Philips Research Laboratory Brussels
Av. E. Van Becelaere 2, Box 8
B-1170 Brussels, Belgium

Report R.482

# BLOOD VESSEL DETECTION THROUGH RIDGE SKELETONIZATION

Christian Ronse

December 1984

**Abstract:** *This report describes the work done in PRLB towards blood vessel detection and enhancement in digital subtraction angiography.*

*We work with digitized X-ray images and we assume that the blood vessel patterns correspond to ridges in the grey level of these images. We describe 3 algorithms for the detection and enhancement of ridges:*

- *The first one builds the skeleton of the ridges of the image.*
- *The second one filters noisy portions of that skeleton.*
- *The third one builds an enhanced representation of the ridges of the original image by subtracting the background grey level from the ridge grey level in the neighborhood of the skeleton.*

*Technical details, including computer programs, will be dealt with in a Technical Note.*

# I. Introduction

In recent years the field of blood vessel diagnostic has seen the development of a method called *Digital Subtraction Angiography* (in brief, *DSA*) which produces enhanced images of blood vessels. It consists in taking two X-ray images of the portion of the patient body under analysis, where the second exposure is preceded by the injection of a contrast-enhancing medium in the blood system; then a subtraction of the two images should yield an image representing the blood vessels.

This method involves two problems: the need of a high dosage of the contrast-enhancing product and the patient motion between the two exposures (which leads to errors in the blood vessel detection).

A new direction for DSA is to make only one X-ray exposure (the post-injection one), and to compute from it the pre-injection mask with the use of pattern recognition methods. When the contrast is high enough, this is possible by a digital enhancement of blood vessel-shaped patterns. Although several methods exist for this purpose, the CT team at PFH has developed a new one called *pseudomask*, which is a nonlinear processing of the median-filter type. Early work on this field has been reported in [2] (see particularly the Section 2.2 in it); the present state of the technique can be found in [1], where the efficiency of the pseudomask is compared to that of other methods.

Let us describe it briefly. We assume that the grey level of blood vessels is higher than that of the background. Given an input image $I$, one computes from it a background mask $I'$ by replacing in it the grey level of each pixel $p$ by a low rank (for example the minimum) grey level in a given window $W(p)$ around that pixel. A simple choice for $W(p)$ is described in [1]. For every pixel $p$, $I'(p)$ represents then an approximation of the background grey level in the vicinity of $p$. Applying a smoothing operator to $I'$ one gets the *pseudomask image* $I''$. Then the blood vessel features of $I$, forming ridges in it, will be erased from $I''$; thus one gets an image of the blood vessels by taking $I - I''$ (where the $-$ represents the pixelwise grey level subtraction between the two images).

This method is very easy, but it can have some defects. For example, the pseudomask subtraction $I - I''$ produces noisy ridges at places where there is a ramp in the grey level in $I$. The simplest solution to this defect is to require that for a pixel $p$, the low rank grey level $I'(p)$ must be attained on both sides of $p$ inside the window $W(p)$.

Some other refinements of this method have been envisaged:

(1) One can restrict the pseudomask to blood vessel locations.

(2) The width and orientation of the window $W(p)$ can be made adaptive in function of an approximation of the actual width and orientation of the vessel at $p$.

(3) Different variants of the pseudomask can be applied in succession.

We have made some work on the refinements (1) and (2), and the purpose of this

report is to describe our methods and results. A brief summary of our work can also be found in Section III of [1]. Basically, we have designed and tested three algorithms (ridge skeletonization, skeleton filtering, skeleton-controlled pseudomask). Let us explain here how they can be related to (1) and (2):

In order to implement (1), one must produce a set of pixels corresponding to the location of blood vessels. This set is called the *skeleton*. Now (2) requires that one associates to each skeleton pixel $p$ a width and an orientation for the window $W(p)$. They will be called the *local width* and the *local orientation* of the pixel $p$. The determination of these features is the purpose of the first algorithm that we have designed and implemented. Assuming that the grey level of blood vessels is higher than that of the background, blood vessels will form ridge patterns in the image. Our algorithm produces the skeleton of the ridges in the image and associates to each skeleton pixel a local approximative width (2, 4, 8, ...) and a local approximative orientation (N, NW, ...).

As the skeleton is produced by local methods, it contains portions which are too short in comparison to their width. We have thus designed and tested a second algorithm, which filters the skeleton in order to eliminate such noisy portions.

Finally a third algorithm implements a variant of the pseudomask to the original image, restricting it to the pixels of the the filtered skeleton and using the associated local width and orientation in order to determine the size and orientation of the pseudomask windows.

To these three algorithms correspond the Chapters II, III and IV of this report. Let us describe them with more details:

### Ridge skeletonization

According to whether blood vessels have higher or lower grey levels than the background, they correspond to *ridge* or *valley* patterns respectively. We will assume that their grey levels are higher, and so restrict ourselves to ridges. The *skeleton* of such a pattern consists in a union of digital lines locating all center points of these ridges. To each skeleton pixel one associates a local *width* and *orientation*. We will give only a very rough estimation of these two parameters: the local width will be approximated by a power of 2 and the local orientation by one of the 4 basic orientations modulo 180° forming with the horizontal an angle multiple of 45°, as shown in Figure 1.

There are three reasons for such a rough approximation. First, this skeleton and these two local parameters will be used mainly for the control of the pseudomask operator, and so they do not need to be precise: the details will be obtained by the pseudomask. Second, in many cases it is difficult to locate a detected feature accurately, and the edges of a ridge-shaped pattern may be fuzzy. Third, this approximation simplifies the ridge detection algorithm and the control of the pseudomask.

The skeleton and the associated local widths and orientations will be found by

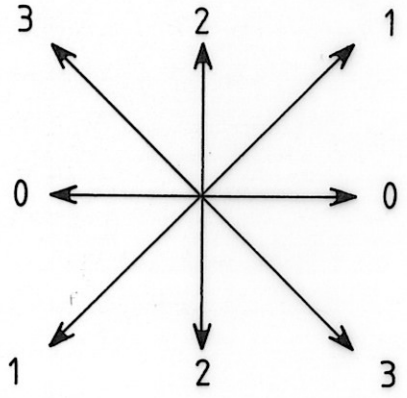



**Figure 1.** *The 4 basic orientations modulo* 180°.

convolving the input image with various windows corresponding to all orientations and sizes, and then using the resulting values to select, through a succession of eliminations of candidate skeleton pixels by local criteria, the appropriate skeleton pixels and their local width and orientation.

### Skeleton filtering

Having found the skeleton of the ridges by local methods, there still remains the problem that certain pixels, while satisfying local properties of ridge points, actually do not fit in globally recognizable ridge-shaped patterns. Indeed, local criteria are unsufficient for the definition of such a pattern.

We have made some steps toward more global methods with a skeleton filtering algorithm which combines a local and regional analysis for the selection of skeleton pixels corresponding to actual ridge points.

### Skeleton-controlled pseudomask

When the filtered skeleton is at hand, together with the local widths and orientations, one can perform on the original image a pseudomask operation restricted to the neighborhood of the filtered skeleton. One obtains thus a pseudomask image to be subtracted from the original one in order to obtain the blood vessels. This pseudomask image is built iteratively; one associates to every skeleton pixel $p$ a window $W(p)$ forming a line segment centered on it, whose length is equal to twice the local width on $p$ and whose orientation is perpendicular to the local orientation on $p$; then he changes the grey level values of the pixels of $W(p)$ in the pseudomask image according to certain rules. At places where the skeleton ends or where there is a change in the local width or orientation, discontinuities appear in the set of pseudomask windows $W(p)$, and so one must associate pseudomask windows of decreasing size to some pixels neighboring the skeleton in order to ensure a smooth transition in the pseudomask image.

The contrast in the resulting subtraction image showing blood vessels can be en-

hanced by maximizing the dynamic range, either linearly or rather proportionally to the square root (in order to improve low grey levels).

### Efficiency of the algorithms

Let us discuss briefly the efficiency, quality and speed of our algorithms. Experiments show that most image features which do not correspond to blood vessels are eliminated from the pseudomask subtraction image. In particular, the defect mentioned for the uncontrolled pseudomask, *i.e.*, that grey level ramps in the input produce ridges in the output, does not appear here. Moreover, the noise level in the resulting images is relatively low. Image features which do not correspond to blood vessels but nevertheless appear in the output of the pseudomask subtraction image, form generally fuzzy elongated shapes, which can clearly be dismissed by the human observer. Such features can be eliminated only if we take into consideration other properties of blood vessels patterns than merely their ridge shape.

One should however note that the first two algorithms (the skeleton construction and the skeleton filtering) start with a candidate skeleton consisting of all image pixels, and operate by successive eliminations of pixels which seemingly do not correspond to ridge locations. Thus it should come as no wonder that such a method, working by the application of a succession of restrictions on the set of possible skeleton pixels, without any possibility of recovering skeleton parts in function of the global context, can in certain cases eliminate some real blood vessel portions. Indeed, it happens that at some places, a part of a blood vessel does not satisfy with enough strenght the local properties of a ridge pattern, while it still can be recognized by the human observer thanks to the global context. This is for example the case with a portion of a vessel which becomes fuzzy or at the crossing of two or more vessels, which our skeleton construction and filtering algorithms tend to reject as "too noisy".

Thus some improvement to our work can be made in the direction of the restoration of missing blood vessel portions in function of the global context. For example, one can relax some of the restrictions imposed on candidate skeleton pixels in our skeletonization algorithm. In their present form, our algorithms are more suitable for situations where low noise is more important than complete recovery of small blood vessel portions, but one might modify them in order to meet other requirements (see [1] for a more detailed discussion on the use of distinct variants of the pseudomask according to distinct purposes).

Our algorithms have been implemented in PASCAL on a VAX 11/780 computer running under VMS and tested on $256 \times 256$ images having 256 grey levels. Due to memory limitation, we have implemented the three algorithms separately (leading to a duplication of some of the calculations, for example the averaging of grey levels on windows of increasing size). We chose as widths the values $2^k$ for $k = 1, 2, 3, 4$. It

takes then approximatively 10 minutes to process an image, the largest part of this time being taken by the skeletonization algorithm. (It is however possible to have a faster implementation with a more sophisticated handling of data; however this is not our goal: our programs should only illustrate our methods, and we believe that the underlying ideas would be hidden by the introduction of more complicated data structures).

The first two algorithms (skeleton construction and filtering) are absolutely parallel. This is not completely true for the third one, because a pixel $q$ can belong to several pseudomask windows $W(p)$, but only the lowest resulting grey level on $q$ is selected. In particular the result of the pseudomask is independent of the scan order on the image.

It is thus possible to devise a fast hardware implementation for our three algorithms, using parallelism on a wide scale. However this would require a very large memory, of size proportional to: (the number of pixels) $\times$ (the logarithm of the number of grey levels) $\times$ (the number of width levels).

**Organization**

The following three chapters of this report are devoted to each of the three algorithms (skeletonization, skeleton filtering, skeleton-controlled pseudomask). We have left out all technical details in order to keep our work concise. They will be dealt with in a technical note which will be referred to by [TN] in the text. The PASCAL programs will be reproduced and explained in it.

Last but not least, we show some sample data in the Appendix.

## II. Ridge detection by windowing

We recall our two assumptions made in the introduction, that the grey level increases from dark to light, and that blood vessels are lighter than the background. Thus they correspond to *ridges* in the image. (They would correspond to *valleys* if they were darker than the background, but this situation would be equivalent to the former one up to a grey level reversal).

We will thus consider the detection of ridges in the input image. This will be done by convolving that image with windows of various sizes and orientations, representing an approximation of the local shape of ridges having certain local widths and orientations. The resulting convolution images represent the likelihood functions of the match of the input image with ridge templates corresponding to these sizes and orientations. The selection of the skeleton pixels and their actual local width and orientation will be done by local comparisons between the values of these likelihood functions. More precisely, we will subject all possible candidate skeleton pixels and their possible local widths and orientations to four tests which will reject pixels outside the skeleton or wrong local orientations or sizes.

### II.1. Some local features of blood vessel patterns

Let us define the concept of a ridge with some precision. We recall that an *edge* is a line separating two relatively homogeneous regions having significantly different grey levels or textures. A *ridge* is a thin and elongated portion of an image enclosed by two regions having significantly lower grey levels. It is delineated by its edges with the two regions.

To an edge point we can associate two local orientations (modulo 360°): the *tangent orientation* (the one of the tangent on this point leaving the higher grey level on the left) and the *gradient orientation* (the one along which the grey level increases). We illustrate them on Figure 2. Clearly the gradient orientation forms a square angle counterclockwise with the tangent orientation.

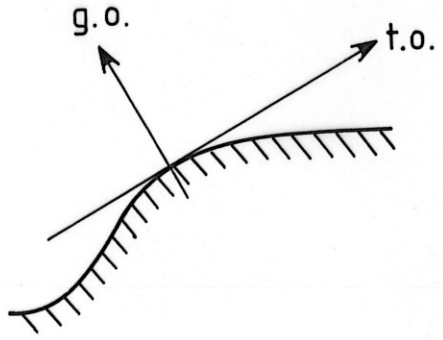



**Figure 2.** *The tangent orientation (t.o.) and gradient orientation (g.o.).*

For a ridge, the *skeleton* is the set of points equidistant from both edges. To each

skeleton point we can associate a local *width* and two local orientations modulo 180°: the *stream orientation* and *gradient orientation*. The stream orientation is equal to the average modulo 180° of the gradient orientations of the two edges, while the gradient orientation of the ridge is equal to the average modulo 180° of the tangent orientations of the two edges. We illustrate these two concepts in Figure 3.

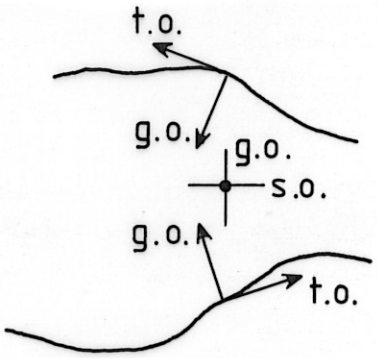



**Figure 3.** *The local stream orientation and gradient orientation.*

Of course, these features can be defined with precision only in a real-plane image having sharp edges. In the case of digital images, one can only have approximations. Moreover, the Euclidean distance is not suited for square grids, and we have to choose between two types of distances, the 4- or 8-distance, for the evaluation of the local width.

However these facts do not constitute a real problem, because we need only very rough approximations of the skeleton and of the associated local widths and orientations, since the precise details can be recovered by the skeleton-controlled pseudomask.

### II.2. Windows, their sizes and orientations

Assuming that the grid has $M$ rows, numbered $0, \ldots, M-1$ from top to bottom, and $N$ columns, numbered $0, \ldots, N-1$ from left to right, the pixel at intersection of row $i$ and column $j$ will be labelled $(i, j)$.

We will sample the orientations (modulo 360°) into the 8 fundamental directions corresponding to the 8 neighbours of a pixel in the square grid. We label them with geographical abbreviations: N, NW, etc. (see Figure 4).

A usual method in feature extraction is to match a simple template with the input picture, in other words to convolve that picture with a window and to search for "good" values in the convolved picture. For example, Sobel's edge-detection operator uses 8 antisymmetric windows corresponding to the 8 fundamental directions; we show in Figure 5 the two windows corresponding to the gradient orientations $N$ and $NW$ respectively.

As explained in [3] in the case of edge detection, a large feature needs a large window in order to be detected. This is especially true for ridges: a skeleton pixel will

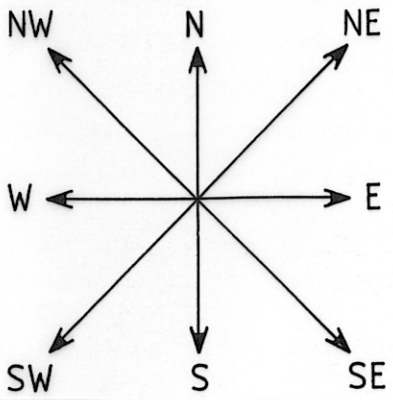



**Figure 4.** *The 8 fundamental directions.*

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 2 | 0 | -2 |
| 1 | 0 | -1 |

**Figure 5.** *Sobel's N and NW windows.*

not be detected unless the window centered on it is larger than the local width and so overlaps both edges.

Thus, given a basic window, one replaces in it every pixel by an averaging subwindow, and multiplies the weight of every pixel in an averaging subwindow by the weight of the corresponding pixel in the original window. We show in Figure 6 what Sobel's N-window becomes with a homogeneous subwindow.

More formally, suppose that we have the basic $x_0 \times x_1$-window $X$ associating to the pixel $(i,j)$ (where $0 \leq i < x_0$ and $0 \leq j < x_1$) the value $X(i,j)$, and an averaging $y_0 \times y_1$-subwindow $Y$ with values $Y(u,v)$ (where $0 \leq u < y_0$ and $0 \leq v < y_1$); then we build the compound window $X \otimes Y$ as a $x_0 \cdot y_0 \times x_1 \cdot y_1$-window defined by:

$$X \otimes Y(iy_0 + u, jy_1 + v) = X(i,j) \cdot Y(u,v) \qquad \text{for} \quad 0 \leq i < x_0,\ 0 \leq j < x_1,\ 0 \leq u < y_0 \text{ and } 0 \leq v < y_1. \tag{1}$$

We illustrate this operation in Figure 7. Note that in the averaging window $Y$ one often has $\sum_{u=0}^{y_0-1} \sum_{v=0}^{y_1-1} Y(u,v) = 1$. In this case one says that the averaging window $Y$ is *normalized*.

In [3] the averaging subwindows have size $2^k$. We will also take the size $2^k$, but

| | | | | | |
|---|---|---|---|---|---|
| $\frac{1}{4}$ | $\frac{1}{4}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{4}$ | $\frac{1}{4}$ |
| $\frac{1}{4}$ | $\frac{1}{4}$ | $\frac{1}{2}$ | $\frac{1}{2}$ | $\frac{1}{4}$ | $\frac{1}{4}$ |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| $-\frac{1}{4}$ | $-\frac{1}{4}$ | $-\frac{1}{2}$ | $-\frac{1}{2}$ | $-\frac{1}{4}$ | $-\frac{1}{4}$ |
| $-\frac{1}{4}$ | $-\frac{1}{4}$ | $-\frac{1}{2}$ | $-\frac{1}{2}$ | $-\frac{1}{4}$ | $-\frac{1}{4}$ |

**Figure 6.**

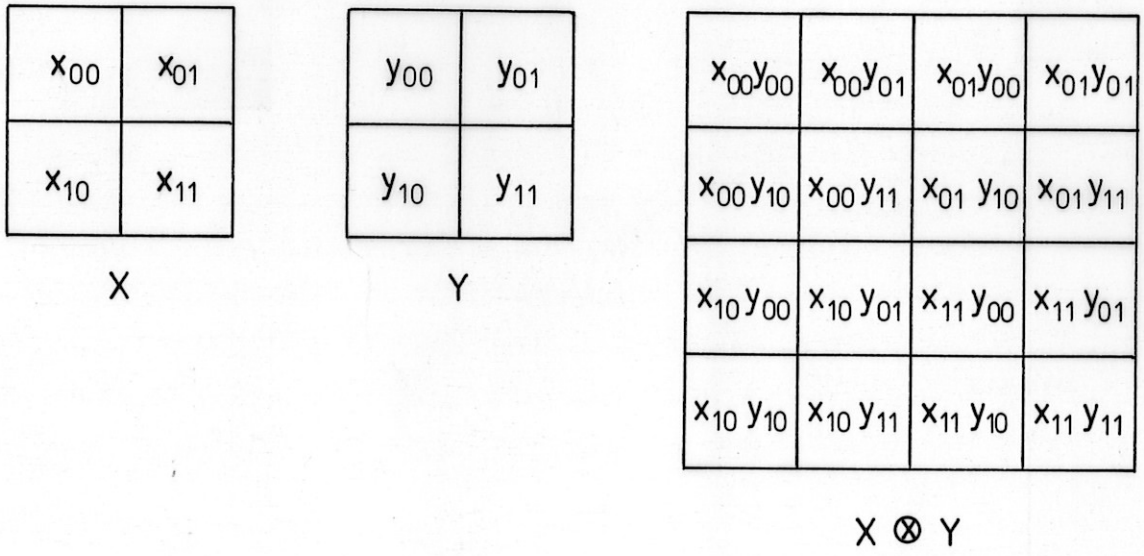



**Figure 7.**

we will restrict the range of $k$ between two integers $kmin$ and $kmax$, where $1 \leq kmin < kmax \leq 5$. In our programs, we have taken $kmin = 1$ and $kmax = 4$.

Given the input image $I$ and a window $Y$, write $I \wedge Y$ for the convolution of $I$ by $Y$. It is well-defined only up to a translation, because it depends upon the positioning of the window around each pixel. Generally one centers the window around that pixel; this is possible if $Y$ has odd size. Indeed, if $Y$ is $(2m+1) \times (2n+1)$, then we get:

$$\begin{aligned}(I \wedge Y)(i,j) &= \sum_{u=0}^{2m} \sum_{v=0}^{2n} I(i-m+u, j-n+v) \cdot Y(u,v). \\ &= \sum_{r=-m}^{m} \sum_{s=-n}^{n} I(i+r, j+s) \cdot Y(r+m, s+n).\end{aligned} \tag{2}$$

When the window has even size (and this happens with windows built from (1) with averaging subwindows of size $2^k$, $k \geq 1$), one can decenter the window by 1/2 pixel

in both directions in order to superpose the window pixels on the image pixels. This is done in [3], and if $Y$ is $2m \times 2n$ we obtain the following instead of (2):

$$\begin{aligned}(I \wedge Y)(i,j) &= \sum_{u=0}^{2m-1} \sum_{v=0}^{2n-1} I(i-m+u, j-n+v) \cdot Y(u,v). \\ &= \sum_{r=-m}^{m-1} \sum_{s=-n}^{n-1} I(i+r, j+s) \cdot Y(r+m, s+n).\end{aligned} \tag{3}$$

However we will not proceed in this way, but we will keep our windows absolutely centered. Thus in the even size case each pixel of the window will intersect several pixels of the image; but then the convolution will be computed in a similar way as in the case of functions defined on the real plane: the weight of a window pixel accounts as a coefficient of the grey level of an image pixel proportionally to the area of their intersection. We show the working of centered windowing on Figure 8.

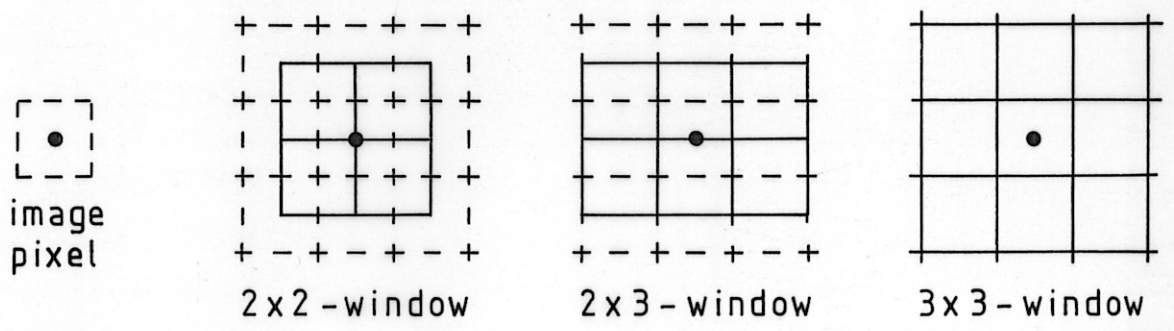



**Figure 8.** *Centered windowing.*

Although centered windowing seems computationally more complex than uncentered one, it won't be in practice because all our windows will be computed iteratively starting from small windows, and it is always easy to replace a small even-sized window by an odd-sized window giving the same result in centered windowing, as shown in Figure 9.

Now let us give the two types of averaging subwindows that we will use. Normally the sum of the weights of all pixels of such a window must be 1, and so these weights have fractional values. However computations are generally easier with integers, and so we will give unnormalized averaging subwindows together with a normalizing coefficient (equal to the sum of all pixel weights in the subwindow); then the result of the convolution with the unnormalized window must be divided by that coefficient.

We consider first the (unnormalized) homogeneous averaging $2^k \times 2^k$-subwindows $H_k$ (with $k \geq 0$), where all pixels have the same weight $pw$ (pixel weight). We take then the normalizing coefficient $hc_k$ (homogeneous window coefficient) which is defined by:

$$hc_k = 4^k \cdot pw. \tag{4}$$

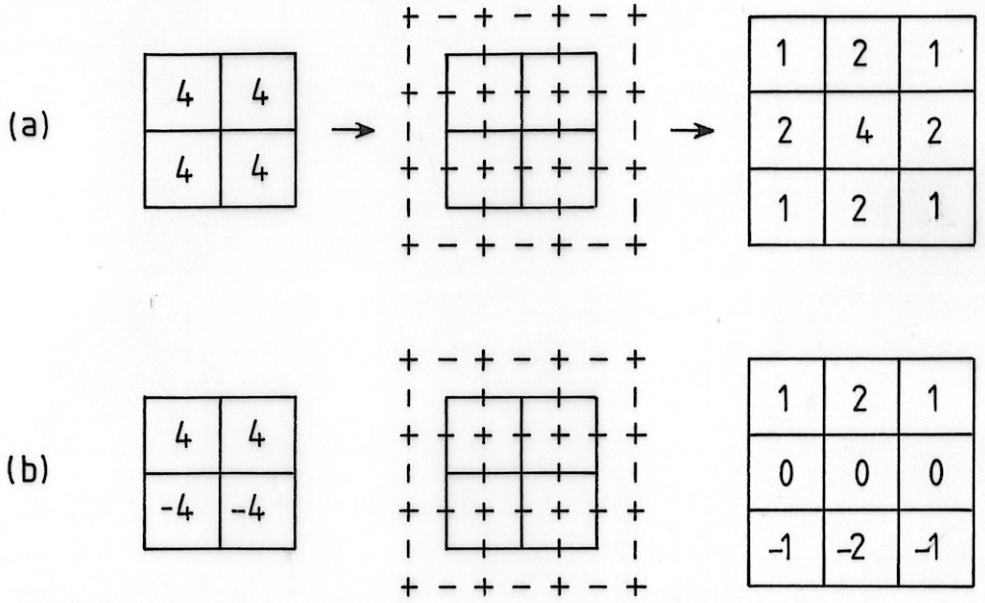



**Figure 9.** *Equivalence of windows for centered windowing.*

From Figure 9.a, where we show the transformation of $H_1$ in a suitable $3 \times 3$-window, it is clear that we must take:

$$hc_0 = pw = 4. \tag{5}$$

It is easily seen that $H_k$ is equivalent to a $(2^k + 1) \times (2^k + 1)$-window with respective weights 1 at corners, 2 along the edges and 4 in the interior. As noted in [3] (but for uncentered windowing), the convolution of the input image $I$ by $H_k$ can be easily computed by iteration on $k$, because for $k \geq 2$ we have:

$$(I \wedge H_k)(i,j) = \sum_{a=0}^{1} \sum_{b=0}^{1} (I \wedge H_{k-1})(i + (-1)^a \cdot 2^{k-2}, j + (-1)^b \cdot 2^{k-2}). \tag{6}$$

The coefficients $hc_k$ can also be computed iteratively, since we have for $k \geq 1$:

$$hc_k = 4 \cdot hc_{k-1}. \tag{7}$$

Although the homogeneous subwindows $H_k$ are the easiest ones to compute, it is preferrable to have unhomogeneous averaging subwindows in which pixels near the center have the highest weights and those near the border have comparatively very low weights. Indeed, when one builds large windows with averaging subwindows by (1), he should have a relatively smooth transition of the weights between the subwindows corresponding to the pixels of the basic window having different weights, or between the window and its surrounding. Further arguments against homogeneous subwindows can be found in [TN], for example in the discussion of "non-maxima suppression".

Now these subwindows should be easy to compute using some form of iteration. We have chosen as solution the pyramidal averaging $2^k \times 2^k$ subwindows $P_k$ $(k \geq 1)$, to which we associate the normalizing coefficient $pc_k$.

The window $P_k$ is built by adding centered copies of $H_0, \ldots, H_{k-1}$ to $H_k$. Thus we have:

$$(I \wedge P_k)(i,j) = \sum_{u=0}^{k} (I \wedge H_u)(i,j), \tag{8}$$

and so by (4)

$$pc_k = \sum_{u=0}^{k} hc_u = \left( \sum_{u=0}^{k} 4^u \right) \cdot pw = \frac{4^{k+1} - 1}{3} \cdot pw. \tag{9}$$

Clearly $I \wedge P_k$ and $pc_k$ can be computed iteratively, using $I \wedge P_{k-1}$, $I \wedge H_k$ and $pc_{k-1}$ (using (4, 7, 9)):

$$I \wedge P_k = I \wedge P_{k-1} + I \wedge H_k, \tag{10}$$

and

$$pc_k = pc_{k-1} + hc_k = 4 \cdot pc_{k-1} + pw. \tag{11}$$

Now that the averaging subwindows are at hand, let us give the basic windows for ridge detection. Their number is 8, since each one of them corresponds to one of the fundamental directions (see Figure 4). We show the N and NW windows in Figure 10; the other six ones can be found by rotating them by 90°, 180° and 270° respectively. The justification for our choice of windows can be found in [TN].

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 1 | 2 | 1 |
| 0 | 0 | 0 |

N window

| | | |
|---|---|---|
| 0 | -2 | 1 |
| -2 | 2 | 0 |
| 1 | 0 | 0 |

NW window

**Figure 10.**

These windows detect one side of a ridge of width 1. When two windows corresponding to two opposite orientations (modulo 360°) give "good" positive results in the convolution, then we have a ridge whose gradient orientation (modulo 180°) corresponds to that pair of orientations.

They correspond to the level 0 and they will be labelled $V_0^N$, $V_0^{NW}$, etc. (where $V$ stands for vessel). Then for $k \geq 1$ we can build the corresponding level $k$ windows $V_k^N$, $V_k^{NW}$, etc., using the pyramidal averaging subwindows: $V_k^N = V_0^N \otimes P_k$, $V_k^{NW} = V_0^{NW} \otimes P_k$, etc..

These windows must be normalized, because the convolution with the image should give the ridge height at each point. The 8 windows $V_0^N$, $V_0^{NW}$, etc. have the same normalizing coefficient (see [TN]), which is

$$vc_0 = 4, \tag{12}$$

and for $V_k^N$, $V_k^{NW}$, etc., we have the normalizing coefficient

$$vc_k = vc_0 \cdot pc_k. \tag{13}$$

Given an image pixel $(i,j)$, the likelihood of a horizontal ridge passing through it will be measured by $\big(I \wedge V_0^N\big)(i,j)$ and $\big(I \wedge V_0^S\big)(i,j)$ when they are both positive. If at least one of them is less than or equal to 0, then there is no such ridge through that pixel. This suggests the definition of the 4 likelihood functions:

$$\begin{aligned}
L_k^0(i,j) &= \frac{1}{vc_k} f\Big(\big(I \wedge V_k^N\big)(i,j), \big(I \wedge V_k^S\big)(i,j)\Big), \\
L_k^1(i,j) &= \frac{1}{vc_k} f\Big(\big(I \wedge V_k^{NW}\big)(i,j), \big(I \wedge V_k^{SE}\big)(i,j)\Big), \\
L_k^2(i,j) &= \frac{1}{vc_k} f\Big(\big(I \wedge V_k^W\big)(i,j), \big(I \wedge V_k^E\big)(i,j)\Big), \\
L_k^3(i,j) &= \frac{1}{vc_k} f\Big(\big(I \wedge V_k^{SW}\big)(i,j), \big(I \wedge V_k^{NE}\big)(i,j)\Big),
\end{aligned} \tag{14}$$

where $f$ is a "conditional averaging function" which satisfies the following requirements:

$(i)$ $f(a,b) = 0$ when $a \leq 0$ or $b \leq 0$.

$(ii)$ $f$ is continuous.

$(iii)$ $f(a,b) = f(b,a)$.

$(iv)$ $f(\lambda a, \lambda b) = \lambda f(a,b)$. for any $\lambda > 0$.

We choose $f(a,b) = min(a,b)$ for $a > 0 < b$ and $f(a,b) = 0$ otherwise. Another valid choice would be $f(a,b) = \sqrt{a \cdot b}$ for $a > 0 < b$ and $f(a,b) = 0$ otherwise, but it would be computationally more complex. For more details see [TN].

The superscripts 0, 1, 2 and 3 in (16) correspond to the 4 gradient orientations modulo 180°: N-S, NW-SE, W-E and SW-NE respectively, and the corresponding stream orientations are then consistent with the notation of Figure 1.

In fact, the likelihood functions will be somewhat more complicated, because we will need further checks involving auxiliary windows.

Indeed, consider a ridge of width level $k$ passing through some pixel $q$, and such that the flow orientation on $q$ is turning relatively sharply from NE-SW to NW-SE. Then a horizontal ridge for the level $k$ will be detected on some pixels situated to the W of $q$. We illustrate this situation in Figure 11.

Similarly, a large ridge can be detected on pixels neighboring the center of a circular ridge.

This defect arises from the fact that the windows $V_k^N$, $V_k^{NW}$, etc., do not check whether the ridge passes through the central subwindow. This will be done by 8 auxiliary windows $A_k^N$, $A_k^{NW}$, etc.. We show the windows $A_0^N$ and $A_0^{NW}$ in Figure 12 (the other

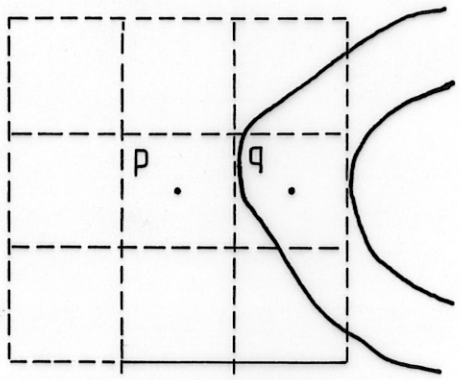



**Figure 11.** *A horizontal ridge is detected on p.*

| | | |
|---|---|---|
| 0 | -1 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

N window

| | | |
|---|---|---|
| 0 | $-\frac{1}{2}$ | 0 |
| $-\frac{1}{2}$ | 1 | 0 |
| 0 | 0 | 0 |

NW window

**Figure 12.**

6 ones for the level 0 are derived from them by rotation). From these 8 windows we derive for each level $k \geq 1$ the auxilliary windows $A_k^N$, $A_k^{NW}$, etc., by replacing pixels by pyramidal subwindows $P_k$ (see (3)).

Then the likelihood function $L_k^0(i,j)$ takes the following form:

$$L_k^0(i,j) = \frac{1}{vc_k} g\Big( (I \wedge V_k^N)(i,j), (I \wedge V_k^S)(i,j), (I \wedge A_k^N)(i,j), (I \wedge A_k^S)(i,j) \Big), \qquad (15)$$

where

$$g(a,b,c,d) = \begin{cases} min(a,b) & \text{if } a,\ b,\ c \text{ and } d \text{ are all } > 0; \\ 0 & \text{otherwise.} \end{cases} \qquad (16)$$

Similar forms of $L_k^1$, $L_k^2$ and $L_k^3$ can be defined instead of those in (14).

With this further check introduced in the likelihood functions $L_k^i$, the defect described above in Figure 11 is greatly reduced, although not completely suppressed. This will be done by the further checks described in the next section, in particular the "comparison with the average noise" (see Subsection II.3.1).

We can now define the best match function $B_k$ and the local stream orientation $O_k$ by:

$$\begin{aligned} B_k(i,j) &= max\{L_k^u(i,j) \mid u = 0,\ldots,3\}. \\ O_k(i,j) &= u \in \{0,1,2,3\} \quad \text{if } L_k^u(i,j) = B_k(i,j). \end{aligned} \qquad (17)$$

If $L_k^u(i,j) = B_k(i,j) = L_k^v(i,j)$ for $u \neq v$ (a very unlikely occurrence), then the ridge of level $k$ on $(i,j)$ must be eliminated. This is done by labelling the triple $(k,i,j)$ as "bad".

The reader will perhaps have noted that for a bounded input image $I$ (which is always the case in practice), it is impossible to compute the convolution of $I$ by a window on a point $(i,j)$ which is close to the border. The solution is to extend $I$ to a larger picture $\hat{I}$ by enclosing it in a frame of width $3 \cdot 2^{kmax-1}$. Then the subwindows $H_k$ and $P_k$ can be convolved with $\hat{I}$ on all pixels whose distance to the border of $\hat{I}$ is at least $2^{k-1}$, and the convolution with the windows of type $V_k$ and $A_k$ can thus be computed on all pixels of $I$. The grey level of the new pixels of $\hat{I}$ surrounding $I$ will be set to the *maximum* grey level value, so that no spurious ridge will be detected because of these pixels. (If we had been looking for valleys, we would have thus set the grey level to the *minimum* value).

Now that we have computed for each level $k = kmin, \ldots, kmax$ and each pixel $(i,j)$ of $I$ the best likelihood $B_k(i,j)$ and the local stream orientation $O_k(i,j)$, some selection must be made on the triples $(k,i,j)$ in order to find the skeleton pixels and their corresponding level. We have seen that $(k,i,j)$ is eliminated when $O_k(i,j)$ is not uniquely defined. In the next section we will give 4 more tests for the elimination of triples $(k,i,j)$.

### II.3. Four local tests for skeleton pixels

From the computations of the previous section we obtain for every level $k$ and nearly every pixel $(i,j)$ a best likelihood $B_k(i,j)$ and a local stream orientation $O_k(i,j)$. We need thus to make some selection among them. We will subject the triples $(k,i,j)$ to 4 local tests. Two of them (non-maxima suppression, level selection) come from [3], a third one (comparison with the average noise) is linked to the method of [4], while the last one (multiple ridge suppression) is to our knowledge original. For every level $k$, these 4 tests operate on all pixels in parallel and independently. However the last one depends on the result of the 4 tests for the previous levels $k' < k$.

All triples $(k,i,j)$ which fail any of these 4 tests will be labelled "bad", and the remaining ones will be labelled "good". In two of the tests (comparison with the average noise and multiple ridge suppression), we will make a further subdivision of "good" triples into "fair" and "best". The reason for this new subdivision is that in the skeleton filtering algorithm "fair" triples will be eliminated if there are not enough "best" triples in their neighborhood.

Let us now describe these 4 tests.

#### II.3.1. Comparison with the average noise

In [4] the edge likelihood in a given orientation on a given pixel is estimated by thresholding $\sigma^2/\sigma_L \cdot \sigma_R$, where $\sigma^2$ is the grey level variance in a window $W$ surrounding the pixel, while $\sigma_L^2$ and $\sigma_R^2$ are the variances in the two halves $W_L$ and $W_R$ of $W$ on each side of the supposed edge. As explained in [TN], with some further assumptions this is

equivalent to thresholding $h^2/\sigma^2$, where $h$ is the edge step, i.e. the difference in average grey levels between $W_L$ and $W_R$. We will apply a similar thresholding to ridges.

Here $h$ will be the best likelihood $B_k(i,j)$. Indeed, a ridge portion matching perfectly a pair of opposite windows, with constant grey levels $g$ inside and $g-h$ outside, will give $B_k(i,j)=h$

For the variance $\sigma^2$, we will give a weighted average between the variances of the 9 subwindows inside a $3\cdot 2^k \times 3\cdot 2^k$-window. We recall that for a distribution $X_1,\ldots,X_n$ with associated probabilities $p_1,\ldots,p_n$, one defines the mean $\mu$ and the variance $\sigma^2$ as:

$$\begin{aligned} \mu &= \sum_i p_i X_i; \\ \sigma^2 &= \sum_i p_i (X_i-\mu)^2 = \left(\sum_i p_i X_i^2\right) - \mu^2. \end{aligned} \tag{18}$$

In fact, when $\sigma^2$ is estimated from a small number of samples, one must correct it by some factor, getting thus the unbiased variance $\sigma_u^2$:

$$\sigma_u^2 = \frac{\nu}{\nu-1}\sigma^2, \quad \text{where } \nu = \frac{\left(\sum_i p_i\right)^2}{\sum_i p_i^2}. \tag{19}$$

Note that $\nu = n$ when the probabilities $p_i$ are equal.

Given a $2^k \times 2^k$-subwindow, the respective weights of its pixels correspond to the probabilities $p_i$; for the sake of simplicity, we will take the homogeneous averaging window $H_k$.

Let us compute the variance on the centered window $H_k$ around a pixel $(i,j)$. From the image $I$ (or rather $\hat{I}$), we construct the grey level squares image $J$, where for each pixel $(i,j)$ we have:

$$J(i,j) = I(i,j)^2. \tag{20}$$

As with $I$ (see (6)), the convolution $J \wedge H_k$ can be computed iteratively. Setting

$$\begin{aligned} (I \wedge H_k)(i,j) &= hi \\ \text{and} \quad (J \wedge H_k)(i,j) &= hj, \end{aligned} \tag{21}$$

the variance of the window $H_k$ is by (18):

$$\begin{aligned} \sigma^2 &= \frac{hj}{hc_k} - \left(\frac{hi}{hc_k}\right)^2; \\ &= hc_k^{-1} \cdot \left(hj - hi^2/hc_k\right). \end{aligned} \tag{22}$$

Now the factor $\nu$ of (19) is equal to the number $4^k = hc_k/hc_0$ of pixels of $H_k$, and so we have:

$$\frac{\nu}{\nu-1} = \frac{hc_k}{hc_k - hc_0}, \tag{23}$$

and so by (19) and (22):

$$\sigma_u^2 = \frac{\nu}{\nu - 1}\sigma^2 = \frac{hj - hi^2/hc_k}{hc_k - hc_0}. \tag{24}$$

In practice the computation of $hi^2$, whose value may attain

$$\left(4^{kmax+1} \times \textit{maximum grey level}\right)^2, \tag{25}$$

may easily cause an integer overflow. One can thus replace in (24) the term $hi^2/hc_k$ by $(hi/rthc_k)^2$, where $rthc_k = \sqrt{hc_k} = 2^{k+1}$ by (4, 5). In practice, one computes the coefficient $rthc_k$ iteratively (see (7)).

When we have computed $\sigma_u^2$ on each of the 9 subwindows around a pixel $(i, j)$, we make a weighted average of them, with weight proportional to those shown in the window $SA$ ("sigma averaging") of Figure 13.

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

**Figure 13.** *The variance averaging window SA.*

To $SA$ corrresponds the normalizing coefficient

$$sac = 16. \tag{26}$$

The averaged variance for level $k$ on $(i, j)$ will be written $\Sigma_k(i, j)^2$. It represents in some way the average noise around pixel $(i, j)$.

The test consists then in thresholding $B_k(i, j)/\Sigma_k(i, j)^2$. From our experiments it appears that the threshold should be set between 0.4 and 0.5. We make thus the following decision:

— If $B_k(i, j)/\Sigma_k(i, j)^2 < 2/5$, then $(k, i, j)$ is "bad".

— If $2/5 \leq B_k(i, j)/\Sigma_k(i, j)^2 < 1/2$, then $(k, i, j)$ is "fair".

— If $1/2 \leq B_k(i, j)/\Sigma_k(i, j)^2$, then $(k, i, j)$ is "best".

In the first case the triple $(k, i, j)$ must be eliminated as a skeleton pixel. The distinction between the second and third cases will intervene in the skeleton filtering algorithm of the next chapter.

With this test the defect described in Figure 11, which was reduced by the auxilliary windows $A_k^N, \ldots$, is completely eliminated.

### II.3.2. Non-maxima suppression

Given a ridge of width corresponding to the level $k$, the image representing the likelihoods $B_k(i,j)$ of all pixels $(i,j)$ will be a blurred representation of its skeleton.

Indeed, given a pixel $(i,j)$ in the skeleton, the function $B_k(u,v)$ will be positive for all pixels $(u,v)$ within a certain range (proportionnal to $2^k$) along the gradient orientation on $(i,j)$. As the pixel $(i,j)$ corresponds to a best match with the ridge-detecting windows, $B_k(i,j)$ should be a local maximum of the function $B_k$ along the gradient orientation on $(i,j)$ (which is perpendicular to $O_k(i,j)$).

This suggests Rosenfeld's method of "non-maxima suppression" [3] for deblurring the skeleton. For every pixel $(i,j)$ such that $B_k(i,j) > 0$, let $N_k(i,j)$ be the set of all pixels $(u,v)$ such that the 8-distance between $(i,j)$ and $(u,v)$ is not larger than $\mu \cdot 2^k$ (where $\mu$ is some fixed constant) and the line joining $(u,v)$ to $(i,j)$ (when $(u,v) \neq (i,j)$) is parallel to the gradient orientation of $(i,j)$ (which, according to Figure 1, corresponds to the number $O_k(i,j)+2$ modulo 4). If $B_k(i,j)$ is not the maximum of all $B_k(u,v)$ for $(u,v) \in N_k(i,j)$, then the triple $(i,j)$ is "bad" and must be eliminated.

As in [3], we will choose $\mu = 1/2$. We can give two arguments in favor of this choice:

$(i)$ Assuming that the local maximum of $B_k$ along the gradient orientation lies on the center of the ridge, a pixel at distance $2^{k-1} = \mu \cdot 2^k$ from it along the gradient orientation lies generally outside that ridge and so should be eliminated by the other checks.

$(ii)$ One could have two parallel ridges of width level $k$ whose skeletons lie at a distance of approximately $2^k$ one from another. Thus by taking $\mu = 1$, pixels of one skeleton could be eliminated by a comparison with those of the other one.

A further argument for our choice of $\mu$ can be found in [TN].

Two problems can arise in the selection of skeleton pixels by non-maxima suppression.

$(1°)$ One can obtain a skeleton which is not well-centered with respect to the corresponding ridge. This happens if the two ridge sides have distinct heights or steepnesses, etc.. This is not really harmful because, as we said earlier, we seek only a rough estimation of the skeleton, while precise details will be obtained by the pseudomask.

$(2°)$ The maximum of $B_k$ can be attained on several members of the set $N_k(i,j)$ of pixels to be compared to $(i,j)$. In particular the function $B_k$ could form a local plateau along the gradient orientation. In other words the skeleton could be unsufficiently deblurred and several pixels would be equal candidates as skeleton pixels. This is a real problem occurring for example with perfect symmetrical ridges having sharp edges. A possible remedy is to consider that in a sequence of maxima along the

gradient orientation, only the middle one (or the two middle ones if there is an even number of them) may be retained as candidate skeleton pixels. Thus for a pixel $(i,j)$ such that $B_k(i,j)$ is indeed the maximum of all $B_k(u,v)$ for $(u,v) \in N_k(i,j)$, the set $N_k(i,j) - \{i,j\}$ is divided into its two halves $N_k^0(i,j)$ and $N_k^1(i,j)$ corresponding to its two sides with respect to $(i,j)$. We count the two numbers $m_k^0(i,j)$ and $m_k^1(i,j)$ of pixels $(u,v)$ in $N_k^0(i,j)$ and $N_k^1(i,j)$ respectively, for which $B_k(u,v) = B_k(i,j)$. If $\left|m_k^0(i,j) - m_k^1(i,j)\right| > 1$, then $(k,i,j)$ is labelled "bad" and rejected.

With this refinement there remains at most two skeleton pixels corresponding to a ridge location. This is however a rare occurrence, and anyway it does not represent a major problem for further processing.

### II.3.3. Level selection

Given a ridge skeleton pixel $(i,j)$ with local width corresponding to the level $k$, a ridge will be detected on $(i,j)$ for other levels than $k$, especially those which are close to $k$ (one can expect that levels which are too distant from $k$ will be eliminated by the other tests, for example the comparison with the average noise).

Now the selection of levels for which there is a ridge through a pixel $(i,j)$ will be done by comparing the values $B_k(i,j)$ for $k = kmin, \ldots, kmax$. However our selection of levels should not be too restrictive: if we keep only one "good" level $k$ and eliminate all others, then the triple $(k,i,j)$ could still be eliminated by the 3 other tests, so that $(i,j)$ would not be a skeleton pixel.

In fact we can restrict ourselves to the comparison of $B_k(i,j)$ with $B_{k-1}(i,j)$ (if $k > kmin$) and $B_{k+1}(i,j)$ (if $k < kmax$). If any of these two is "better" than $B_k(i,j)$ (in a sense that we will define below), then $(k,i,j)$ is labelled "bad" and must be eliminated.

Let us now explain what is meant by "better". Suppose that we have a ridge of width $\alpha \cdot 2^k$ (where $1/2 \leq \alpha \leq 1$) centered on $(i,j)$, and we want to know whether it should correspond to level $k$ or level $k-1$. As $\alpha$ increases from $1/2$ to $1$, $B_k(i,j)$ increases also, while $B_{k-1}(i,j)$ decreases. Thus the quotient $B_k(i,j)/B_{k-1}(i,j)$ can be considered as a function of $\alpha$, although this function will depend on the shape of the ridge and its orientation. We can take a threshold $\theta$ such that for $\alpha \geq \theta$ the ridge has width level $k$, while for $\alpha < \theta$ it has width level $k-1$ (ideally one should take $\theta = \sqrt{2}/2$). To $\theta$ corresponds a certain value $\lambda$ of $B_k(i,j)/B_{k-1}(i,j)$; of course this value $\lambda$ depends on circumstances, but we can take an approximation of its average, and the resulting error will lead to a lowering or raising of the threshold $\theta$. Our criterion is then the following:

— If $B_k(i,j)/B_{k-1}(i,j) \geq \lambda$, then $(k,i,j)$ is "better" than $(k-1,i,j)$ and so $(k-1,i,j)$ is rejected as "bad".

— If $B_k(i,j)/B_{k-1}(i,j) < \lambda$, then $(k-1,i,j)$ is "better" than $(k,i,j)$ and so $(k,i,j)$

is rejected as "bad".

The fact that the value of the threshold $\theta$ corresponding to $\lambda$ changes in function of the orientation and shape of the ridge is not a real problem, because we seek only an approximation of the width. In particular, one should not be strict on the possible values of $\theta$. The main requirement is that when $\alpha$ increases from $1/2$ to 1, the value of $B_k(i,j)/B_{k-1}(i,j)$ passes from below $\lambda$ to above $\lambda$.

In [3] the value $\lambda = 3/4$ is chosen for edge detection. In [TN] we show that $\lambda = 1$ is an acceptable choice for ridge detection. Thus the triple $(k,i,j)$ is eliminated as "bad" if $B_{k-1}(i,j) > B_k(i,j)$ or if $B_{k+1}(i,j) \geq B_k(i,j)$.

**II.3.4.** The elimination of multiple ridges and related features

Given several neighboring ridges, they can be seen as a single larger ridge containing smaller valleys. We illustrate this situation in Figure 14.

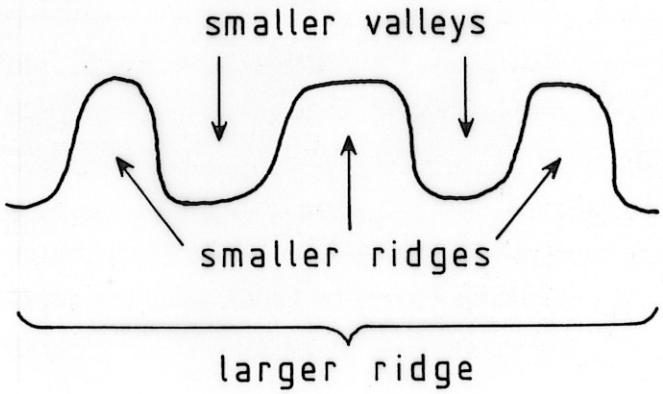



**Figure 14.**

Here our algorithm will detect both the smaller ridges and the larger one. However it is clear that the latter must often be eliminated as a false ridge, while in other circumstances we can have a superposition of several smaller ridges and a genuine larger one, as in Figure 15 for example. This problem, where several smaller features are assimilated to a larger feature of the same type, is not restricted to ridges. For example a succession of $n$ edges in a stair can also be detected as a $n$ times wider staircase-shaped edge.

While in Figure 14 the spurious larger ridge arises from smaller ones on *both* sides, the algorithm can also give a false ridge arising from one or several smaller ridges on *one* side of it only. Consider the situation shown in Figure 16, where we suppose that $h$ is very large in comparison to $v$.

A ridge $R$ of width level $k-1$ is detected on $p$. It induces a ridge $R^*$ of width level $k$ on $q$. It is easy to check that $R^*$ gives a local maximum for $B_k$ and has best level $k$; it is thus preserved by "non-maxima suppression" and "level selection"; it is also likely to be preserved by the "comparison with the average noise". Thus the ridge $R$ of level $k-1$ induces an erroneous detection of a ridge $R^*$ of level $k$ on $q$.

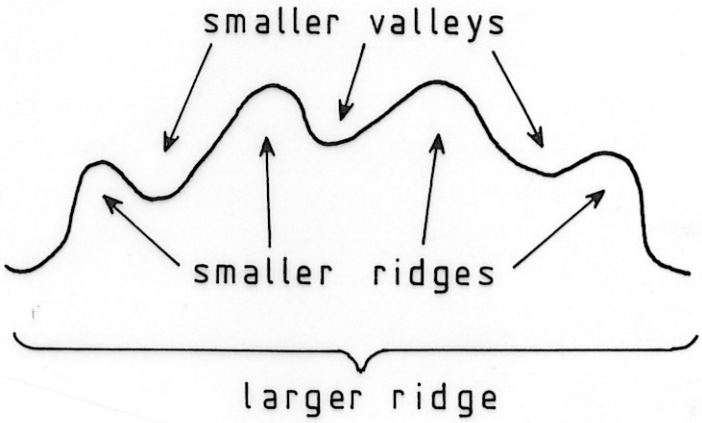



**Figure 15.**

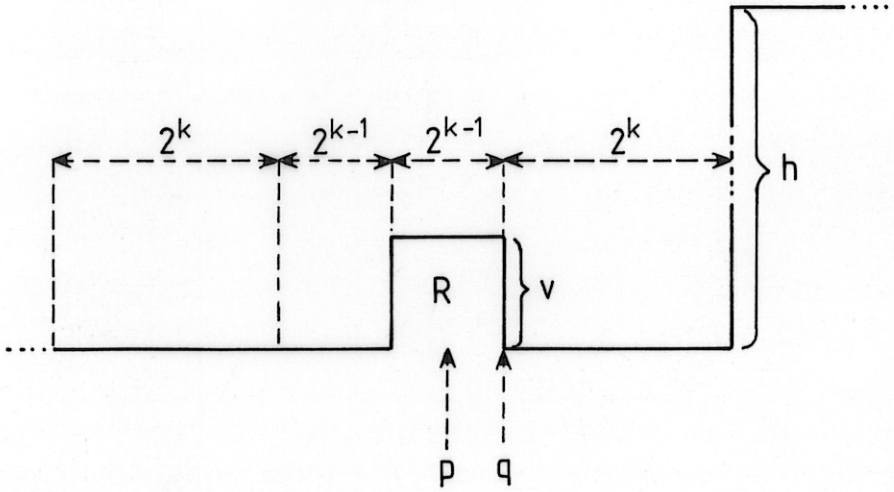



**Figure 16.**

The two examples given above (in Figures 14 and 16) have in common the fact that a false ridge of a given level is detected on a pixel $q$ because of ridges of small levels near $q$. The difference between the two is that in the first case these smaller ridges are located on both sides of $q$ along the gradient orientation, while in the second one they are located on one side of $q$ only.

Our test deals with both situations. It is applied to every level $k > kmin$ and assumes that the result of the 4 tests are known for all levels smaller than $k$. This means in particular that any implementation of our ridge detection algorithm must operate by iteration on the levels, from $kmin$ to $kmax$.

Let $k > kmin$ and consider a pixel $(i, j)$. As in the case of "non-maxima suppression" (see Subsection II.3.2 above), we consider the two sets $N_k^0(i, j)$ and $N_k^1(i, j)$ consisting of all pixels at 8-distance between 1 and $2^{k-1}$ of $(i, j)$ along the gradient orientation on either side of $(i, j)$ respectively. For every pixel $(u, v)$, let $sl_k(u, v)$ be the largest level $k' < k$ such that $(k', u, v)$ is a "good" triple; if $(k', u, v)$ is "bad" for every $k' < k$, then we set $sl_k(u, v) = 0$.

For a pixel $(u,v)$ in $N_k^0(i,j) \cup N_k^1(i,j)$, the size (both in width and depth) of a ridge of level smaller than $k$ on $(u,v)$ can be estimated as:

$$\begin{array}{ll} B_{k'}(u,v) \cdot 2^{k'} & \text{if } k' = sl_k(u,v) > 0; \\ 0 & \text{if } sl_k(u,v) = 0; \end{array} \tag{27}$$

Thus the total amount of ridges of smaller levels in $N_k^0(i,j)$ and $N_k^1(i,j)$ can be estimated by the respective sums $sum0$ and $sum1$ of all numbers of the form (27) corresponding to their pixels. These two numbers must be compared to the number

$$cent = 2^k \cdot B_k(i,j) \tag{28}$$

corresponding to the size of the ridge of level $k$ detected on $(i,j)$. We decide to reject the triple $(k,i,j)$ as "bad" in the following two cases, which correspond to the two situations described above (in Figures 14 and 16 respectively):

$$\begin{array}{lllll} (i) & \frac{1}{2}cent < sum_0 + sum_1 & \text{AND} & sum_0 \cdot sum_1 > 0. \\ (ii) & \frac{7}{8}cent < sum_0 + sum_1 & \text{AND} & sum_0 \cdot sum_1 = 0. \end{array} \tag{29}$$

When $(k,i,j)$ is "good", we have —as in Subsection II.3.1— a further subdivision into "fair" and "best": $(k,i,j)$ is "fair" in the following two cases:

$$\begin{array}{lllll} (iii) & \frac{1}{3}cent < sum_0 + sum_1 \leq \frac{1}{2}cent & \text{AND} & sum_0 \cdot sum_1 > 0. \\ (iv) & \frac{3}{4}cent < sum_0 + sum_1 \leq \frac{7}{8}cent & \text{AND} & sum_0 \cdot sum_1 = 0. \end{array} \tag{30}$$

On the other hand, $(k,i,j)$ is "best" in the following two cases:

$$\begin{array}{lllll} (v) & sum_0 + sum_1 \leq \frac{1}{3}cent & \text{AND} & sum_0 \cdot sum_1 > 0. \\ (vi) & sum_0 + sum_1 \leq \frac{3}{4}cent & \text{AND} & sum_0 \cdot sum_1 = 0. \end{array} \tag{31}$$

This subdivision into 6 cases has been deduced from our experiments. The comparison between the 3 numbers $cent$, $sum_0$ and $sum_1$ described above is sufficient to eliminate most pixels locating spurious ridges corresponding to the situations shown in Figures 14 and 16. The remaining ones will then be eliminated by the skeleton filtering algorithm of Chapter III.

### II.3.5. Summing up

To each triple $(k,i,j)$ one associates a ridge likelihood estimation $B_k(i,j)$ and a local orientation $O_k(i,j)$ (when $O_k(i,j)$ is not uniquely determined, then the triple $(k,i,j)$ is rejected as "bad"). One subjects then that triple to the following 4 tests.

(1°) Given the weighted average variance $\Sigma_k(i,j)^2$, one must have $B_k(i,j)^2/\Sigma(i,j)^2 \geq 2/5$.

(2°) Given the two sets $N_k^0(i,j)$ and $N_k^1(i,j)$ consisting of the two halves of the neighborhood of radius $2^{k-1}$ of $(i,j)$ along the gradient orientation $O_k(i,j)+2$ modulo 4, these two sets satisfy the following two conditions:

— For every pixel $(u,v)$ in $N_k^0(i,j) \cup N_k^1(i,j)$, $B_k(u,v) \leq B_k(i,j)$.

— The number of pixels $(u,v)$ such that $B_k(u,v) = B_k(i,j)$ differs by at most 1 in $N_k^0(i,j)$ and $N_k^1(i,j)$.

(3°) If $k > kmin$, then $B_k(i,j) \geq B_{k-1}(i,j)$; if $k < kmax$, then $B_k(i,j) > B_{k+1}(i,j)$.

(4°) If $k > kmin$, then the two statements of (29) are false.

If $(k,i,j)$ fails any of these 4 tests, then it is labelled "bad" and must be eliminated from the skeleton. The remaining triples are labelled "good", and are subdivided into the two subcategories "best" and "fair" according to whether or not they satisfy *both* the following two conditions, which are related to the tests (1°) and (4°) respectively:

(*i*) $B_k(i,j)^2/\Sigma(i,j)^2 \geq 1/2$.

(*ii*) The two statements of (30) are false.

These two tests do a lot of filtering among all triples. For a pixel $(i,j)$ it may happen that two distinct levels $k$ and $k'$ give "good" triples $(k,i,j)$ and $(k',i,j)$; thus $(i,j)$ can correspond to the location of two ridges of distinct widths. This is a legitimate occurrence. However we can be more restrictive and select for every pixel $(i,j)$ the highest level $k$ such that $(k,i,j)$ is "good"; if every level $k$ gives a "bad" triple $(k,i,j)$, then the selected level for $(i,j)$ will be 0.

The ridge skeleton can then be described by giving for every pixel $(i,j)$:

(*a*) The selected level $k$; for non-skeleton pixels we have $k = 0$, while for skeleton pixels we have $k > 0$ and $2^k$ approximates the local width.

(*b*) The local orientation $O_k(i,j)$ corresponding to the level $k$; if $k = 0$, it takes an arbitrary value.

# III. Skeleton filtering

With the ridge detection algorithm presented in the previous chapter, the skeleton contains pixels which satisfy the local properties of ridges but do not correspond to blood vessels; indeed, they are isolated pixels or they form short segments in the skeleton. As they reproduce noisy features of the original image in the pseudomask, they must be eliminated, and this may be done only by an analysis of the skeleton on a regional or a global scale, where connected portions of the skeleton which are too short in comparison to their width level are deleted.

Counting the length of each connected portion of the skeleton is a complex operation, and we have devised a computationally simpler algorithm which filters the skeleton by a combination of local and regional tests.

Our experiments show that it eliminates most noisy portions of the skeleton. However it preserves certain portions of the skeleton which correspond to elongated ridge-shaped parts of the image which nevertheless are not blood vessels. But they cannot be eliminated simply by filtering the skeleton; for this purpose we need to analyse their features such as the texture, the edge sharpness, etc..

We will first introduce a local connectivity feature of the skeleton, its set of *junctions*.

## III.1. Junctions

A connected branch of the skeleton satisfies adjacency properties which depend upon its orientation. When this orientation is close to the vertical or horizontal direction (and so when the local stream orientation of its pixels is along that direction), the skeleton is an 8-connected path. On the other hand, when this orientation is close to a diagonal direction (in which case the local stream orientation of its pixels is along that direction), the skeleton is a 4-connected path. We show this in Figure 17.

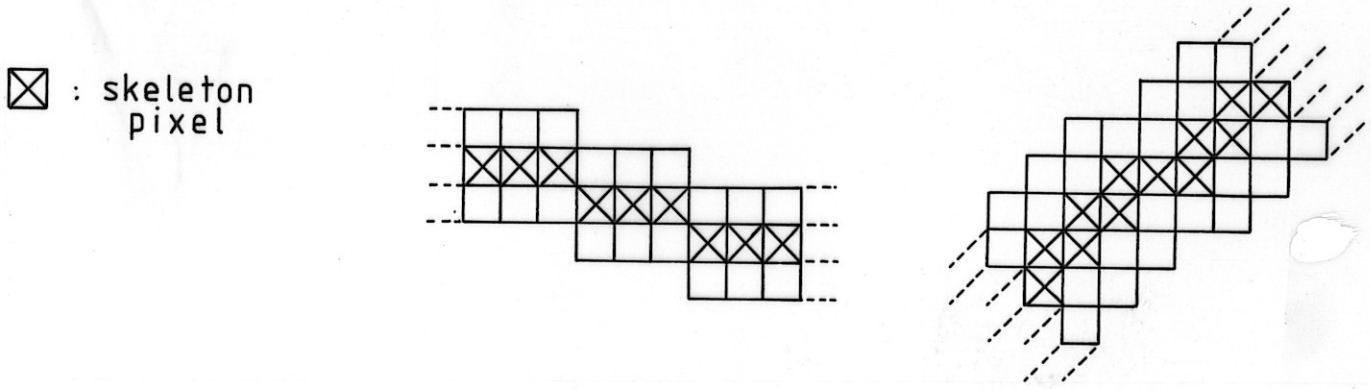



**Figure 17.**

Let us give an explanation for this fact. For a skeleton pixel $(i, j)$, let $k$ be the corresponding width level. We recall from Subsection II.3.2 the set $N_k(i, j)$ of all pixels

$(u, v)$ at 8-distance at most $2^{k-1}$ from $(i, j)$ along the gradient orientation $O_k(i, j) + 2$ modulo 4. We set $V(i, j) = N_k(i, j)$ if $(i, j)$ is a skeleton pixel of level $k$, and $V(i, j) = \{(i, j)\}$ if $(i, j)$ is not a skeleton pixel. As $V(i, j) - \{(i, j)\}$ is the set of pixels to be compared with $(i, j)$ in the "non-maxima suppression" test, its pixels should normally be eliminated from the skeleton, and so $V(i, j)$ represents an approximation of the portion of the ridge corresponding to $(i, j)$. Hence we get the connectivity patterns shown in Figure 17.

The connecting criterion for two skeleton pixels can then be described as follows. Let us say that a skeleton pixel is *axial* if its stream orientation is horizontal or vertical, and is *diagonal* otherwise.

Given two skeleton pixels $p = (i, j)$ and $p' = (i', j')$ having the same stream orientation, we will say that they are *well-connected* if:

$(i)$ they are 8-adjacent; and

$(ii)$ $V(i, j)$ and $V(i', j')$ are adjacent but do not intersect.

Thus this means that $p$ and $p'$ are 4-adjacent if they are diagonal, and that they are 8-adjacent not along their gradient orientation if they are axial. We show this in Figure 18.

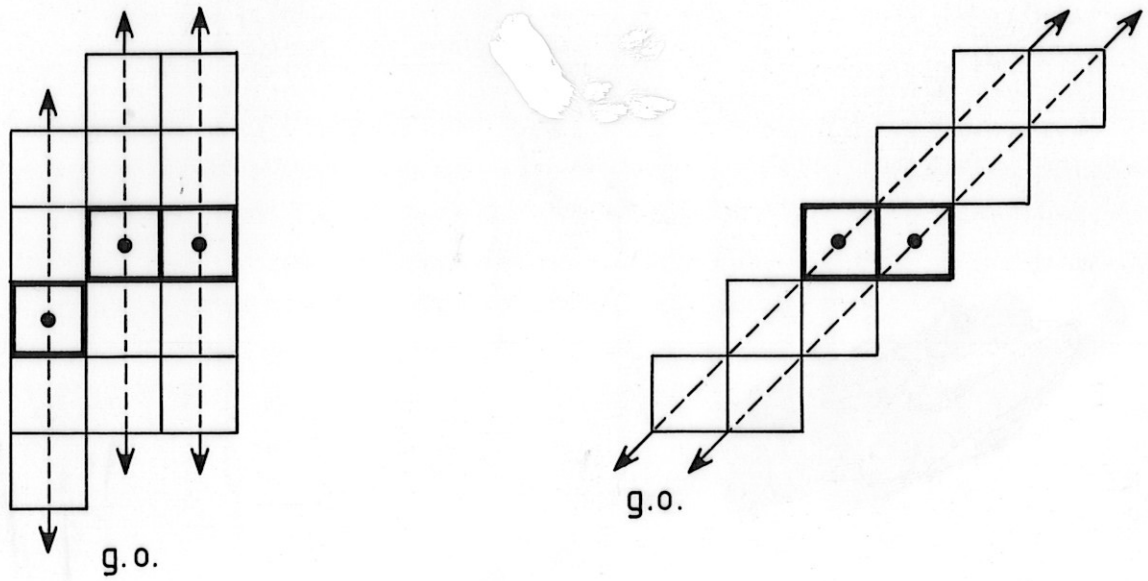



**Figure 18.**

We consider also well-connectedness between two skeleton pixels $p = (i, j)$ and $p' = (i', j')$ whose respective stream orientations form an angle of 45° (in other words between an axial and a diagonal pixel). They will be well-connected if:

$(i)$ they are 8-adjacent; and

$(ii)$ $V(i, j) \cap V(i', j') = \{q\}$, where $p \neq q \neq p'$, but $q$ is 8-adjacent to both $p$ and $p'$.

This means that $p$ and $p'$ are 4-adjacent not along the gradient orientation of the

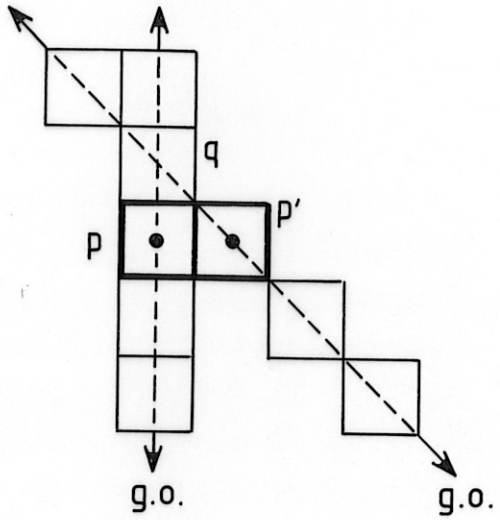



**Figure 19.**

one of them which is axial. We illustrate this in Figure 19.

We recapitulate in Figure 20 the possible configurations of well-connected skeleton pixels, up to a symmetry of the square.

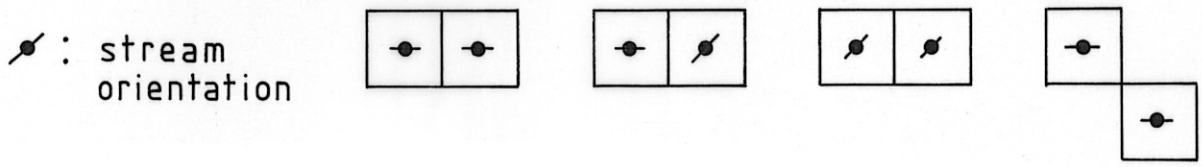



**Figure 20.**

Given a skeleton pixel $(i, j)$, $V(i, j)$ separates its neighborhood into two sides, and another skeleton pixel well-connected to it belongs to either side. Now in a skeleton branch, the pixels are well-connected to other ones on both sides, and the width level should not change too abruptly. This induces the following definition:

> A skeleton pixel $p$ is a *junction* if and only if it is labelled "best" for the two tests of Subsection II.3.1 and II.3.4, and there exist two other pixels $q$ and $r$ well-connected to it and lying on either side of it, such that the width levels of $p$, $q$ and $r$ differ by at most 1.

We will require that every neighborhood of a skeleton pixel, up to a certain width, contains a sufficient number of junctions having some specified levels. Indeed, a skeleton branch must be long enough and most of its pixels should be junctions (small breaks in the skeleton may sometimes arise because of irregularities in the ridge shape).

### III.2. Skeleton filtering by junction counting

Consider a portion of the skeleton containing a pixel $p$ of width level $k$. It can have

connectivity breaks, but the distance between two successive connected parts should be small in comparison to the width of their pixels, so that these breaks can be filled during the ridge reconstruction in the pseudomask. Secondly, these connected parts should be long enough in comparison to this width. Finally the level of the pixels in this portion should not vary too much: two skeleton pixels at small distance whose levels differ by at least 2 must be considered as members of two distinct portions of the skeleton.

Thus, given this skeleton pixel $p$ of level $k$, we expect that there is around it a set of skeleton pixels which are junctions and whose width levels are either all equal to $k$ or $k-1$, or all equal to $k$ or $k+1$. For every integer $u$ such that $kmin \leq u < kmax$, write $J[u+1/2]$ for the set of junctions having width level $u$ or $u+1$. Readily, we will search for a good configuration of pixels around $p$ in $J[k+1/2]$ or in $J[k-1/2]$. This can be expressed as a number of necessary conditions on the number of elements of $J[k-1/2]$ (or $J[k+1/2]$) in certain windows. We can temporarily restrict ourselves to $J[k-1/2]$.

In fact, these conditions correspond to the width levels from 0 to $k$. The level 0 condition is that $p$ belongs to $J[k-1/2]$ (*i.e.*, $p$ is a junction) or one of its 8-neighbors not along the gradient orientation is in $J[k-1/2]$.

For $1 \leq u \leq k$, the level $u$ condition will be expressed in terms of the number of members of $J[k-1/2]$ inside $2^u \times 2^u$-windows centered on certain pixels around $p$. We can represent $J[k-1/2]$ as a binary image assigning the value 1 to its elements and 0 to other pixels. Then this number of elements of $J[k-1/2]$ inside these windows is given by:

$$\frac{1}{pw}(J[k-1/2] \wedge H_u), \tag{32}$$

where $pw = 4$ is the weight of a pixels in $H_u$ (see (5)).

The level $u$ condition states that one can find at least $3 \cdot 2^{u-1} - 1$ junctions around $p$ inside two $2^u \times 2^u$-windows, one centered on $p$, and the other centered on some pixel $q$ at 8-distance $2^u$ from $p$, not along a direction close to the gradient orientation of $p$. We show this graphically in Figure 21.

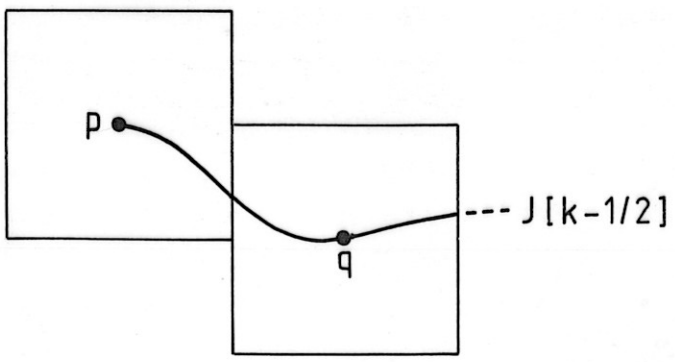



**Figure 21.**

The number $3 \cdot 2^{u-1} - 1$ corresponds to the minimum length of a branch in $J[k-1/2]$ within these two windows. The term $-1$ has been added in order not to be too restrictive for $u = 1$, in which case a pixel of $J[k - 1/2]$ neighboring $p$ may overlap the window borders.

Let us now describe how we can choose $q$. If one restricted oneself to the the 8 fundamental directions (more precisely, to the 6 ones which are not parallel to the gradient orientation), then one could have portions of the skeleton passing between two possible windows, say the ones along the NW and W directions, and any choice of $q$ along one of these two directions would miss a part of that portion. Thus the set of possible choices for $q$ should give a set of overlapping windows, and we need to consider a larger set of orientations than the 8 fundamental ones. The simplest extension is to consider the 16 directions shown in Figure 22, which are determined by the 16 vectors $(c, d)$, where

$$\begin{aligned} -2 \leq c \leq 2, \\ -2 \leq d \leq 2, \\ \text{and} \quad max(|c|, |d|) = 2. \end{aligned} \tag{33}$$

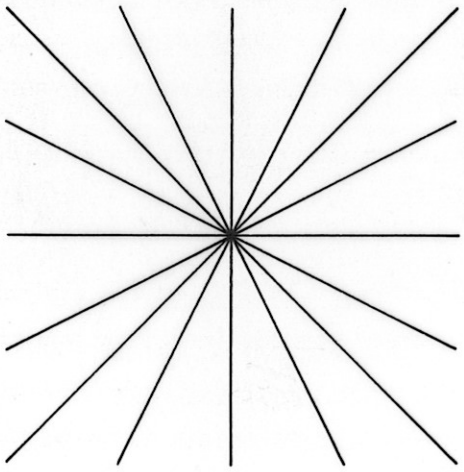


**Figure 22.**

Now $q$ will be at 8-distance $2^u$ from $p$ along one of these directions, and so, given $p = (i, j)$, it will take the form

$$(i + c \cdot 2^{u-1}, j + d \cdot 2^{u-1}), \tag{34}$$

where $c$ and $d$ satisfy (33). We will require that the direction of $q$ with respect to $p$ is not parallel to the gradient orientation of $p$ or any of its two neighboring orientations in Figure 22. We show this in Figure 23.

If we represent the gradient orientation of $p$ by a vector $(a, b) = (1, 0), (1, 1), (0, 1)$

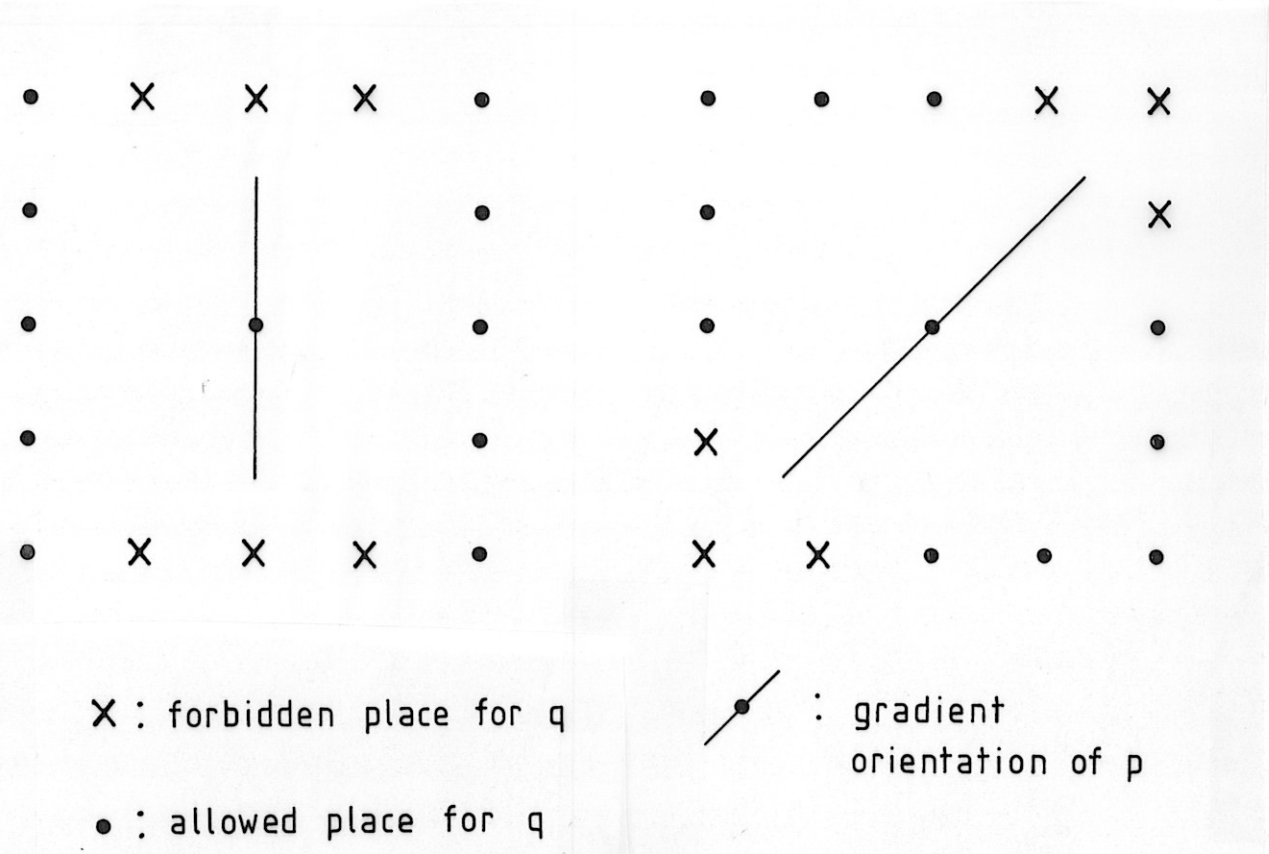



**Figure 23.**

or $(-1,1)$, this restriction on $q$ can be expressed as:

$$\begin{aligned} & max(|c-2a|\,,|d-2b|) > 1 \\ \text{and} \quad & max(|c+2a|\,,|d+2b|) > 1. \end{aligned} \tag{35}$$

Let us sum up the level $u$ condition. Write $Q(i,j)$ for the set of possible choices for $q=(i',j')$ given the gradient orientation of $p=(i,j)$. Then we will require that:

$$\frac{1}{pw}\left( \left(J[k-1/2] \wedge H_u\right)(i,j) + max_{(i',j')\in Q(i,j)} \left\{ \left(J[k-1/2] \wedge H_u\right)(i',j') \right\} \right) \geq 3 \cdot 2^{u-1} - 1. \tag{36}$$

We recall the level 0 condition, that $p$ or one of its 8-neighbors not along the gradient orientation belongs to $J[k-1/2]$.

The level $u=0,\ldots,k$ conditions using $J[k-1/2]$ are rough approximations for the requirement that $p$ belongs to a "sufficiently connected" skeleton branch of level varying between $k$ and $k-1$. For a branch of level varying between $k$ and $k+1$, we can do with $J[k+1/2]$ the same as we did above with $J[k-1/2]$. We have thus two tests on a skeleton pixel $p$ of level $k$:

— The *inferior test*: $p$ satisfies the level $u$ condition on $J[k-1/2]$ for every $u=0,\ldots,k$.

— The *superior test*: $p$ satisfies the level $u$ condition on $J[k+1/2]$ for every $u=0,\ldots,k$.

Now our skeleton filtering criterion is the following: a skeleton pixel is eliminated if and only if it fails *both* the inferior and the superior tests. Note that if $p$ has width level $kmin$, only the superior test is available, while if $p$ has width level $kmax$, only the inferior test is available.

Our filter is based on *necessary* requirements, *not* on *sufficient* ones. Thus it sometimes fails to eliminate spurious disconnected portions of the skeleton. On the other hand, it can sometimes eliminate large clusters of skeleton pixels labelled "average", which correspond to fuzzy portions of a genuine blood vessel. However our experiments show that it gives satisfactory results on most skeleton parts. In fact, filtering errors generally occur on blood vessel portions where the skeletonization led to a poor result.

## IV. Skeleton-controlled pseudomask

Now that the filtered skeleton is at hand, we can use the local width and orientation of each of its pixels in order to determine the size and orientation of the corresponding windows for the pseudomask; within these windows we will compute an approximation of the grey level of the background around the corresponding ridge locations. This will lead to the pseudomask image, whose difference with the original image will display the ridges.

Let us describe first the pseudomask windows that we will use. A window will be associated to every skeleton pixel, but also to some other pixels at places where there are discontinuities, turns or changes of width level in the skeleton.

We said in the previous chapter that the set $V(i,j)$ of pixels at 8-distance at most $2^{k-1}$ from $(i,j)$ along the gradient orientation (where $k$ is the width level of $(i,j)$) can be seen as the ridge portion corresponding to a skeleton pixel $(i,j)$. Thus the pseudomask window corresponding to a pixel $(i,j)$ should contain $V(i,j)$, but it should extend farther in order to reach the background.

We will thus associate to the skeleton pixel $(i,j)$ the pseudomask window $W(i,j)$ consisting of all pixels at 8-distance at most $2^k$ from $(i,j)$ along the gradient orientation; indeed, we may expect that the background to be at a distance larger than $2^{k-1}$ from $(i,j)$, and our experiments show that extending this distance to $3 \cdot 2^{k-1}$ leads to improper results.

Ideally, each $W(i,j)$ should contain the ridge portion corresponding to $(i,j)$, and so the windows $W(i,j)$ should cover the whole of each ridge, while the transition between them should be smooth. This means that we would have a succession of pixels for which the corresponding windows would be 4-adjacent and parallel, with their respective extremities touching each other. In other words, we would require that neighboring skeleton pixels should be well-connected (in the sense of Chapter III) and have the same local orientation and width level.

When this does not happen, one must "fill the gap" in this covering of the ridges at pixels where one has breaks, turns, or changes of width in the skeleton. For a skeleton pixel $(i,j)$, $W(i,j)$ separates its neighborhood into two sides. If such a skeleton change occurs inside one given side of the neighborhood of some pixel $(i,j)$, then we complete the set of windows in the following way:

For $t = 1, \ldots, 2^k - 1$ (where $k$ is the level of $(i,j)$), let $(i_t, j_t)$ be the pixel at distance $t$ from $(i,j)$ along the stream orientation on that side. To each $(i_t, j_t)$ we associate a window $W(i_t, j_t)$ of width $2(2^k - t)$ (instead of $2^{k+1}$ for $(i,j)$). If $(i,j)$ is diagonal, these windows are not 4-adjacent, and so we take a pixel $(i'_1, j'_1)$ 4-adjacent to both $(i,j)$ and $(i_1, j_1)$; then for $t = 2, \ldots, 2^k - 1$, let $(i'_t, j'_t)$ be the pixel at distance $t-1$ from $(i'_1, j'_1)$, and we associate to $(i'_t, j'_t)$ a window $W(i'_t, j'_t)$ of width $2(2^k - t)$. We illustrate this for

$k = 2$ on Figure 24. The pixels $(i_t, j_t)$ (and $(i'_t, j'_t)$ in the diagonal case) will be considered as having level $k$.

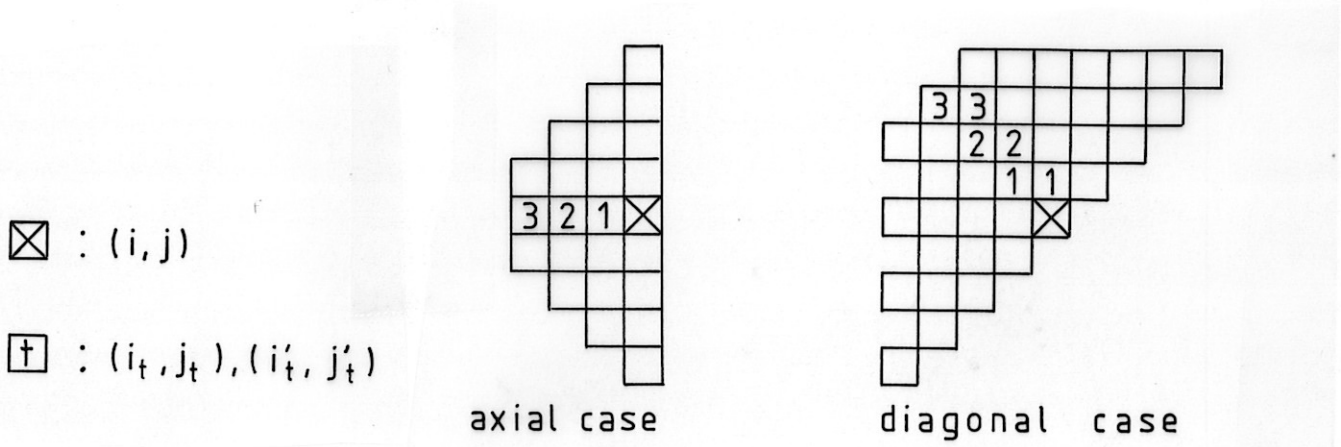



**Figure 24.**

Now let us state more precisely the circumstances where we have to "fill the gap" on a certain side of some pixel. Given a pixel $(i, j)$ of width level $k$, we look on each side of its neighborhood for a skeleton pixel $(i', j')$ such that:

(*a*) $(i', j')$ is 8-adjacent to $(i, j)$ not along its gradient orientation if $(i, j)$ is axial, and 4-adjacent to $(i, j)$ if $(i, j)$ is diagonal (see the definition of well-connected skeleton pixels in the preceding chapter);

(*b*) $(i', j')$ has the same local stream orientation as $(i, j)$.

(*c*) The level $k'$ of $(i', j')$ is not smaller than $k$.

If there is no such $(i', j')$ on a given side of $(i, j)$, then we "fill the gap" on this side of $(i, j)$ in the way described above. Note that in (*c*) we require only that $k' \geq k$, and not $k' = k$. Indeed, if $k' > k$, we will have a gap to fill on the side of $(i', j')$ containing $(i, j)$, but not on the side of $(i, j)$.

Now that we have a set of windows covering the ridges of the image (at least those described by the skeleton), let us describe what we do in each window for the construction of the pseudomask image.

At the start, the pseudomask image $I'$ is set equal to the original image $I$. Then it is modified successively on each window $W(i, j)$ described above. Given a skeleton or "gap-filling" pixel $(i, j)$ of width level $k$, we will modify the value of $I'$ in $W(i, j)$ in function of the value of the level $k$ average grey level $pc_k^{-1}(I \wedge P_k)$ inside it. We show in Figure 25 two diagrams: the first one gives the grey levels $I(i', j')$ of pixels in a window $W(i, j)$, while the second one gives the level $k$ averages $pc_k^{-1}(I \wedge P_k)(i', j')$ on $W(i, j)$.

On each side of $(i, j)$ in $W(i, j)$, we search for the minimum of the level $k$ average $pc_k^{-1}(I \wedge P_k)$. We get thus two numbers $b_1$ and $b_2$. Let then $b$ be the maximum of $b_1$ and $b_2$. On each side of $(i, j)$ in $W(i, j)$, we look for the farthest pixel $(i', j')$ such that

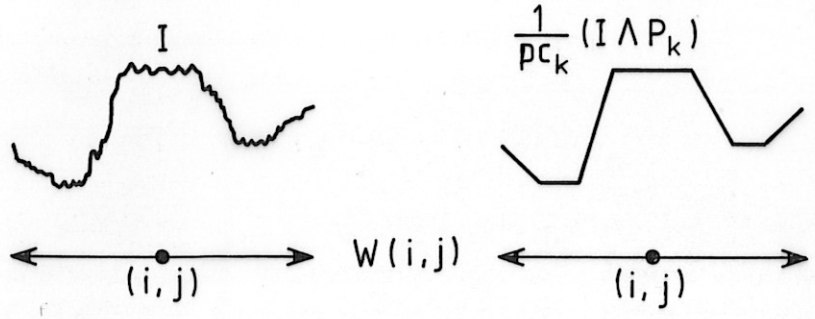



**Figure 25.**

$pc_k^{-1}(I \wedge P_k)(i', j') \geq b$; we get thus two pixels $(i_1, j_1)$ and $(i_2, j_2)$. Then $b$ will be the value for $I'$ inside the interval enclosed between $(i_1, j_1)$ and $(i_2, j_2)$. In other words, for every pixel $(i', j')$ lying between $(i_1, j_1)$ and $(i_2, j_2)$ included, we replace $I'(i', j')$ by $b$ provided that $b < I'(i', j')$. We illustrate this process in Figure 26 with the image portion of Figure 25.

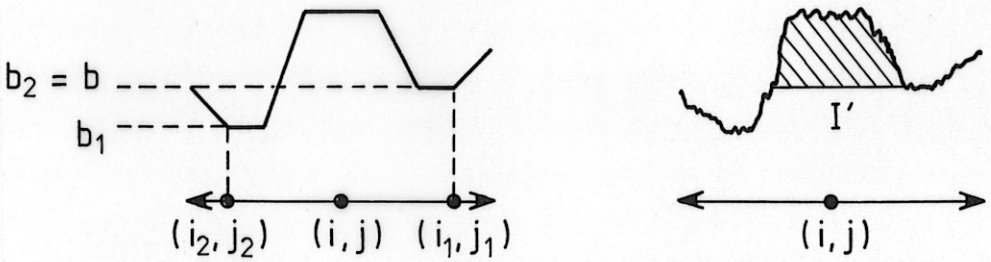



**Figure 26.**

When we have done this for all windows $W(i, j)$ covering the ridge, we get the final form of the pseudomask image $I'$. Now $I - I'$ will show the ridge patterns around the skeleton.

If we were looking for valleys instead of ridges, then we should intervert "maximum" and "minimum", "$\geq$" and "$\leq$", ">" and "<" in the above discussion, and take $I' - I$ instead of $I - I'$.

Our skeleton-controlled pseudomask algorithm is not parallel in the usual sense because the value of $I'$ on a given pixel may depend on several local processes, since that pixel may belong to several windows. In fact, the value of $I'$ on a pixel is the minimum of all values obtained on all windows containing it (for valleys, it would be the maximum). It follows thus that the algorithm is independent of the order of the processing of windows and so of the scan order of the image.

From our experiments it appears that the pseudomask based on the filtered skeleton reproduces with a low noise the ridge-shaped patterns of the original image, or at least those which were detected by the skeletonization process. Moreover the "gap-filling" method allows a partial recovery of ridges at skeleton breaks.

The contrast in the difference image $I - I'$ can be enhanced by maximizing its dynamic range. This can be done linearly for example. But if we want a higher enhancement of lower grey level, then this can be done by a linear function of the square root of the grey levels.

# V. Conclusion

The skeleton found by the method of Chapter II gives generally a good approximation of the location of ridges corresponding to blood vessel-shaped patterns and a rough estimation of the local width and orientation, which are nonetheless sufficient for the pseudomask image construction.

The three main problems encountered with our skeleton are the following ones:

($i$) It contains small noisy portions which are too short to correspond to a vessel.

($ii$) There are gaps in the skeleton, and sometimes portions of it corresponding to narrow or fuzzy parts of blood vessels, or to crossings of several vessels, are missing.

($iii$) The concept of ridge is unsufficient to characterize the shape of blood vessel images: with our algorithm we get skeleton portions corresponding to elongated ridge-shaped patterns which can nevertheless be dismissed by the human observer as too fuzzy or too large to be blood vessels.

A satisfactory solution to ($i$) is given by our filtering algorithm of Chapter III.

Now ($ii$) is a more serious problem, because it can lead to diagnostic errors in the case of blood vessel enhancement: a small gap in the skeleton leads to a sthenosis in the pseudomask subtraction image (see also the discussion on 3-D reconstruction in [1]). Some correction to this defect has been made by the gap-filling procedure in the pseudomask, but it is unsufficient in some cases. Thus a possible domain of research is the improvement of our ridge-detection and skeletonization algorithm through the recovery of missing small portions in the skeleton. It should be noted that missing portions can be sometimes recovered if we make the local tests of Section II.3 less severe, but this will lead to the introduction of noisy pixels in the skeleton. A trade-off between low noise and completeness of the skeleton should be sought for in each circumstance.

At the present stage, our skeletonization algorithm is more suited to situations where low noise is more important than a complete recovery of the blood vessel system.

Problem ($iii$) is rather minor at the present stage, because it does not lead to diagnostic errors.

The skeleton-controlled pseudomask algorithm gives a neat reproduction of ridge-shaped patterns of the image, thanks to the smoothing of discontinuities by the gap-filling procedure, but gaps or missing portions in the skeleton lead to missing portions or narrowings in the blood vessels shown in the pseudomask difference image.

## References


[1] P.A. Devijver, C. Ronse, P. Haaker, E. Klotz, R. Koppe, R. Linde: Pseudo-mask technique for digital subtraction angiography (DSA). *Mustererkennung 1984, DAGM 6th Symposium, Proceedings* (1984).

[2] H. Möller: *Rekonstruktion aus Projektionen.* PFH Laborbericht NR. 578/84 (1984).

[3] A. Rosenfeld, M. Thurston: Edge and curve detection for visual scene analysis. *IEEE Trans. Computers*, Vol. C-20, n° 5, 562–569 (1971).

[4] Y. Yakimovsky: Boundary and object detection in real world images. *J. ACM*, Vol. 23, n° 4, 599-618 (1976).

[TN] C. RONSE: *Blood vessel detection, technical details.* PRLB Technical Note in preparation.

## Appendix. Some sample data

The 3 algorithms have been implemented in PASCAL on a VAX 11/780 computer and tested on six digitized X-ray images; each one has size $256 \times 256$ and 256 grey levels.

Figures 28 to 33 show the result of the processing on the six input images. Each figure contains four images, namely:

(*a*) A photograph of the original digitized X-ray image.

(*b*) A representation of the skeleton (by a raster scan plotter), together with the level of skeleton pixels: to level 1, 2, 3, 4 pixels correspond dots, hollow octogons, full octogons and squares respectively, as shown in Figure 27.

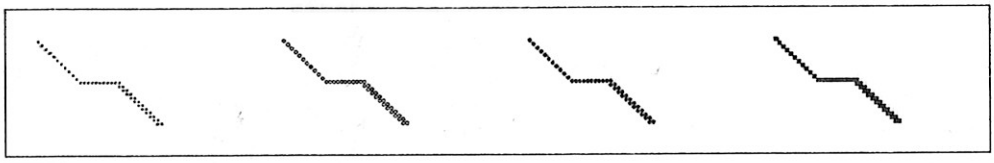
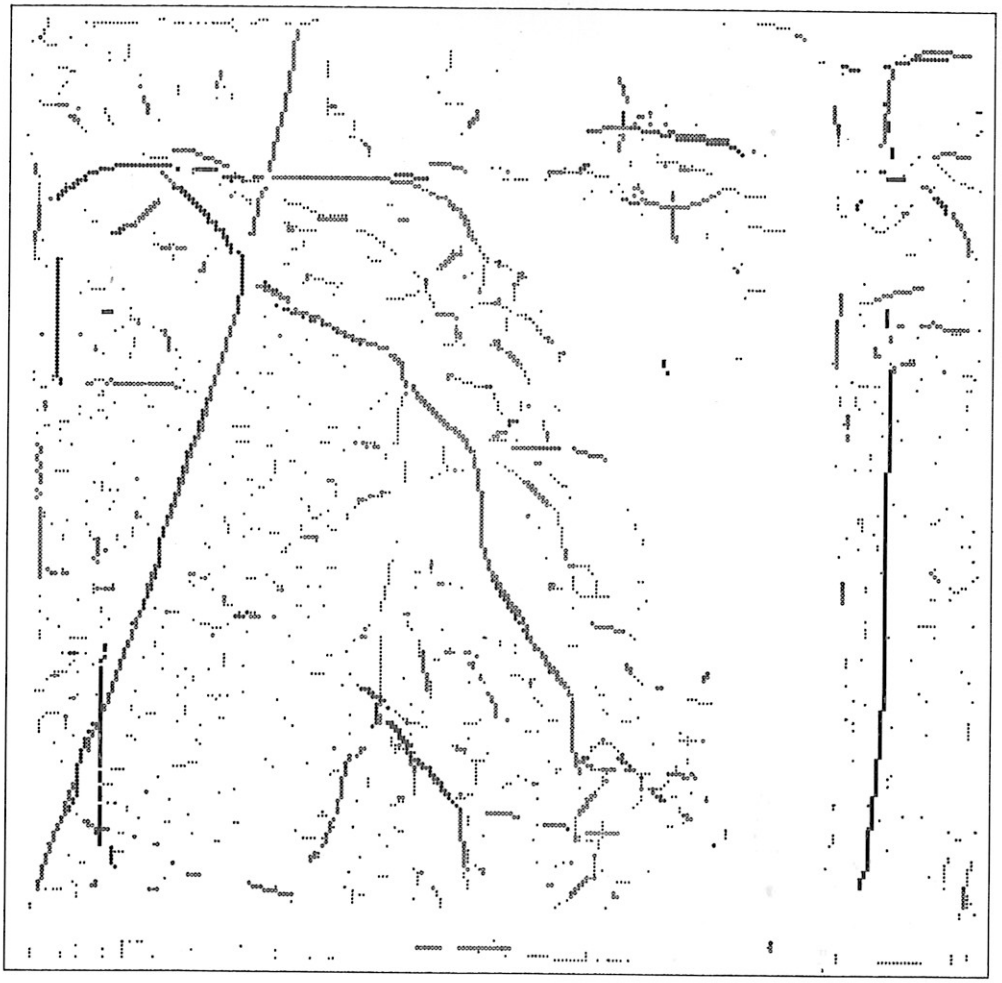


**Figure 27.**

(*c*) A similar representation of the filtered skeleton together with the level of skeleton pixels.

(*d*) A photograph of the pseudomask difference image built with the filtered skeleton. The contrast has been enhanced by a square root function of the grey level (in order to enhance low grey levels).

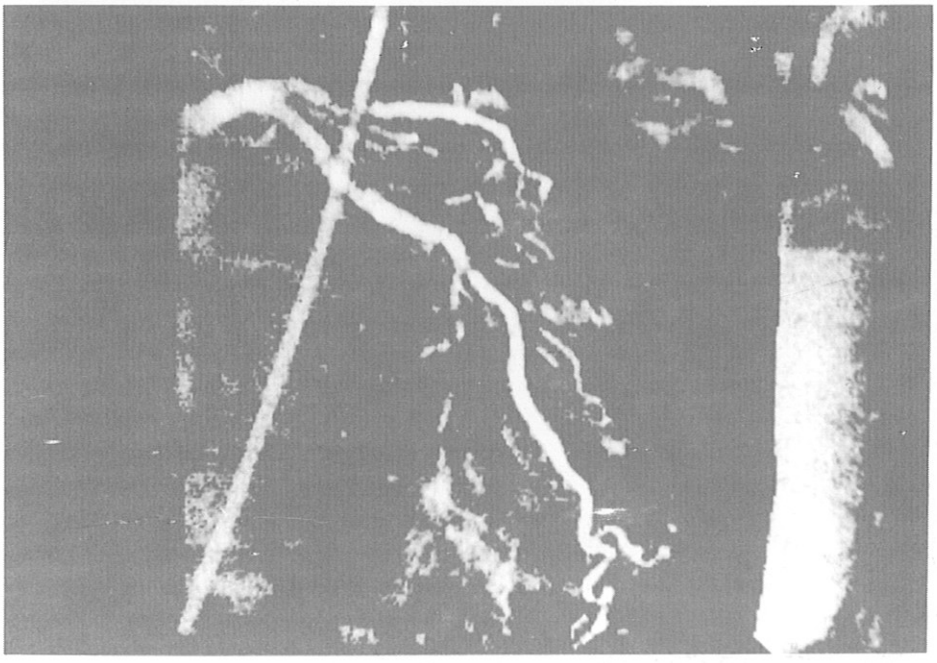
(*a*) *Original image.*



(*b*) *Skeleton.*

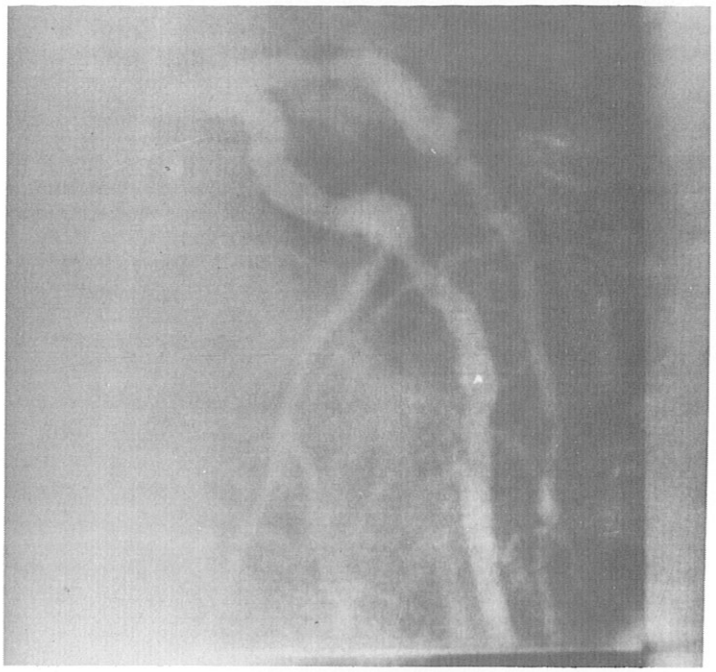
**Figure 28.**

(*c*) *Filtered skeleton.*

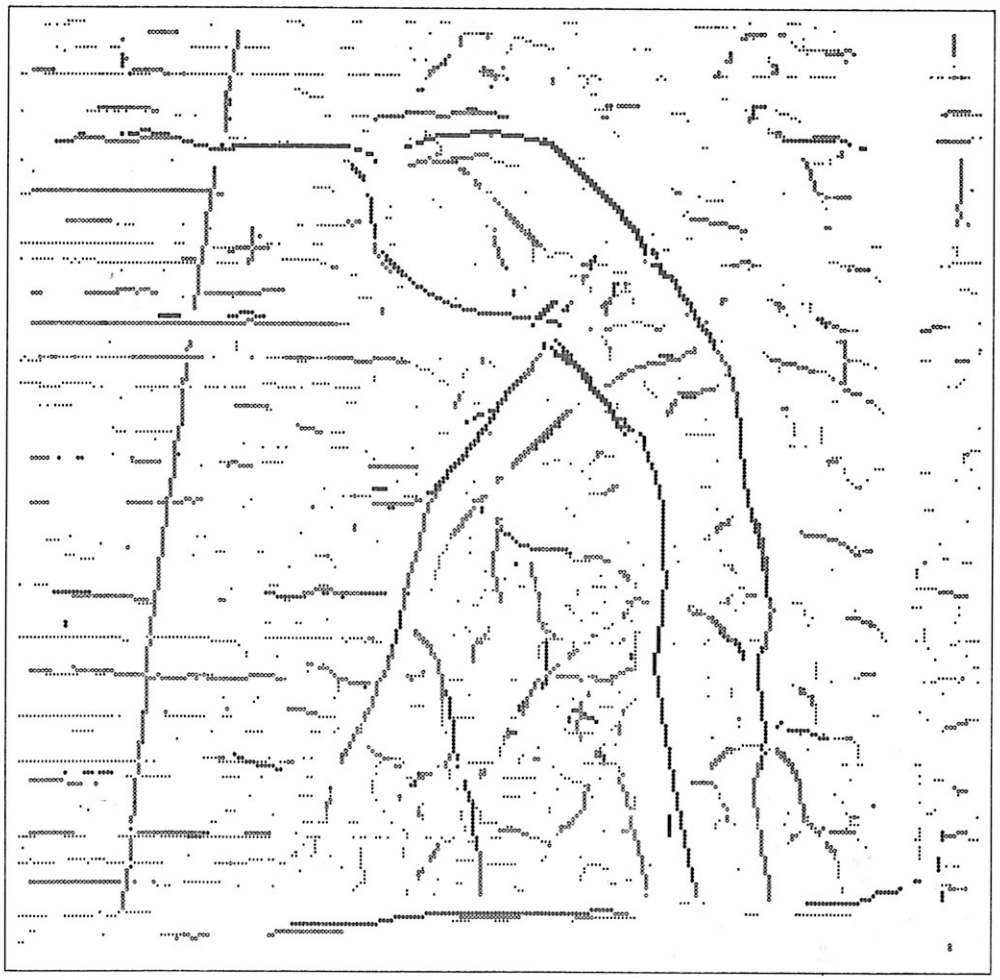


(*d*) *Pseudomask difference image.*

**Figure 28.** *(contd.)*

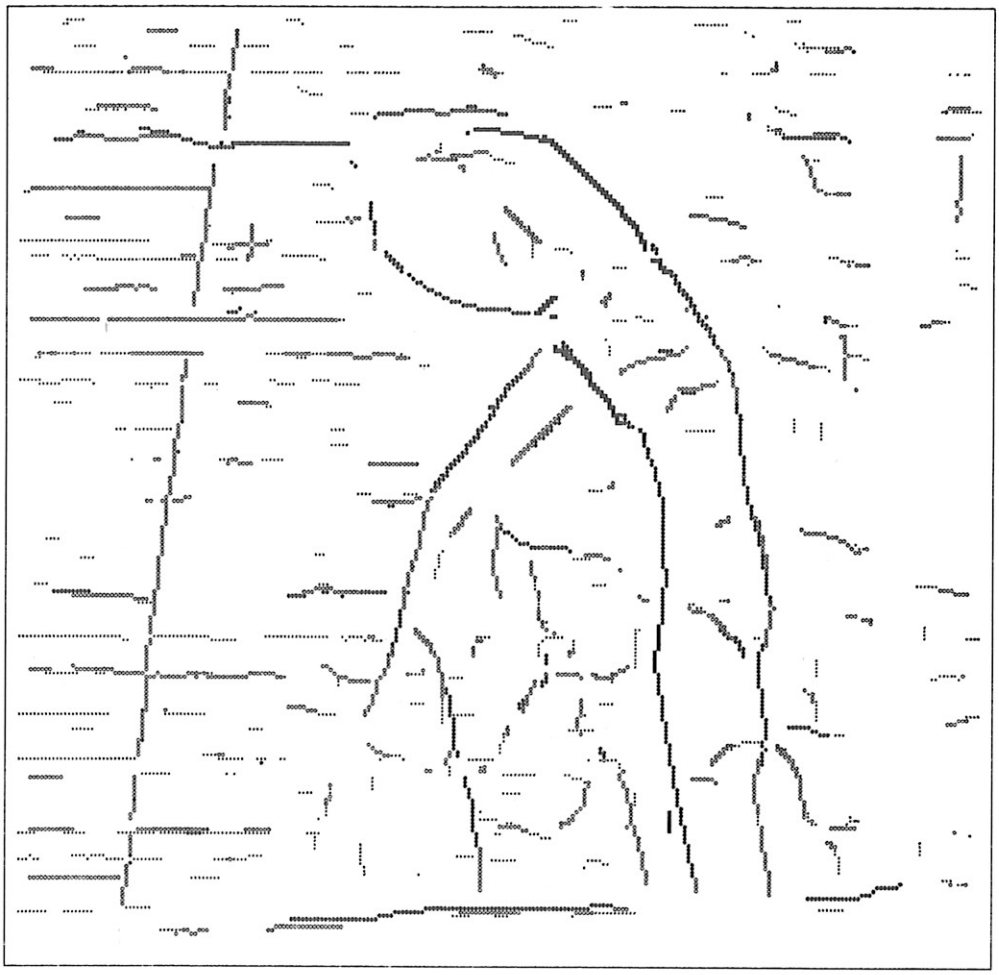


(*a*) *Original image.*

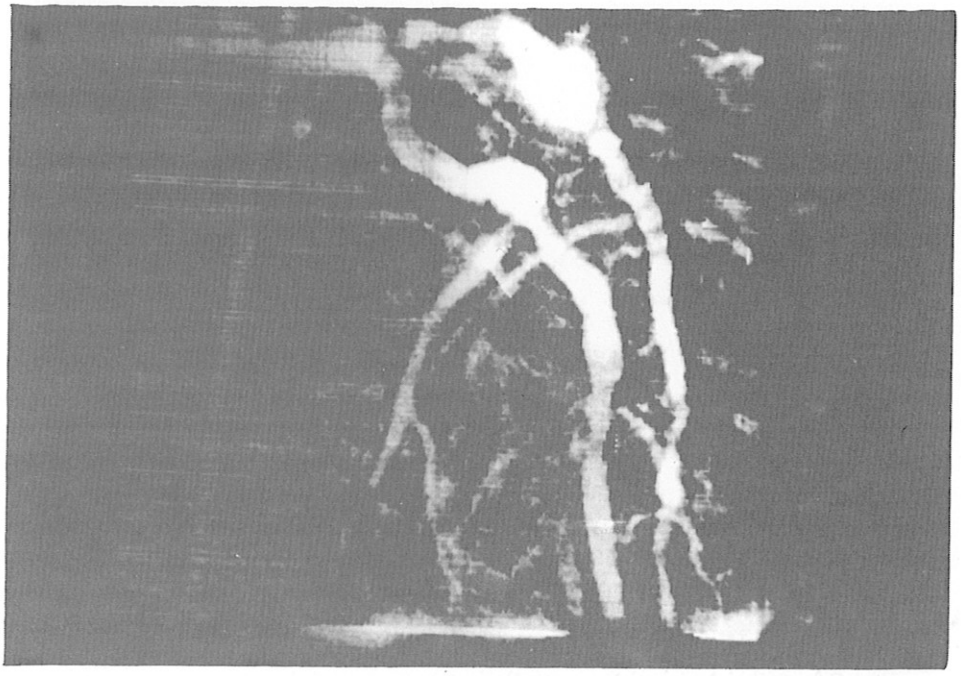


(*b*) *Skeleton.*

**Figure 29.**

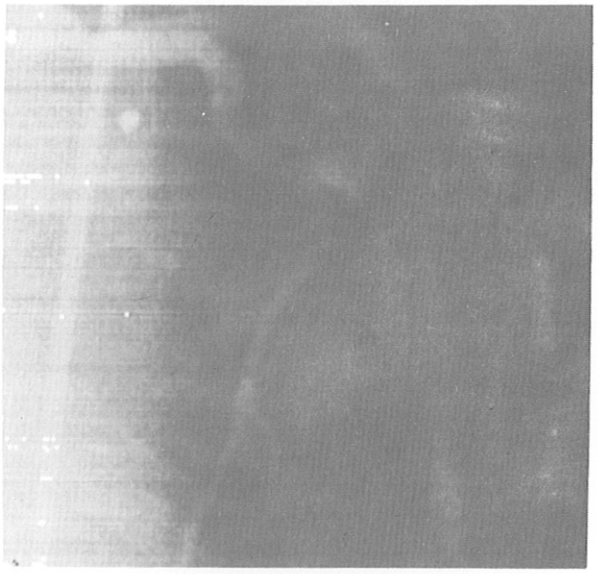


(*c*) *Filtered skeleton.*

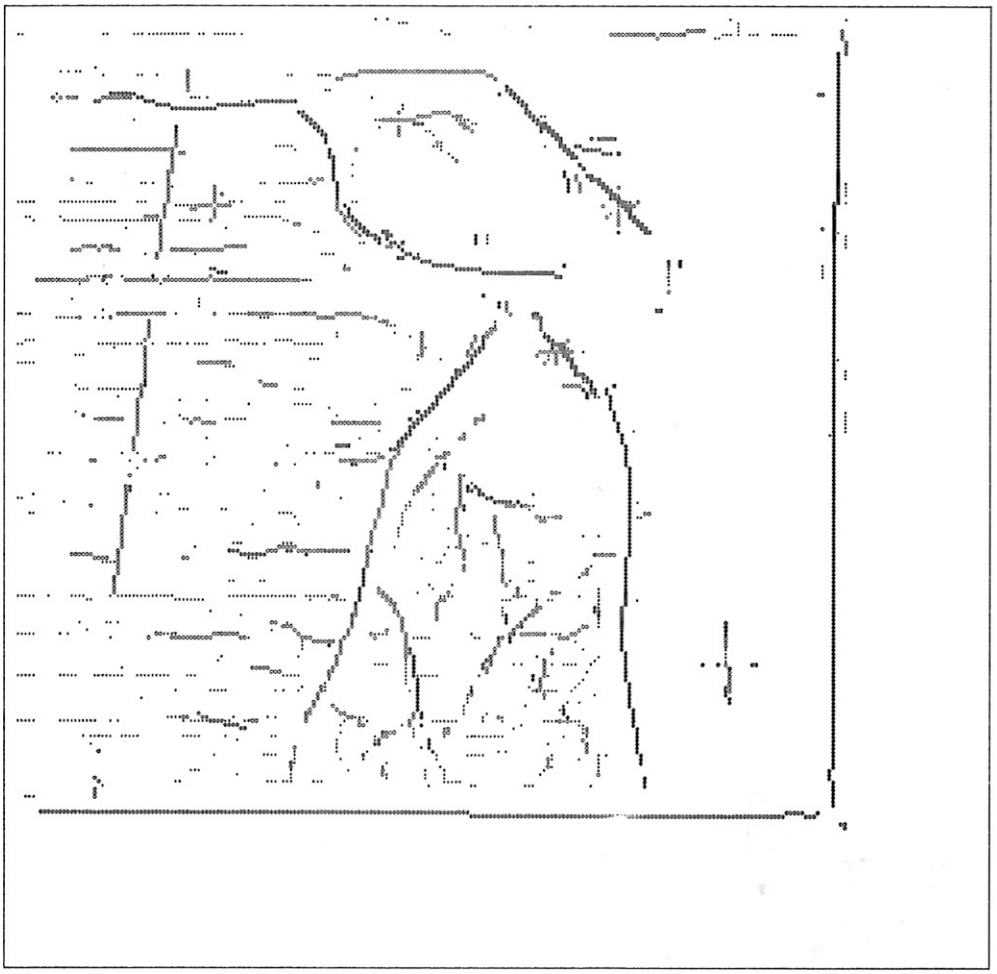


(*d*) *Pseudomask difference image.*

**Figure 29.** *(contd.)*

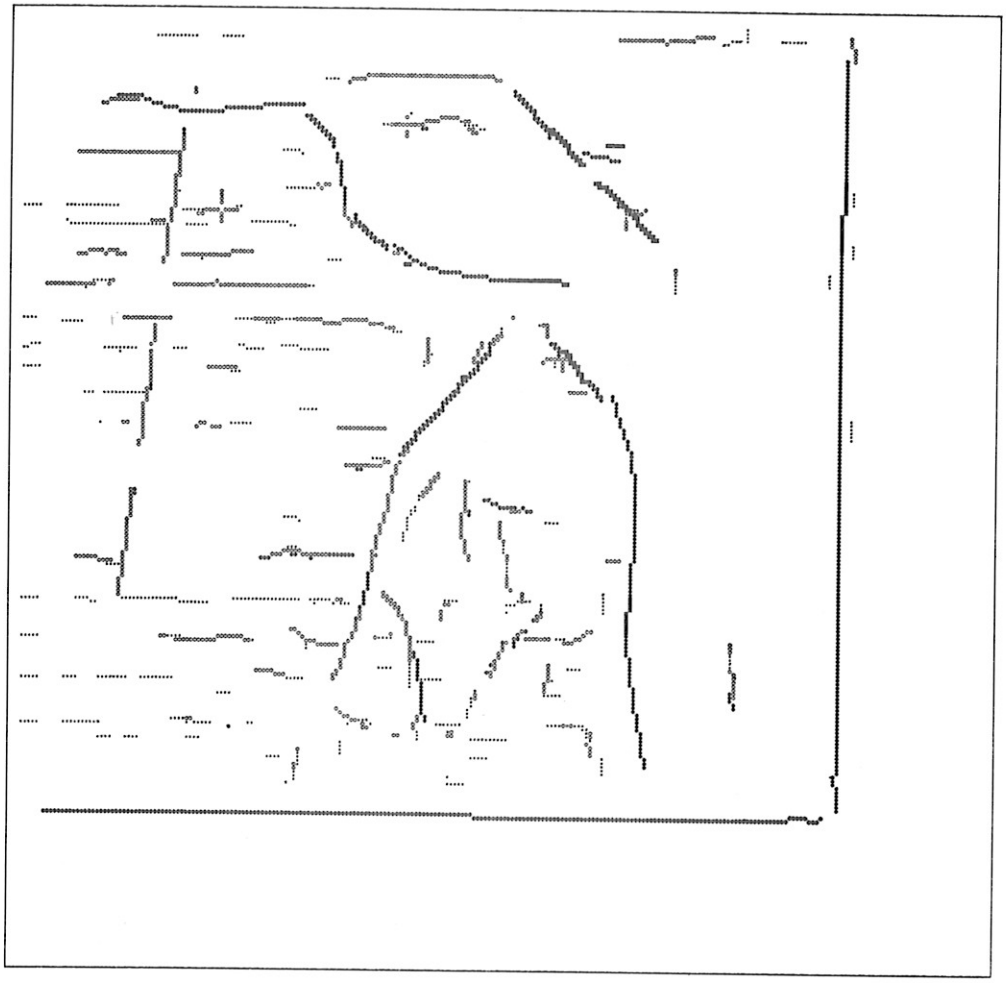


(*a*) *Original image.*

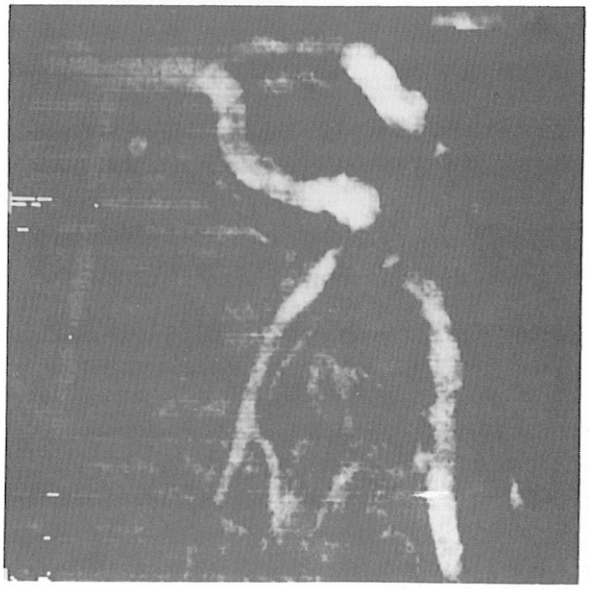


(*b*) *Skeleton.*

**Figure 30.**

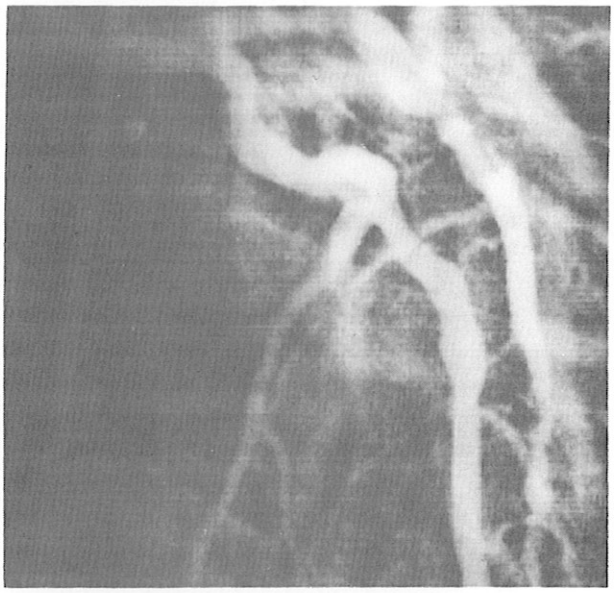


(*c*) *Filtered skeleton.*

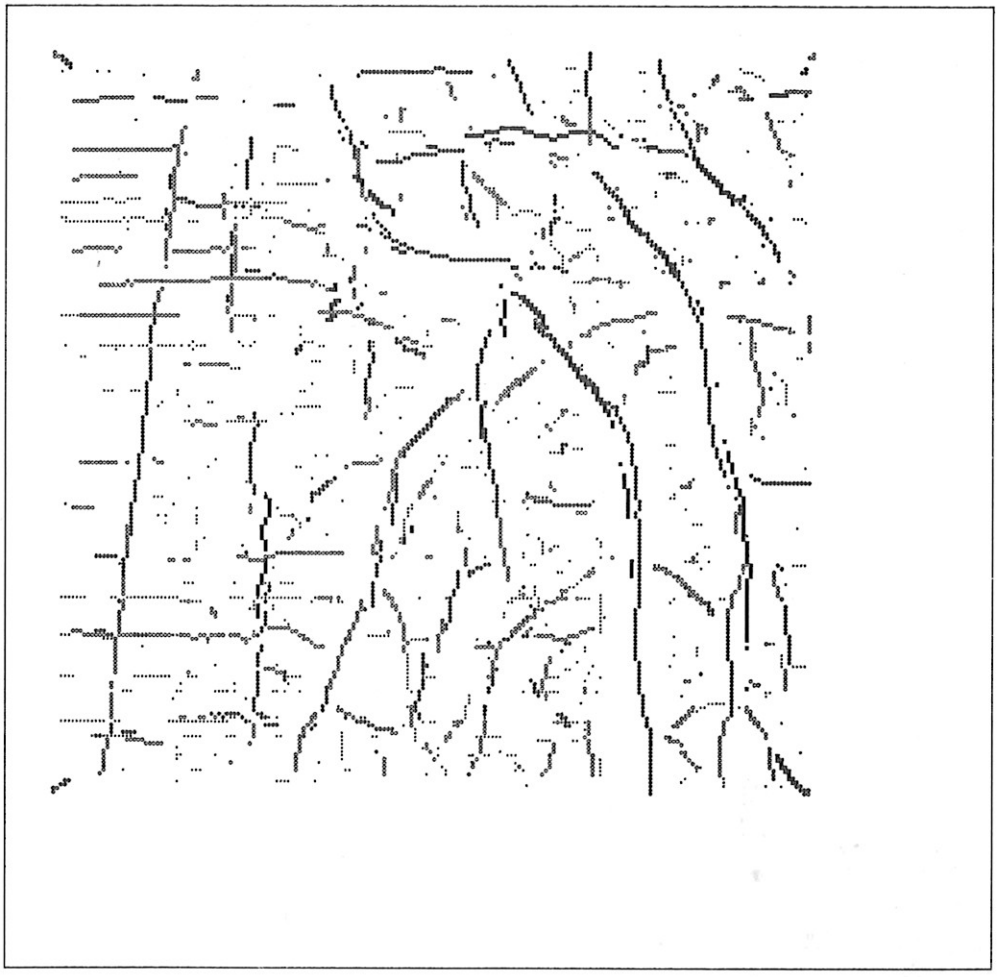


(*d*) *Pseudomask difference image.*

**Figure 30.** *(contd.)*

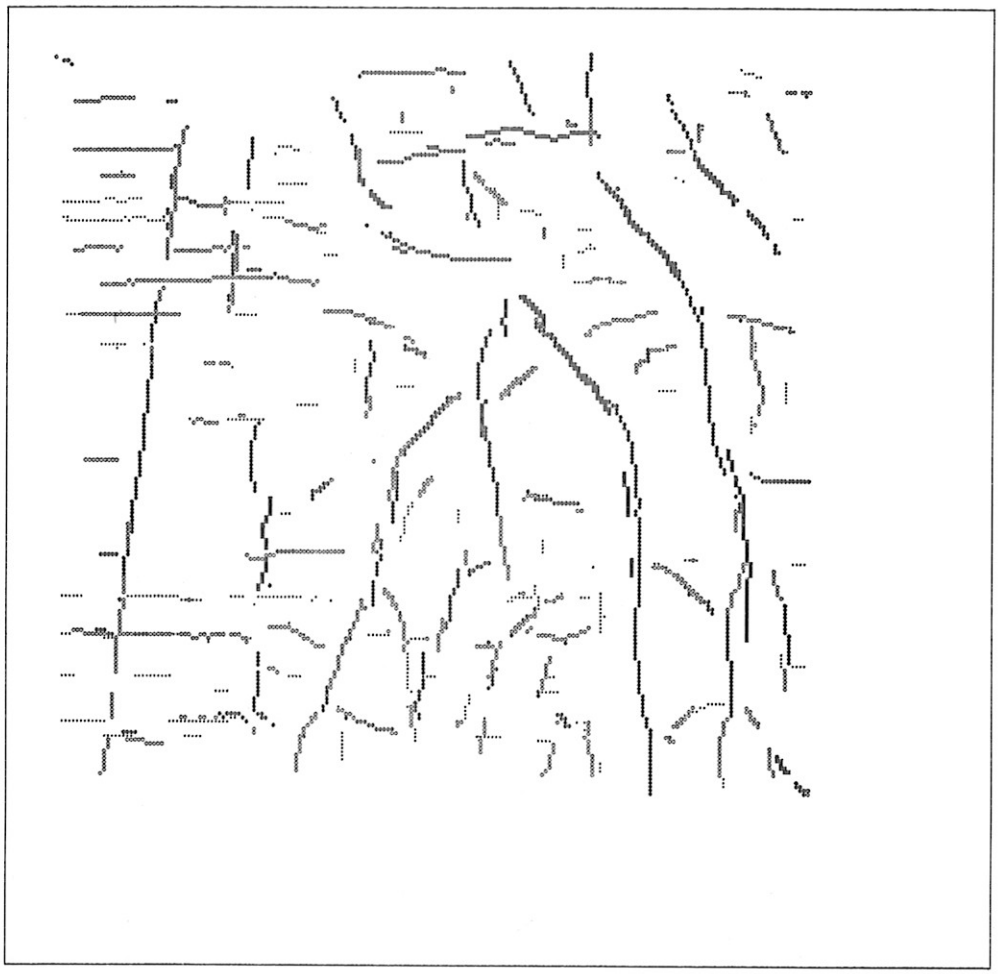


(*a*) *Original image.*

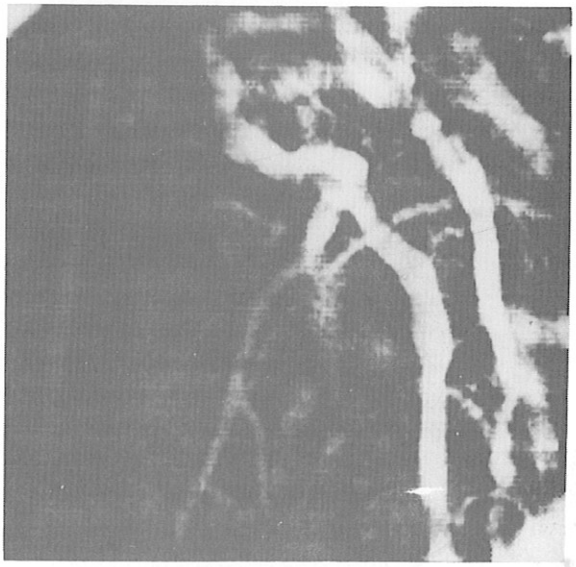


(*b*) *Skeleton.*

**Figure 31.**

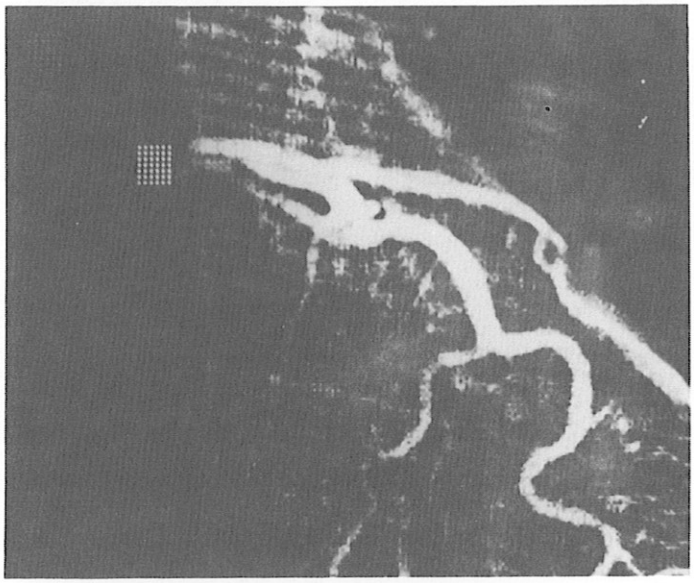


(*c*) *Filtered skeleton.*

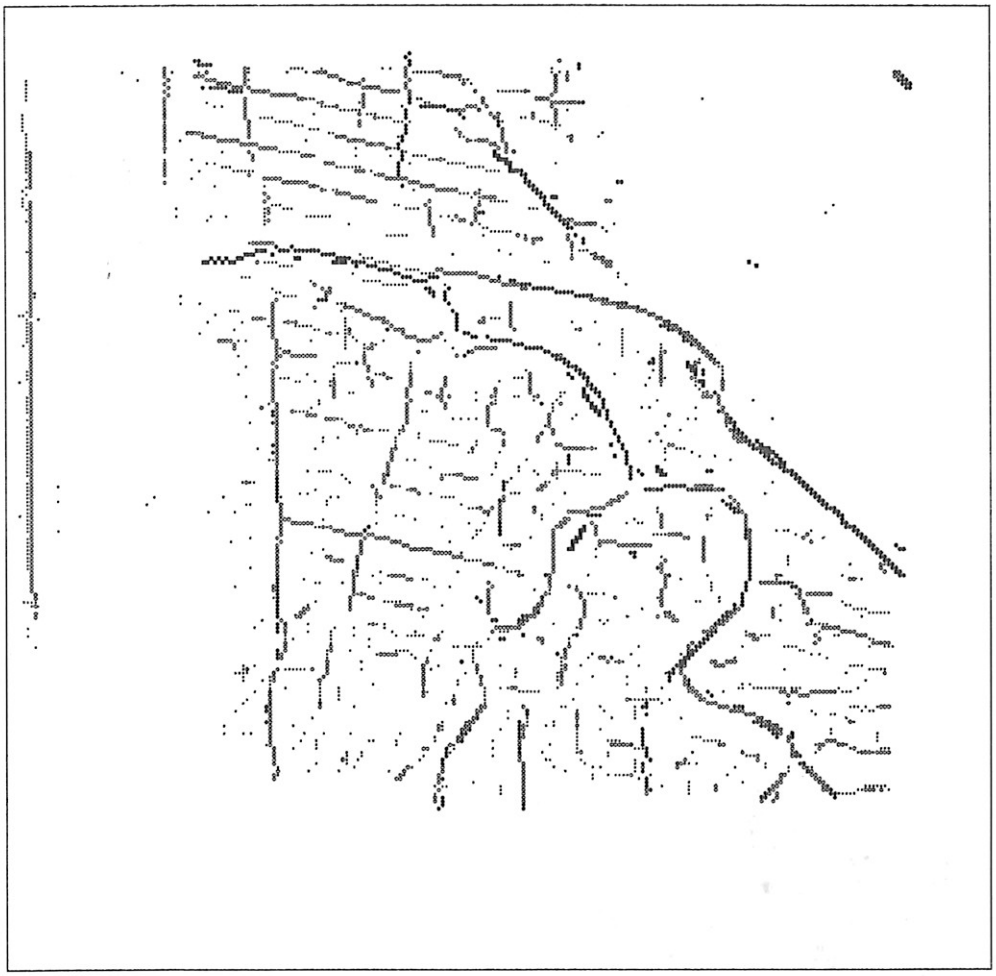


(*d*) *Pseudomask difference image.*

**Figure 31.** *(contd.)*

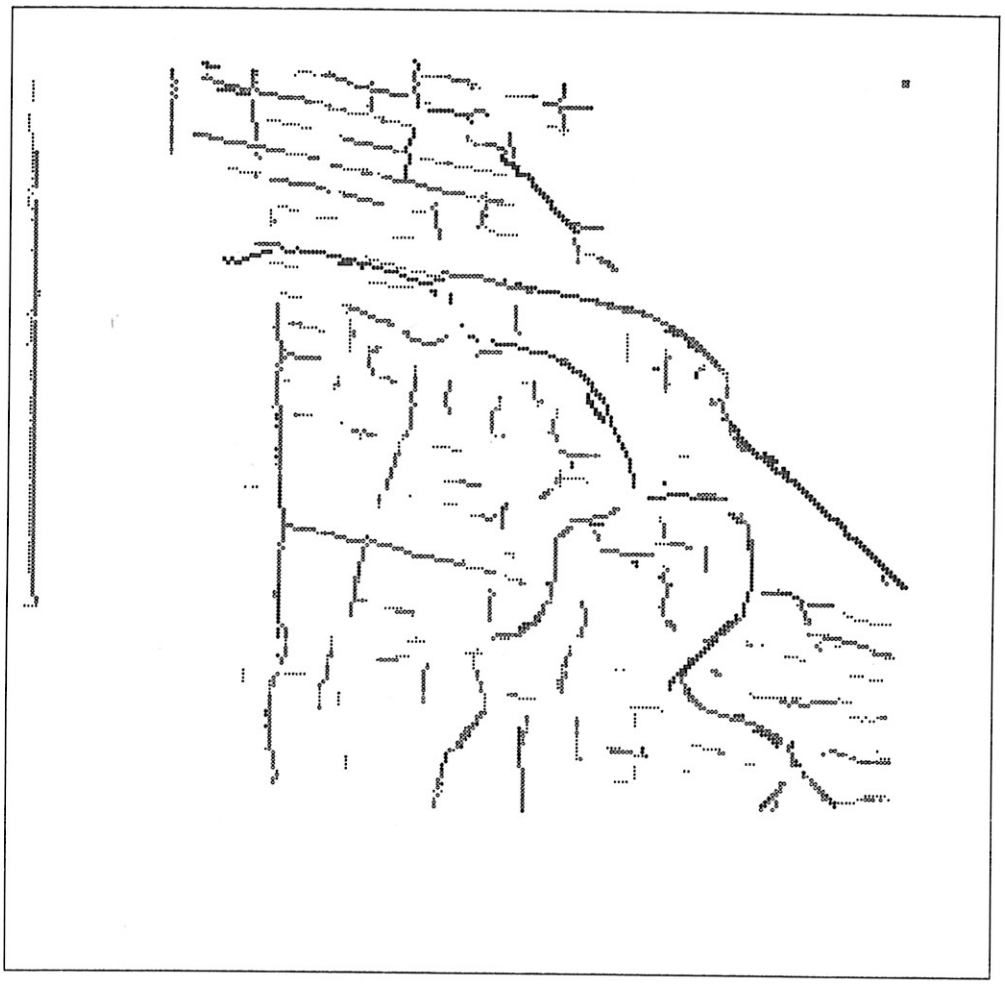


(*a*) *Original image.*

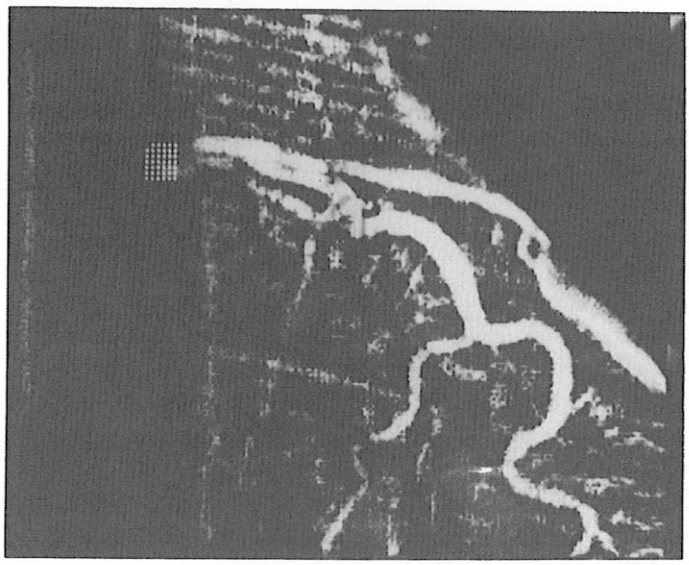


(*b*) *Skeleton.*

**Figure 32.**

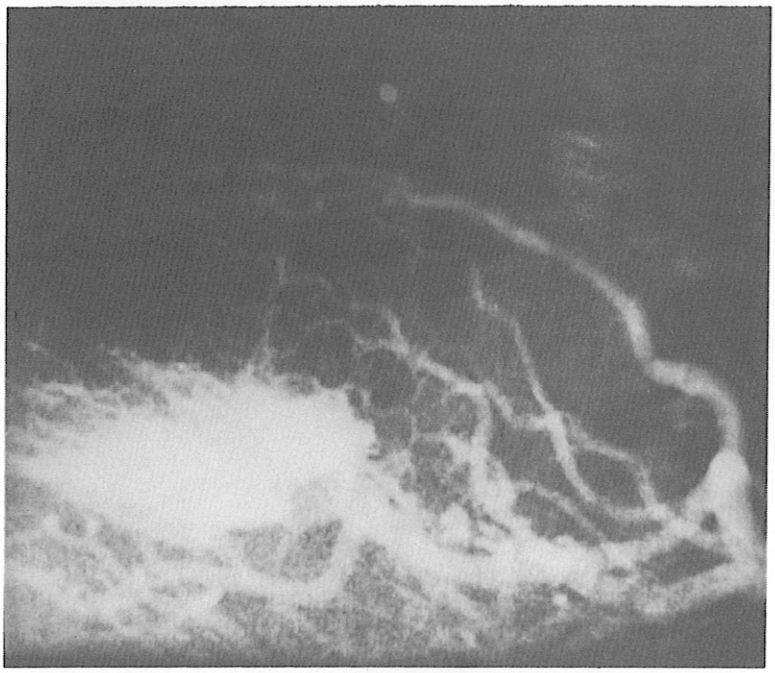


(*c*) *Filtered skeleton.*

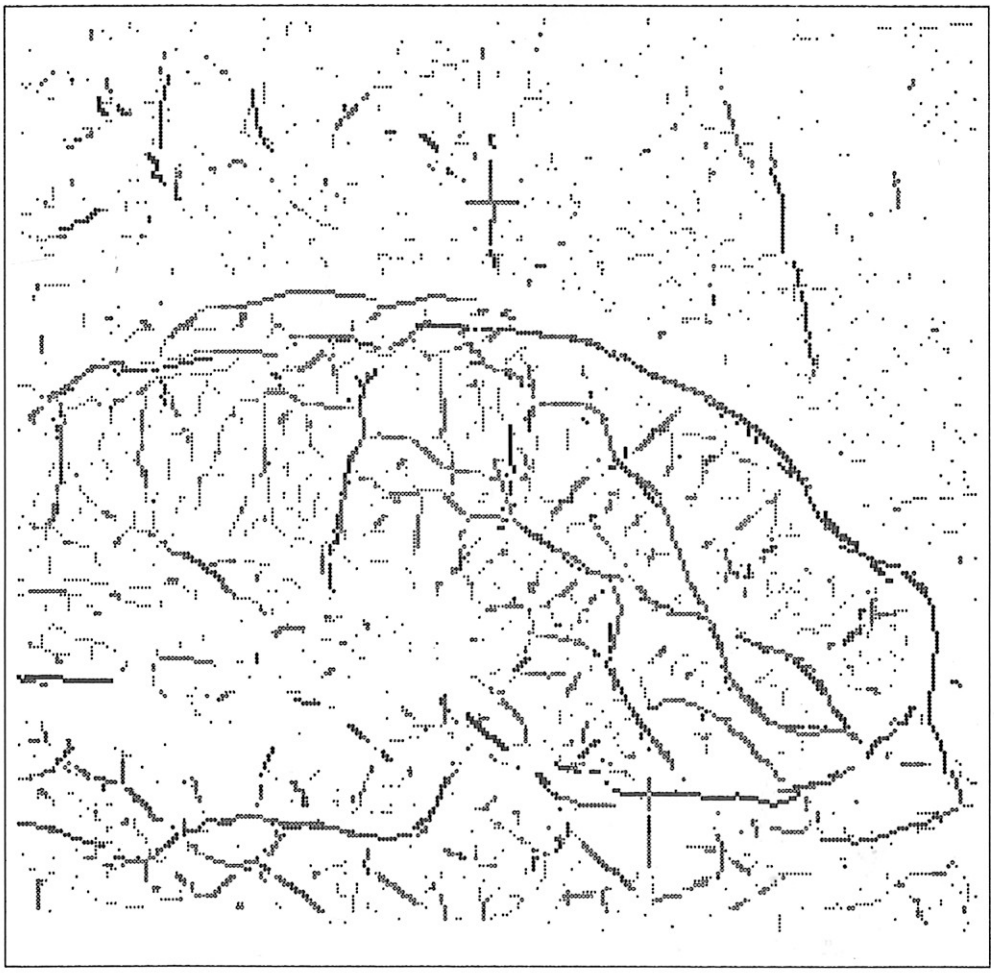


(*d*) *Pseudomask difference image.*

**Figure 32.** *(contd.)*

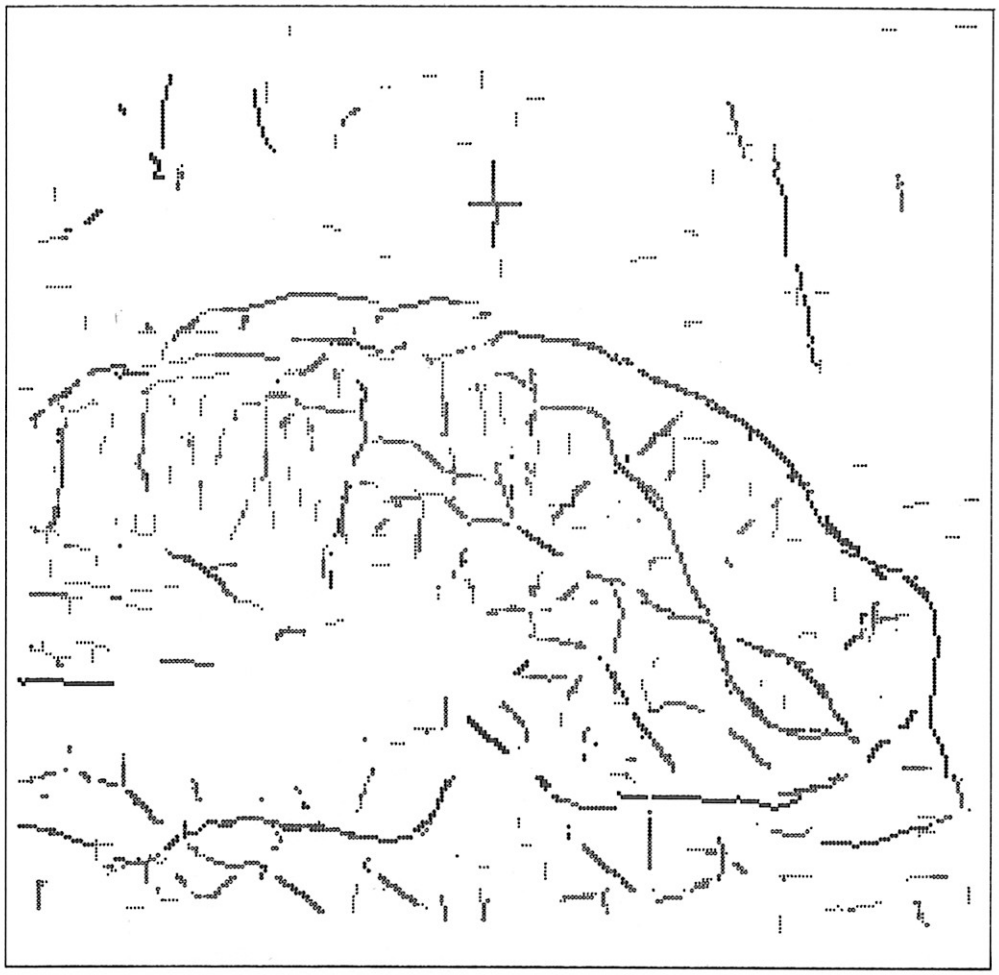


(*a*) *Original image.*

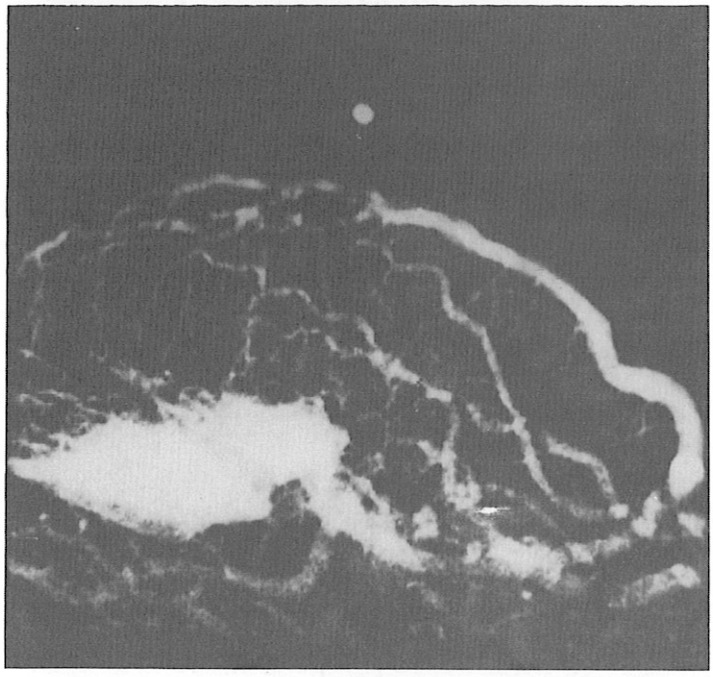


(*b*) *Skeleton.*

**Figure 33.**



(*c*) *Filtered skeleton.*



(*d*) *Pseudomask difference image.*

**Figure 33.** *(contd.)*