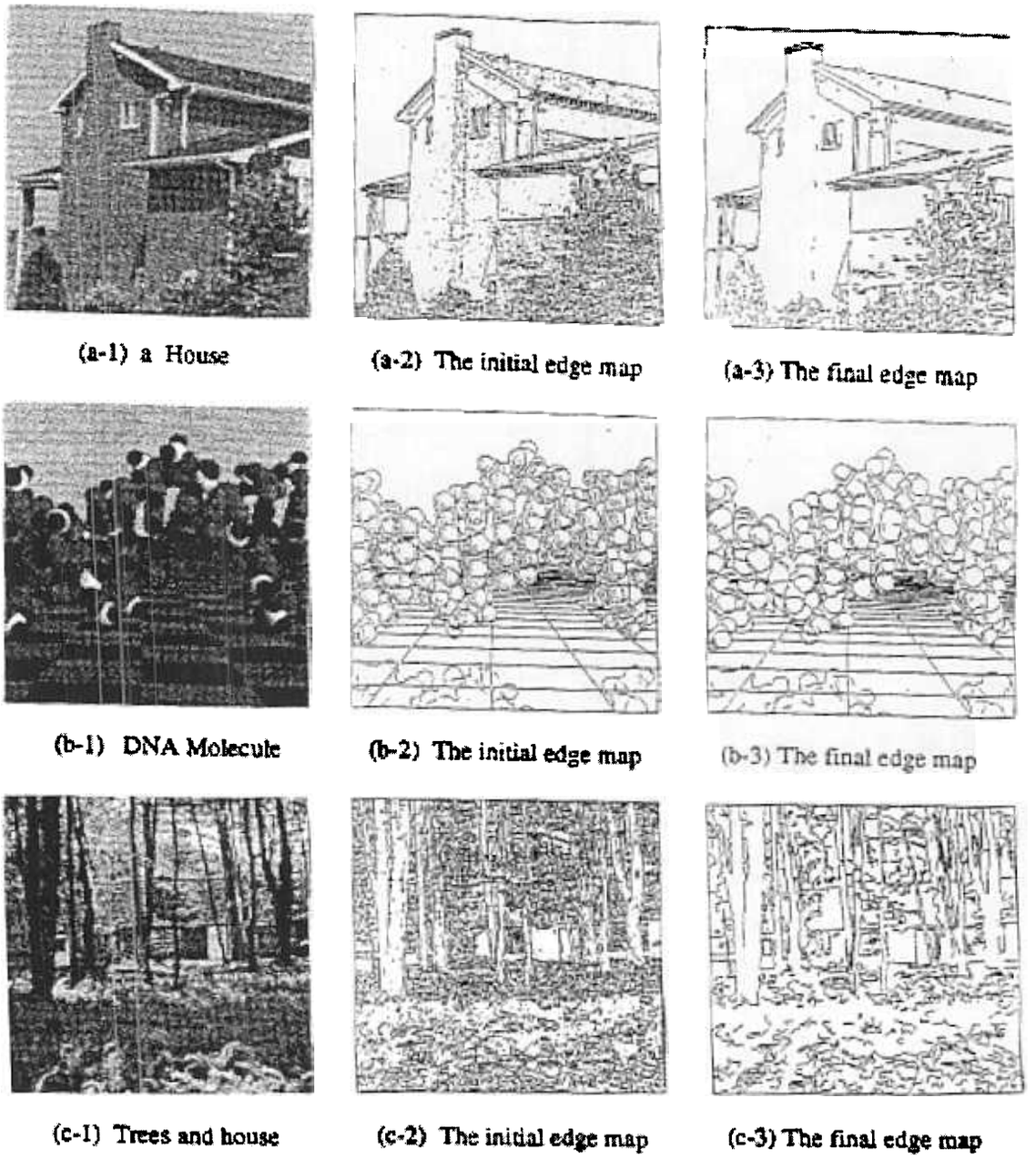


Figure 9. More Examples.

As mentioned in section 2.3, in this work we focused on using directional tracing for the purpose of confirming edge point and removing noise. We can extend this technique to edge linking by searching and following an edge segment pixel by pixel; linking will extend the searching range across the terminal of an edge segment and connect two segments together according to a certain criteria. Notice that our present approach is domain independent. It can be further improved by edge linking and combining with approaches [10, 11]. We are currently working to add domain related information and knowledge-based guidance in our approach.

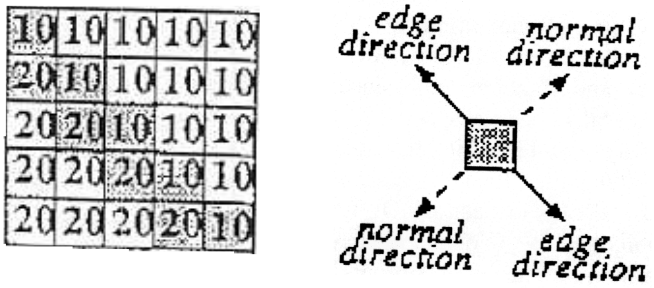


Figure 10. A double-wide edge.

5. ACKNOWLEDGEMENTS

The authors would like to thank Sandi Brown for a careful review of the paper and for many insightful comments to improve the presentation. The authors would also like to thank Amit Nadgar and Sunil Babu for help in the preparation of this document.

REFERENCES

[1] W.K. Pratt, Digital Image Processing, Wiley, New York, 1978.
[2] D.H. Ballard and C. Brown, Computer Vision, Prentice-Hall, Englewood Cliffs, NJ, 1982.
[3] D. Marr and E. Hildreth, "Theory of Edge Detection", Proc. R. Soc. Lond. B 207, pp.187-217, 1980.
[4] J.Canny "A Computational Approach to Edge Detection", IEEE Trans. on PAMI, Vol.8, No.6, pp.679-698, 1986.
[5] R. Nevatia and K.R. Babu, "Linear Feature Extraction and Description", CGIP, Vol.13, pp.257-269, 1980.
[6] V. Venkateswar and R. Chellappa, "Extraction of Straight Lines in Aerial Images" IEEE Trans. on PAMI, Vol.14, No.11, pp.1111-1114, 1992.
[7] M.K. Selmana and C.R. Allen, "Implementation of a Fast Programmable Edge Detection Preprocessor", PRL, Vol., p.423-427, 1983.
[8] E.R. Hancock and J. Kittler, "Edge-labeling Using Dictionary-based Relaxation", IEEE Trans. on PAMI, Vol. 12, No.2, pp.165-181, 1990.
[9] R.M. Haralick, "Digital Step Edges from Zero Crossings of Second Derivatives", IEEE Trans. on PAMI, Vol.6, pp.58-68, 1984.
[10] M. D. Levine, and A. M. Nazif, "Low Level Image Segmentation: An Expert System", IEEE Trans. on PAMI, VOL. 6, pp.555-577, 1984.
[11] A.C. Sleigh, "The Extraction of Boundaries Using Local Measures Driven by Rules", PRL, Vol. 4 pp.247-258, 1986.
[12] M.K. Kundu and S.K. Pal, Thresholding for Edge Detection Using Human Psychovisual Phenomena", PRL, Vol.4 pp.433-441, 1986.
[13] Wein Deng and S.S. Iyengar, "A New Probability Relaxation Scheme and Its Application to Edge Detection", Research Report, Robotics Research Laboratory, Department of Computer Science, Louisiana state University Baton Rouge, 1983.
[14] W-H.H.J. Lunscher and M.P. Beddoes, "Optimal Edge detector Design I: Parameter Selection and Noise Effects", IEEE Trans.on PAMI, Vol.8, pp.164-177, 1986.
[15] W-H.H.J. Lunscher and M.P. Beddoes, "Optimal Edge detector Design 2 : Coefficient Quantization", IEEE Trans.on PAMI, Vol.8, pp.178-187, 1986.
[16] H. Jeong and C. Kim, "Adaptive Determination of Filter Scales for Edge Detection," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 14, no. 5, pp. 579-585, 1992.
[17] T. Lindberg, Scale-space Theory in Computer Vision. Dordrecht, The Netherlands: Kluwer, 1994.

[18] T. Lindberg, "Edge Detection and Ridge Detection With Automatic Scale Selection," IEEE Conf. Computer Vision Pattern Recognition, pp. 465-470, San Francisco, IEEE CS Press, June 1996.

[19] J. Elder and S. Zucker, "Local Scale Control for Edge Detection and Blur Estimation," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 20, No. 7, pp. 699-716.

[20] P. Perona and J. Malik, "Scale-Space and Edge Detection Using Anisotropic Diffusion," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 12, pp. 629-639, 1990.

[21] V. Phoha and W. Oldham, "Image Restoration and Segmentation using Competitive Learning in a Layered Network". IEEE Transactions On Neural Networks, pp. 843-856, Vol. 7, No. 4, July 1996. (See also V. Phoha and W. Oldham, Corrections to "Image Restoration and Segmentation using Competitive Learning in a Layered Network" the IEEE Transactions On Neural Networks, Vol. 7, No. 6, November 1996.)

[22] V. Phoha and W. Oldham, "Image Restoration and Segmentation using Competitive Learning" in proceedings of the Neural and Stochastic Methods in Image and Signal Processing II in The International Symposium on Optics, Imaging, and Instrumentation at The International Society for Optical Engineering (SPIE), Annual Meeting, San Diego, July 12-13, 1993.