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

Working Document WD47

Extraction of Narrow Peaks and Ridges in Images by Combination of Local Low Rank and Max Filters: Implementation and Applications to Clinical Angiography

Christian Ronse

October 1988

Abstract: *We describe a non-linear filter for removing from a digital image narrow positive features, such as peaks and ridges. It is based on a composition of a low rank filter by a max filter, and is thus a generalization of the min-max filter of Nakagawa and Rosenfeld. This new filter is an algebraic opening, and its relation to Mathematical Morphology is briefly discussed.*

The low rank in the first filtering step can be adjusted according to the noise level of the image. An economical PASCAL implementation is given. Results on digitized X-ray arterial images are shown, indicating the possible use of this filter in clinical angiography.

I. Description and main properties of the new filter

I.1. Presentation

Suppose that we have a *finite* set E of points (say, a digital grid in one, two, or more dimensions), and a set $\mathcal{T}$ of *templates* in E (in other words particular subsets of E, for example squares or circles of a given diameter). For any point $p \in E$, write $\mathcal{T}_p$ for the set of all templates $T \in \mathcal{T}$ containing p. We consider images $E \to G$, where G is a set of grey-levels; each such image I associates to every point $p \in E$ a grey-level $I(p)$.

Consider an image I from which we want to erase all narrow peak features (for example isolated peaks or ridges). By 'narrow' we mean 'not large enough to contain a template $T \in \mathcal{T}$'. For example if E is a two-dimensional grid and $\mathcal{T}$ consists in all $d \times d$ squares in E, then 'narrow' means 'whose width or height is less than d'. Let us now explain what we mean by 'erasing' narrow peaks. A peak is a set of points having a relatively higher grey-level than the background. Consider a peak S, in other words a set S of points whose grey-level is not smaller than a background grey-level g. If S contains a template $T \in \mathcal{T}$, then the peak on T (w.r.t. the background grey-level g) will be preserved, in other words the points in T will keep their grey-level $\geq g$. Thus for any point $p \in E$ and grey-level g such that $I(p) \geq g$, the grey-level of p in the new image will still be $\geq g$, provided that there exists a template $T \in \mathcal{T}$ containing p, such that all points in T have grey-level $\geq g$. Hence we obtain from I a new image I' defined by

$$I'(p) = \max_{T \in \mathcal{T}_p} \min_{r \in T} I(r) \qquad \text{for} \quad p \in E. \tag{1}$$

(Note that when $\mathcal{T}_p$ is empty, in other words no element of $\mathcal{T}$ contains p, then $I'(p)$ is set equal to the the smallest grey-level in G. But this generally does not happen in practice, because $\mathcal{T}$ is usually chosen to cover the whole of E).

With this operation, in every peak each portion not large enough to contain an element of $\mathcal{T}$ is erased. This is illustrated in Figure 1 in the case where E is a one-dimensional grid and $\mathcal{T}$ consists in the set of all segments of a given length d. Here all peak portions narrower than d are simply levelled, leaving only the portions of width at least d.

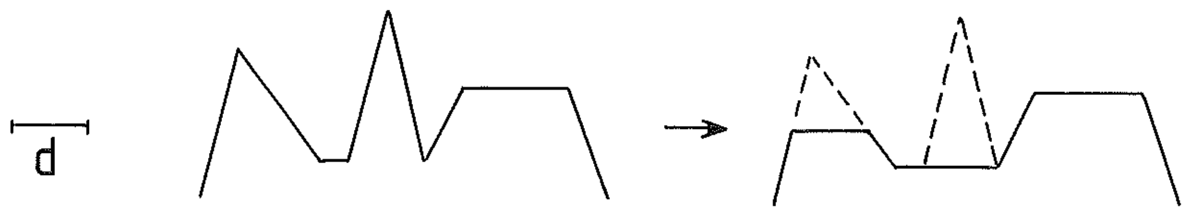

Figure 1. *Levelling all peaks narrower than d.*

One usually takes as templates in $\mathcal{T}$ local windows associated to points of E, in other words to each $p \in E$ corresponds a window $W(p)$ (generally containing it), and $\mathcal{T}$ is the set of all $W(p)$ for $p \in E$. Now given any $p \in E$, we define the dual window $W^*(p)$ by

$$q \in W^*(p) \quad \text{iff} \quad p \in W(q). \tag{2}$$

Thus $\mathcal{T}_p$ is the set of all $W(r)$ containing p, in other words of all $W(q)$ for $q \in W^*(p)$. Then (1) becomes, by substituting $W(q)$ for T:

$$I'(p) = \max_{q \in W^*(p)} \min_{r \in W(q)} I(r) \qquad \text{for} \quad p \in E. \tag{3}$$

We get thus the min-max filter of Nakagawa and Rosenfeld [4]; it is the composition

$$Max_{W^*} \circ Min_W \tag{4}$$

of the min filter Min_W followed by the max filter Max_{W^*}, these two filters being defined by

$$\begin{aligned} Min_W(I)(p) &= \min_{q \in W(p)} I(q) \\ \text{and} \qquad Max_{W^*}(I)(p) &= \max_{q \in W^*(p)} I(q) \end{aligned} \qquad \text{for} \quad I : E \to G \quad \text{and} \quad p \in E. \tag{5}$$

Often one choose the windows $W(p)$ as translates of a fixed template B about the origin, in other words for each $p \in E$ we have:

$$W(p) = \{p + b \mid b \in B\}. \tag{6}$$

Then the dual windows $W^*(p)$ are translates of the template obtained from B by a central symmetry, in other words:

$$W^*(p) = \{p - b \mid b \in B\}. \tag{7}$$

The filter described above succeeds in removing from an image narrow peaks and ridges. Then by subtracting the filtered image from the original one, one obtains an enhanced image of these peaks and ridges. This will indeed be shown in Chapter III with angiographic images. However, such a filter is too sensitive to dark speckles, for example isolated points of low grey-level. This is shown in Figure 2, where the same filter as in Figure 1 is applied to an image corrupted by speckle noise. One sees that a few isolated points of low grey-level are sufficient to provoke the deletion of peaks which would have otherwise been preserved.

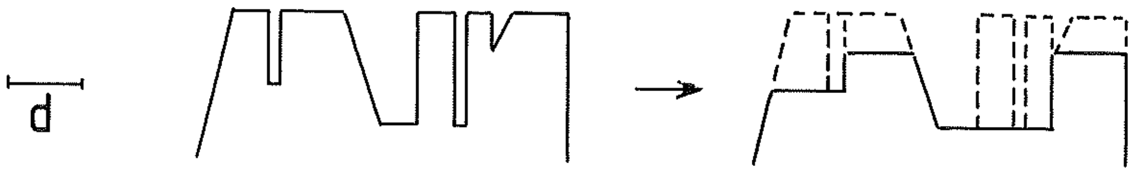

Figure 2. *Peaks wider than d levelled because of holes.*

We want to have a filter with the same global behaviour as that min-max filter, but which preserves such peaks. As we will explain below, this will be possible by using a low rank filter instead of the min filter in (4), and taking the minimum between the original image and the one resulting from the low rank and max filterings. The low rank can be chosen according to the degree of speckle noise. Then a wide peak feature with small holes in it will be preserved (with its holes of course). Moreover, by taking templates of small width in $\mathcal{T}$, our new filter can also be used to erase bright speckle noise, for example isolated points of high grey-level. Other advantages will be mentioned later. The comparison between the min-max filter and our new one will be illustrated in practice in Chapter III, where we will present results on angiographic images.

Let us analyse the behaviour of the min-max filter more closely. Write I for the original image, and I' for the one obtained after filtering. Given an arbitrary background grey-level g, we will call 'high' all grey-levels $\geq g$, and 'low' all those $< g$. A peak can be considered as a set of points with 'high' grey-level. The behaviour of the min-max filter is as follows:

A point p will have a 'high' grey-level $I'(p)$ in the filtered image iff there is at least one template $T \in \mathcal{T}$ containing p, such that all points $q \in T$ have 'high' grey-levels $I(q)$ in the original image. (This requires in particular that $I(p)$ is high).

The problem is that if we have a template T such that all points of T except one have 'high' grey-levels, the peak on T will not be preserved in I'. We must thus relax the condition that *"all points $q \in T$ have 'high' grey-levels $I(q)$ in the original image"*. Let us say *"most points"* instead of *"all points"* (we will give below the precise meaning of 'most'). Now if *most points* $q \in T$ have 'high' grey-levels $I(q)$, this does not require that $I(p)$ is high (p can be an exception); but then a 'low' grey-level $I(p)$ in the original image can give rise to a 'high' grey-level $I'(p)$ in the filtered image, in other words a hole in the peak on template T can be filled. We do not want this, and so we make the specific requirement that *"p has a 'high' grey-level $I(p)$ in the original image I"* in order to get a 'high' $I'(p)$ (this requirement was redundant in the min-max filter, where *"all points"* in T included p). We get thus the following general behaviour:

A point p will have a 'high' grey-level $I'(p)$ in the filtered image iff p has a 'high' grey-level $I(p)$ in the original image, and there is at least one template $T \in \mathcal{T}$ containing p, such that most points $q \in T$ have 'high' grey-level $I(q)$ in the original image.

We have thus defined a new class of filters, whose members are specified by the particular meaning that we give to the expression 'most points'. This class was first presented in [5], and a few examples were given. Let us recall two of them here.

Suppose first that *"most points"* means *"all points, except at most $k-1$"*, where k is a small positive integer ($k = 1$ for the min-max filter). Then the filter is given by the following equation generalizing (1), where rank^k is the k-th rank function (which selects the k-th smallest element in a sample):

$$I'(p) = \min\{I(p), \max_{T\in\mathcal{T}_p} \operatorname*{rank}^k_{r\in T} I(r)\} \qquad \text{for} \quad p \in E. \tag{8}$$

In the case where the templates in $\mathcal{T}$ are windows associated to points of E, then the filter has the decomposition

$$\mathbf{id} \wedge [Max_{W^*} \circ R^k_W] \tag{9}$$

generalizing (4), where $\wedge$ is the minimum operation between two filters, $\mathbf{id}$ is the identity filter, and R^k_W is the k-th rank filter defined by:

$$R^k_W(I)(p) = \operatorname*{rank}^k_{q\in W(p)} I(q) \qquad \text{for} \quad I : E \to G \quad \text{and} \quad p \in E. \tag{10}$$

In other words, our new filter applies to an image I the composition of the k-th rank filter R^k_W followed by the max filter Max_{W^*}, and then takes the minimum between the resulting image and the original one I.

Suppose next that *"most points"* means *"all points, except isolated ones"*, in other words a peak on template T can be levelled only if there are two adjacent points of T not belonging to that peak. Then in (8) we must replace the function rank^k by the function $mm[I, T]$ which takes the minimum of all maxima of greylevels in pairs of adjacent points of T, in other words defined by

$$mm[I,T] = \min_{\{p,q\} \in A(T)} \max\{I(p), I(q)\}, \tag{11}$$

where $A(T)$ is the set of pairs of adjacent points in T. We get thus:

$$I'(p) = \min\{I(p), \max_{T \in \mathcal{T}_p} mm[I,T]\} \qquad \text{for} \quad p \in E. \tag{12}$$

Similarly (9) becomes

$$\mathbf{id} \wedge [Max_{W^*} \circ Mm_W], \tag{13}$$

where Mm_W is the filter corresponding to the function $mm[I, T]$.

As can be seen in Figure 3, in the case where E is a one-dimensional grid and $\mathcal{T}$ consists in the set of all segments of length d, the above filter will not level peaks of width $\geq d$ having isolated holes of width 1. It will preserve them, without filling their holes of course.

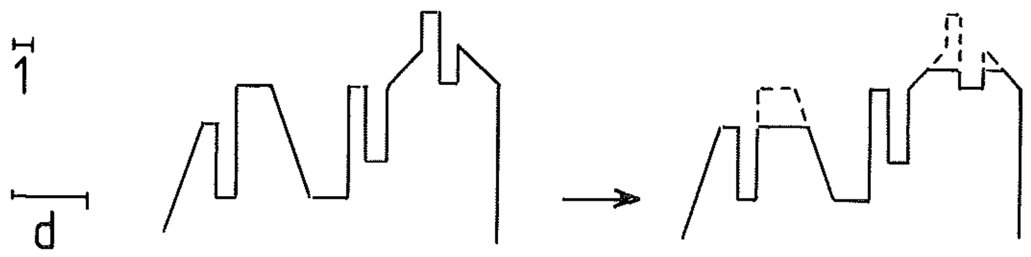

Figure 3. *Peaks wider than d with isolated holes of width 1 are preserved.*

Now (12) gives the general form of our new type of filters, where $mm[I, T]$, instead of being defined as in (11), can be any function giving a result $\geq g$ (a 'high' grey-level) whenever 'most' of the points of T have a grey-level $\geq g$. If $\mathcal{M}(T)$ is the set of minimal subsets of T which can be considered as containing 'most' points of T, then $mm[I, T]$ is given by

$$mm[I,T] = \max_{S \in \mathcal{M}(T)} \min_{r \in S} I(r). \tag{14}$$

In other words, $mm[I, T]$ can be any composition of min and max functions on grey-levels of points in T. Consequently in (13) Mm_W can be any filter applying a composition of min and max functions inside each window $W(p)$. It is in this form (13) that our class of filters was presented in [5], and other examples of such filters were suggested. As we will see later in this chapter, all filters given by (12) share some general properties.

However for practical purposes the implementation and angiographic application of such filters will be restricted to the ones expressed in the form of (9), where all windows $W(p)$ are translate of a given rectangular mask (see (6) and (7)). In other words, we will consider those filters where each template is a $h \times w$ window, and the function applied within each window is the k-th rank filter, with $1 \leq k \leq hw$.

In Section I.2 we will state without proof general mathematical properties of the filters given by (12) and (14), as particular cases of a wider class of image operators called *openings*. Then in Section I.3 some practical and technical considerations on the behaviour of the rank-max filters of (9) will be discussed. The PASCAL implementation of these rank-max filters will be described in Chapter II, and their application to digitized angiographic images will be the object of Chapter III.

I.2. General mathematical properties

Unless one is interested in the general theoretical background in which our new class of filters has been proposed, this section can be skipped. The main properties of the subclass of rank-max filters that we have implemented (see Chapter II) will be recalled in Section I.3.

Call $\mathcal{M}$ the set of filters given by (12) and (14). The first important property of the elements of $\mathcal{M}$ is that they are what one calls (algebraic) *openings*. These are generally defined as operators on general algebraic structures called *complete lattices* (see [2,7]), but we will consider them here as operators on the set $\mathcal{I}$ of images $I : E \to G$. Indeed, if we assume that G contains the adherent points of every sequence of elements in it (that is, it is closed in the compact topological space $\mathbf{R} \cup \{\pm\infty\}$ of real numbers with $\pm\infty$ added), then $\mathcal{I}$ will effectively be a complete lattice.

Let us thus first recall the definition and main properties of an opening (on $\mathcal{I}$). The ordering $\leq$ on grey-levels in G extends naturally to images $E \to G$. Indeed, given two images $I, J : E \to G$, we write $I \leq J$ iff $I(p) \leq J(p)$ for every point $p \in E$, and this can also be written $J \geq I$. A map $\alpha : \mathcal{I} \to \mathcal{I}$ is called an *opening* iff it is

(i) *antiextensive*: for any $I \in \mathcal{I}$, $\alpha(I) \leq I$.

(ii) *idempotent*: $\alpha \circ \alpha = \alpha$, in other words for any $I \in \mathcal{I}$, $\alpha(\alpha(I)) = \alpha(I)$.

(iii) *increasing*: for any $I, J \in \mathcal{I}$, $I \leq J$ implies that $\alpha(I) \leq \alpha(J)$.

As announced, we have the following:

Proposition 1. *A filter in $\mathcal{M}$ is an opening.*

We will not prove this result here. Note that the fact that such a filter applies only compositions of min and max functions to grey-levels, implies (see [6]) that its behaviour is determined in a unique way by that of its restriction to binary grey-levels. One has thus to prove this proposition only for $G = \{0, 1\}$. That it is antiextensive and increasing ((i) and (iii) above) is fairly evident, but idempotence (ii) requires some reasoning. Note also that it can be shown that such a type of filter is a particular case of a new type of opening defined in [7] for complete lattices.

What does it mean in practice that a filter $F \in \mathcal{M}$ is an opening? Antiextensivity means that F will delete some positive features from an image I. Idempotence means then that once F has deleted some features, nothing remains to be deleted in a further application of F; in other words F converges in one pass. Increasingness implies then that if F removes some positive feature from an image, it should also remove any smaller positive feature. Thus an opening is an operator which removes from an image all positive features smaller than 'something', and does its job in one pass. The definition of a particular opening depends then on the definition of that 'something'.

In the algebraic theory underlying Mathematical Morphology (see [2,7]), an opening is characterized by its set of invariants, and the latter set has an interesting structure. Before explaining this in detail, we must introduce the structure of *complete lattice* on $\mathcal{I}$.

We assume that G contains the adherent points of any sequence of elements inside itself. This is the case for example if G is finite, or if $G = \mathbf{R} \cup \{\pm\infty\}$. Then any set U of grey-levels has a *supremum* $\sup U$ and an *infimum* $\inf U$, both belonging to G. They are defined as follows:

— The supremum $\sup U$ is the least upper bound of U, in other words $\sup U \geq u$ for every $u \in U$, and for any $x \in G$ such that $x \geq u$ for every $u \in U$, $\sup U \leq x$.

— The infimum $\inf U$ is the greatest lower bound of U, in other words $\inf U \leq u$ for every $u \in U$, and for any $x \in G$ such that $x \leq u$ for every $u \in U$, $\inf U \geq x$.

In particular G has a greatest element $\mathbf{I}_G = \sup G$ and a least element $\mathbf{O}_G = \inf G$. In the same way as one sets an empty sum equal to zero and an empty product equal to one, we set $\sup \emptyset = \mathbf{O}_G$ and $\inf \emptyset = \mathbf{I}_G$. The existence of the supremum and infimum of any subset of G defines it as a *complete lattice*.

As the order relation $\leq$ on G extends to $\mathcal{I}$, the complete lattice structure of G extends also to $\mathcal{I}$. Given a subset $\mathcal{J}$ of $\mathcal{I}$, the *supremum* $\bigvee \mathcal{J}$ and the *infimum* $\bigwedge \mathcal{J}$ of $\mathcal{J}$ are respectively the lowest upper bound and the highest lower bound of $\mathcal{J}$ in $\mathcal{I}$, and they satisfy the following:

$$\left.\begin{array}{rl} & [\bigvee \mathcal{J}](p) = \sup\{I(p) \mid I \in \mathcal{J}\} \\ \text{and} & [\bigwedge \mathcal{J}](p) = \inf\{I(p) \mid I \in \mathcal{J}\} \end{array}\right. \quad \text{for} \quad p \in E.$$

Also $\mathcal{I}$ has a greatest element $\mathbf{I}_\mathcal{I} = \bigvee \mathcal{I}$ and a least element $\mathbf{O}_\mathcal{I} = \bigwedge \mathcal{I}$ defined by

$$\left.\begin{array}{rl} & \mathbf{I}_\mathcal{I}(p) = \mathbf{I}_G \\ \text{and} & \mathbf{O}_\mathcal{I}(p) = \mathbf{O}_G \end{array}\right. \quad \text{for} \quad p \in E.$$

We have $\bigvee \emptyset = \mathbf{O}_\mathcal{I}$ and $\bigwedge \emptyset = \mathbf{I}_\mathcal{I}$.

We can now turn to the characterization of openings on $\mathcal{I}$ by their invariants. Given an opening α on $\mathcal{I}$, an image $I \in \mathcal{I}$ is called an *invariant* of α if $I = \alpha(I)$. We will write $Inv(\alpha)$ for the set of all invariants of α in $\mathcal{I}$. We need now to introduce another concept in order to characterize these sets $Inv(\alpha)$. Given a subset $\mathcal{J}$ of $\mathcal{I}$, we will say that $\mathcal{J}$ is *sup-closed* if for any subset $\mathcal{K}$ of $\mathcal{J}$, $\bigvee \mathcal{K} \in \mathcal{J}$ (in particular $\mathbf{O}_\mathcal{I} = \bigvee \emptyset \in \mathcal{J}$). We have then the following characterization of openings, whose proof can be found in [7]:

Proposition 2. *There is a bijection between openings on $\mathcal{I}$ and sup-closed subsets of $\mathcal{I}$. An opening α and a sup-closed set $\mathcal{J}$ which correspond under this bijection define each other as follows:*

$$\mathcal{J} = Inv(\alpha).$$

$$\forall I \in \mathcal{I}, \quad \alpha(I) = \bigvee\{J \in \mathcal{J} \mid J \leq I\}.$$

This correspondence between openings and their sets invariants can also be applied to order relations between openings. The following result is also proved in [7]:

Proposition 3. *Let α and α' be two openings on $\mathcal{I}$. Then the following four statements are equivalent:*

(i) $\alpha \leq \alpha'$.

(ii) $\alpha \circ \alpha' = \alpha$.

(iii) $\alpha' \circ \alpha = \alpha$.

(iv) $Inv(\alpha) \subseteq Inv(\alpha')$.

In the same way as the order relation $\leq$ can be extended from $\mathcal{I}$ to the set of operators $\mathcal{I} \to \mathcal{I}$ (and in particular to openings, see Proposition 3), so can the operation $\bigvee$. For any set $\mathcal{U}$ of operators $\mathcal{I} \to \mathcal{I}$, $\bigvee \mathcal{U}$ is defined by setting

$$(\bigvee \mathcal{U})(I) = \bigvee \{\beta(I) \mid \beta \in \mathcal{U}\}$$

for any image $I \in \mathcal{I}$; in particular when $\mathcal{U} = \emptyset$ we get $\bigvee \mathcal{U} = \underline{\mathbf{O}_\mathcal{I}}$, where $\underline{\mathbf{O}_\mathcal{I}}(I) = \mathbf{O}_\mathcal{I}$ for every $I \in \mathcal{I}$. The following result can then be proved:

Proposition 4. *The set of openings on $\mathcal{I}$ is sup-closed. Given a set $\mathcal{U}$ of openings, $Inv(\bigvee \mathcal{U})$ is the smallest sup-closed subset of $\mathcal{I}$ containing $Inv(\alpha)$ for every $\alpha \in \mathcal{U}$. The least opening is $\underline{\mathbf{O}_\mathcal{I}}$ (which maps every image on $\mathbf{O}_\mathcal{I}$), and the greatest opening is the identity* **id**.

For example if an opening preserves all horizontal peak features in an image, and another one preserves all vertical peak features in that image, then their supremum preserves both horizontal and vertical peak features.

Let us comment these results in the case of openings applying compositions of local min and max functions to grey-levels. For such an opening α, there is a set $\mathcal{B}_\alpha$ of subsets of the space E, such that for any $B \in \mathcal{B}_\alpha$ and any grey-level $g \in G$, the image $B(g)$ defined by setting for any $p \in E$:

$$B(g)(p) = \begin{cases} g & \text{if } p \in B, \\ \mathbf{O}_G & \text{if } p \notin B, \end{cases} \tag{15}$$

is an invariant of the opening α, and every element of $Inv(\alpha)$ is the supremum of a set of images having that form (15). For example, in the min-max filter given by (1), $\mathcal{B}_\alpha = \mathcal{T}$, while in the rank-max filter given by (8), $\mathcal{B}_\alpha$ contains all sets of the form $T - X$, where $T \in \mathcal{T}$, $X \subset T$, and $|X| < k$.

The set $\mathcal{B}_\alpha$ is found by restricting the set G of grey-levels to the binary set $\{0, 1\}$, and by taking the set of all $B \subseteq E$ such that $B(1)$ is a minimal non-zero invariant of that opening α. In the non-binary case, the images $B(g)$ are the basic 'building blocks' of $Inv(\alpha)$. As the invariants $B(g)$ are composed of a flat portion on B superimposed on $\mathbf{O}_\mathcal{I}$, such an opening α applying compositions of min and max functions to grey-levels will be called *flat*.

Point *(iv)* of Proposition 3 means that for any $B \in \mathcal{B}_\alpha$, B is the union of some elements of $\mathcal{B}_{\alpha'}$. In the next section we will give several conditions for having $\alpha \leq \alpha'$ in the particular case of the rank-max filter (9) with rectangular windows.

As the supremum of a finite number of compositions of min and max functions is still a composition of min and max functions, Proposition 4 implies that the supremum of a set $\mathcal{F}$

of flat openings on $\mathcal{I}$ is again flat. (This holds because such a set $\mathcal{F}$ is always finite, due to the finiteness of E; if we remove the assumption of finiteness on E, we must then explicitly require that $\mathcal{F}$ is finite). Note also that both $\underline{\mathbf{O}_{\mathcal{I}}}$ and **id** are flat (the former in the extreme sense that $\mathbf{O}_G$ is obtained by applying the function max to an empty set of variables).

An interesting property of flat openings is given below without proof. Write $\underline{0}$ for the image having grey-level 0 everywhere (this is not necessarily the same as $\mathbf{O}_{\mathcal{I}}$). Then:

Proposition 5. *For any image $I \in \mathcal{I}$ and any flat opening α on $\mathcal{I}$,*

$$\alpha\big(I - \alpha(I)\big) = \underline{0}.$$

Note that this result is not true for any non-flat opening. Its meaning is the following. The flat opening α removes from an image I all 'narrow' positive features. Then $I - \alpha(I)$ represents all these 'narrow' positive features from I. Applying α to these isolated 'narrow' positive features, nothing should remain.

I.3. Technical aspects of the rank-max filter with rectangular windows

As we will only implement the particular filters given by (9) with all windows $W(p)$ being translates of a fixed rectangular mask, it is interesting to give their own properties.

Let us first particularize the mathematical results given in the preceding section (this especially important for the reader who would have skipped that section). Given a rank-max filter α and the identity operator **id**, Propositions 1 and 5 can be expressed as follows:

(*i*) $\alpha \leq \mathbf{id}$.

(*ii*) $\alpha \circ \alpha = \alpha$.

(*iii*) α is increasing.

(*iv*) $\alpha \circ (\mathbf{id} - \alpha) = 0_{\mathcal{I}}$, where $0_{\mathcal{I}}$ is the operator mapping every image I onto the zero image $\underline{0}$ having grey-level 0 everywhere.

As said before, (*i*) means that α removes positive features, (*ii*) that it does it in one pass, and (*iii*) that if α removes some positive feature, then it removes any smaller positive feature. In other words α removes in one pass from an image all positive features which are small enough. Finally $\mathbf{id} - \alpha$ extracts from an image the features which are removed by α, and so (*iv*) means that applying α to these extracted features removes everything.

We recall that given n images $I_1, \ldots, I_n$, their maximum $I_1 \vee \cdots \vee I_n$ and minimum $I_1 \wedge \cdots \wedge I_n$ are defined by setting

$$[I_1 \vee \cdots \vee I_n](p) = \max\{I_1(p), \ldots, I_n(p)\}$$
$$\text{and} \qquad [I_1 \wedge \cdots \wedge I_n](p) = \min\{I_1(p), \ldots, I_n(p)\}$$

for every $p \in E$, and these two operations generalize to operators $\mathcal{I} \to \mathcal{I}$ (where $\mathcal{I}$ is the set of images $I : E \to G$), by setting for any n operators $\beta_1, \ldots, \beta_n$:

$$[\beta_1 \vee \cdots \vee \beta_n](I) = \beta_1(I) \vee \cdots \vee \beta_n(I)$$
$$\text{and} \qquad [\beta_1 \wedge \cdots \wedge \beta_n](I) = \beta_1(I) \wedge \cdots \wedge \beta_n(I)$$

for every $I \in \mathcal{I}$.

By Proposition 4 (see also the discussion on 'flat openings'), if one takes the maximum $\beta = \alpha_1 \vee \cdots \vee \alpha_n$ of rank-max filters $\alpha_1, \ldots, \alpha_n$, then β still satisfies properties (i) to (iv).

The properties given above hold not only for rank-max filters, but also for all 'flat openings' defined in the previous section. Let us now concentrate on the particular properties of rank-max filters. We write $RM[k, W]$ for the filter

$$\mathbf{id} \wedge [Max_{W^*} \circ R^k_W]$$

given in (9) for a rank k, a set of windows $W(p)$ associated to points $p \in E$, and a set of dual windows $W^*(p)$ defined in (2) by

$$q \in W^*(p) \quad \text{iff} \quad p \in W(q).$$

An image I is invariant under $RM[k, W]$ iff for every $p \in E$ there exists some $q \in E$ with $p \in W(q)$ (in other words $q \in W^*(p)$), such that less than k points $r \in W(q)$ have grey-level $I(r) < I(p)$. We recall that the set of invariants of $RM[k, W]$ is written $Inv(RM[k, W])$. This set is generated by taking maxima of sets of images of the form $K[g]$, where $g \in G$, $K \subset W(q)$ for some $q \in E$, and $|W(q) - K| = k - 1$; these images $K[g]$ are defined by setting for every $p \in E$:

$$K[g](p) = \begin{cases} g & \text{if } p \in K[g], \\ \mathbf{O}_G & \text{if } p \notin K[g], \end{cases}$$

where $\mathbf{O}_G$ is the least grey-level in G.

Given $\alpha = RM[k, W]$ and $\alpha' = RM[k', W']$, one can compare them. If $\alpha \leq \alpha'$, this means that α removes from an image I more than α' does. We say then that α is *stronger*, or *more active*, than α'. By Proposition 3, we know that $\alpha \leq \alpha'$ iff $\alpha \circ \alpha' = \alpha$, iff $\alpha' \circ \alpha = \alpha$, iff $Inv(\alpha) \subseteq Inv(\alpha')$. Thus when one applies in succession two rank-max filters such that one is stronger than the other, this amounts to simply applying the stronger filter.

When can we say that α is stronger than α'? If $k < k'$, it is clear that $R^k_W \leq R^{k'}_W$, and so by (9) $RM[k, W] \leq RM[k', W]$. There is also a circumstance where we can say $RM[k, W] \leq RM[k, W']$ for two different set of windows. Let us say that the windows $W'(q)$ *cover* the windows $W(p)$ if for every $p \in E$ the window $W(p)$ is the union of a certain number of windows $W'(q)$. Suppose indeed that the windows $W'(q)$ cover the windows $W(p)$. Given an image I invariant under $RM[k, W]$, for any point $p \in E$ there is some window $W(q)$ containing p such that less than k points $r \in W(q)$ have $I(r) < I(p)$; as the $W'(q')$ cover $W(q)$, there is some $q' \in E$ such that $p \in W(q') \subseteq W(q)$, and so less than k points $r \in W'(q)$ have $I(r) < I(p)$; this means that I is invariant under $RM[k, W']$. Thus $Inv(RM[k, W]) \subseteq Inv(RM[k, W'])$, and so by Proposition 3 $RM[k, W]$ is stronger than $RM[k, W']$. We sum up:

— *If $k < k'$, then $RM[k, W] \leq RM[k', W]$.*

— *If the windows $W'(q)$ cover the windows $W(p)$, then $RM[k, W] \leq RM[k, W']$.*

In practice we will suppose that the space E is a subset of the set $\mathbf{Z}^n$ of n-tuples of integers (in other words of the set points in n-dimensional space having integer coordinates). The points of $\mathbf{Z}^n$ can be considered as vectors, and so they can be added or subtracted. Given a subset X of $\mathbf{Z}^n$, we will write $-X$ for the set of $-x$ for $x \in X$, and for any point $p \in \mathbf{Z}^n$

we will write $p+X$ for the set of points $p+x$ for $x \in X$, and $p-X$ for $p+(-X)$. We assume that the windows $W(p)$ are translates of a fixed template B, in other words $W(p) = p+B$ for every $p \in E$ (see (6)). Then it is easy to see that the dual window $W^*(p)$ is given by $W^*(p) = p - B$ (see (7)).

It should be noted that with this choice of windows the filter $RM[k,W]$ is independent of the way the window is set about a point p, in other words it does not change if we replace B by a translate $h+B$ of it. Indeed, if we choose the translated windows $W_h(p)$ by setting $W_h(p) = p + (h+B)$, then $W_h^*(p) = p - (h+B)$; it follows that the windows $W_h(q)$ for for $q \in W_h^*(p)$ are given by $q+h+B$ for $q \in p-h-B$, in other words by $(p-h-b)+h+B = p-b+B$ for $b \in B$. The set of such windows is independent of h, and so $Max_{W_h^*} \circ R_{W_h}^k$, which assigns to p the grey-level equal to the maximum, among such windows, of the k-th rank function of grey-levels inside a window, is independent of h. It follows then by (9) that $RM[k,W_h] = RM[k,W]$. For example, in our implementation of the rank-max filters in the case where B is a $h \times w$ rectangle, we will set B having 0 at its bottom right corner; then for any $p \in E$, $W(p)$ will be the $h \times w$ rectangle having p as bottom right corner, and $W^*(p)$ the one having p as top left corner.

Given the windows $W(p) = p+B$ and the windows $W'(p) = p+B'$, then the windows $W'(q)$ cover the windows $W'(p)$ iff B is a union of translates of B'. For example if B is a $h \times w$ rectangle and B' a $h' \times w'$ rectangle, this happens if $h' \leq h$ and $w' \leq w$. Write $RM[k, h \times w]$ for $RM[k,W]$ when $W(p) = p+B$ for a $h \times w$ rectangle B; then we get the following:

— *If* $k \leq k'$, $h \geq h'$, *and* $w \geq w'$, *then* $RM[k, h \times w] \leq RM[k', h' \times w']$.

We have outlined the implications of the results of the preceding section for the rank-max filter, especially for particular choices of the windows $W(p)$. Let us now discuss some practical issues concerning these filters.

One problem is that for implementation purposes we have assumed that the space E is *finite*. If the windows $W(p)$ are translates of a fixed template B, then for certain points p relatively close to the border of E the window $W(p) = p+B$ will not be completely contained in E, or p will belong to the window $W(q)$ associated to a point $q \notin E$ (in other words, $W^*(p) = p - B$ will not be completely contained in E). How then does one define the behaviour of the filter $RM[k,W]$ on such points?

A possible solution is to restrict each window $W(p)$ and $W^*(p)$ to its intersection with E. In other words, for every $p \in E$, we set $W(p) = (p+B) \cap E$ and $W^*(p) = (p-B) \cap E$. Then we still have the relation $q \in W^*(p)$ iff $p \in W(q)$ given in (2). In the case where the size of $W(p)$ is less than the rank k, we select (for the filter R_W^k) instead of the "k-th lowest grey-level in $W(p)$" the largest grey-level in it. In other words in each window $(p+B) \cap E$ we select the r_p-th lowest grey-level, where $r_p = min(k, |(p+B) \cap E|)$.

We have not adopted this solution in our implementation. Let us describe the one we have chosen. The basic idea is to extend the space E by a frame E' such that for every point $p \in E$ the window $p+B$ is included in $E \cup E'$, and to fill the image by assigning arbitrary grey-levels to points in E'. This is often done in practical implementations of filters. One often assigns to points in E' the grey-levels of their closest neighbors in E. However we have prefered to assign to them a constant grey-level $grl_{E'}$, which is either above every grey-level in I (say $\mathbf{I}_G$ or $+\infty$), or below every grey-level in I (say $\mathbf{O}_G$ or $-\infty$). These two possible choices for $grl_{E'}$ give the two variants (respectively 'plus' and 'minus') of our implementation. In the 'plus' variant, a narrow peak along the border of E will be part of a wider peak in $E \cup E'$, and so it will be preserved. In the 'minus' variant, that peak will not be part of a wider peak in $E \cup E'$, and it will be erased. Thus the 'plus' variant is to be prefered when one wants to have the least change along the borders of the image.

In order to justify our solution we will show two things:

First, provided that we make (and for the 'minus' variant only) the very reasonable assumption that for every $p \in E$ there is some $q \in E$ such that $p \in q + B \subseteq E$ (for example if both E and B are rectangular and B is smaller than E), the resulting image does not depend on the exact value of $grl_{E'}$ in each variant, only on the fact that for every $p \in E$ we have $grl_{E'} \geq I(p)$ in the 'plus' variant and $grl_{E'} \leq I(p)$ in the 'minus' variant. In particular, the grey-levels of points in the filtered image I' will always be within the set of grey-levels of points in the original image I.

Second, despite the addition of the frame E' having constant grey-level $grl_{E'}$, the filter can still be expressed in the form (9), provided that we define in a proper way the windows $W(p)$ (and $W^*(p)$) for points $p \in E'$. This is important, because it garantees that the properties of the filter given above will remain valid with the modification made here.

For any point q in $E \cup E'$, write f_q for $\text{rank}^k{}_{r \in W(q)} I(r)$. Thus for every $p \in E$ we have

$$I'(p) = \min\{I(p), \max_{q \in W^*(p)} f_q\} = \max_{q \in W^*(p)} \min\{I(p), f_q\}. \tag{16}$$

Let us consider first the 'plus' variant. We assume that $grl_{E'} \geq \max_{p \in E} I(p)$. For every point $p \in E$, we have two cases:

(a) For every $q \in W^*(p)$, there are at least k points r of $W(q) \cap E$ with grey-level $I(r) \leq I(p)$. Then whatever the choice of $grl_{E'}$, for every $q \in W^*(p)$ we have $f_q = \text{rank}^k{}_{r \in W(q) \cap E} I(r)$, which is independent of $grl_{E'}$. Thus by (16) the value of $I'(p)$ is independent of $grl_{E'}$.

(b) There exists some $q \in W^*(p)$ such that less than k points r of $(q + B) \cap E$ have grey-level $I(r) \leq I(p)$. Then whatever the choice of $grl_{E'}$, $f_q > I(p)$ and so by (16) we have $I'(p) = I(p)$, which is independent of $grl_{E'}$.

Now we can choose E' and the windows $W(p)$ and $W^*(p)$ on it in such a way that for every $p \in E'$, there is some $q \in E'$ such that $p \in W(q) \subseteq E'$. Then $\text{rank}^k{}_{r \in W(q)} I(r) = grl_{E'} = I(p)$, and so by applying the filter as expressed in (9) on the image I extended to $E \cup E'$, we will get $I'(p) = grl_{E'} = I(p)$ for $p \in E'$, and so our construction is coherent with formula (9).

Let us consider next the 'minus' variant. We assume that $grl_{E'} \leq \min_{p \in E} I(p)$. Take any point $p \in E$. For any $q \in W^*(p)$ we have two possibilities:

(a) $|W(q) \cap E'| = i_q \geq k$. Then whatever the choice of $grl_{E'}$, we have $f_q = grl_{E'}$.

(b) $|W(q) \cap E'| = i_q < k$. Then we have $f_q = \text{rank}^{k-i_q}{}_{r \in W(q) \cap E} I(r)$, whatever the choice of $grl_{E'}$.

But we said above that we assume that there exists some $q \in E$ such that $p \in W(q) \subseteq E$, and we have then $q \in W^*(p)$ and $|W(q) \cap E'| = 0 < k$. Thus the set X_p of points of

$W^*(p)$ satisfying (b) is not empty, and by (16) we have

$$\begin{aligned} I'(p) &= \min\Big\{I(p), \max\{grl_{E'}, \max_{q\in X_p} f_q\}\Big\} = \min\{I(p), \max_{q\in X_p} f_q\} \\ &= \min\{I(p), \max_{q\in X_p} \operatorname{rank}^{k-i_q}_{r\in W(q)\cap E} I(r)\}, \end{aligned}$$

which is independent of $grl_{E'}$.

Now for any choice of the windows $W(p)$ and $W^*(p)$ on E', by applying the filter as expressed in (9) on the image I extended to $E \cup E'$, for $p \in E'$, $I'(p)$ is the minimum of $I(p) = grl_{E'}$ and an expression which is always $\geq grl_{E'}$, in other words we will get $I'(p) = I(p)$. Thus our construction is coherent with formula (9).

We have thus shown that in both variants, our solution of extending the space E leads to a filter which is still coherent with formula (9) (and in particular satisfies all its properties), and that the actual value of the grey-level $grl_{E'}$ on the frame E' extending E does not influence the behaviour of the filter, provided that:

(i) this value satisfies the constraint inherent to each variant ($grl_{E'} \geq \max_{p\in E} I(p)$ in the 'plus' variant, and $grl_{E'} \leq \min_{p\in E} I(p)$ in the 'minus' variant), and

(ii) in the 'minus' variant the space E is equal to a union of windows $W(q)$ for some points $q \in E$ (a condition which is naturally satisfied when each $W(p) = p + B$, where B and E are rectangular and B is smaller than E).

We said above that the 'plus' variant is the most suitable one when one wants to minimize the changes along the border of the image. Therefore we have used it in our application of the filter $RM[k, h \times w]$ to angiographic images. The behaviour of this filter on the ridges formed by blood vessels will be described in Chapter III, where we will see that it has several advantages over the min-max filter of Nakagawa and Rosenfeld.

We have described the main properties of the filter $RM[k, W]$, in particular for rectangular windows. We have explained how to apply it to finite images without problems at the border. After these clarifications, we are now ready for the description of the PASCAL implementation.

II. Implementation in PASCAL

We will describe the computer implementation of the rank-max filter $RM[k, W]$, where each window $W(p)$ is the translate $p + B$ of a rectangular template B of width w and height h. The input image must have a fixed width, but can be of any height. The three parameters k, w, and h, as well as the type of grey-level on the border surrounding the image (maximum or minimum grey-level) are variables which can be chosen by the user. We first present the algorithm in a general way, and then give the PASCAL code preceded by a brief explanation.

II.1. Description of the algorithm

We suppose that we have an image I on a rectangular grid E of width N, on which we want to perform the filter $RM[k, W]$ with windows $W(p)$ being $h \times w$ rectangles. For every point p, we must consider all windows containing it, and in order to avoid problems along the border, we must extend the grid E by a surrounding frame E' containing $h - 1$ rows above and below E and $w - 1$ columns to the left and right of it. The image I is extended on E' by assigning to points of E' a constant grey-level, the maximum one or the minimum one.

For every point p, $W(p) = p + B$ and $W^*(p) = p - B$ for a $h \times w$ rectangle B. As explained before, it does not matter where B is located w.r.t. the origin. Thus we will set B having 0 at its bottom right corner; then for any $p \in E$, $W(p)$ will be the $h \times w$ rectangle having p as bottom right corner, and $W^*(p)$ the one having p as top left corner. The windows containing p are all $W(q)$ for $q \in W^*(p)$.

The filter $RM[k, h \times w]$ performs the following operation on I: for every point $p \in E$, one computes for every $q \in W^*(p)$ the k-th rank function on the grey-levels of points of $W(q)$ (whose result will be written f_q), then one takes the maximum $m(p)$ of all f_q for $q \in W^*(p)$, and the minimum of $I(p)$ and $m(p)$ gives the grey-level $I'(p)$ in the filtered image I'.

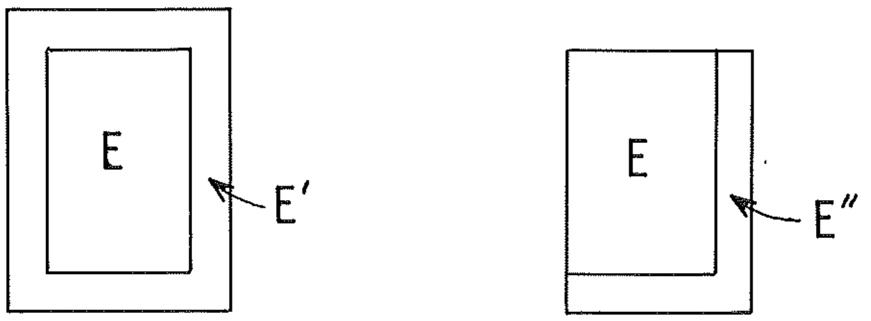

Figure 4. *E, E', and E''.*

Let E'' be the portion of the frame E' remaining after the deletion of its top and left portions (see Figure 4). With our choice of windows, the set of all q for which f_q is computed, namely the union of $W^*(p)$ for $p \in E$, covers $E \cup E''$. This leads to the following description of the behavior of the filter. First for every $h \times w$ rectangle inside $E \cup E'$ one computes the k-th rank of the grey-levels of points inside it, and this value is associated to the bottom right corner q of that rectangle and denoted f_q; thus f_q is defined for every $q \in E \cup E''$. Then for every $h \times w$ rectangle inside $E \cup E''$ one computes the maximum of the values f_q of points q inside it, and this value is associated to the top left corner p of that rectangle and denoted $m(p)$; thus $m(p)$ is defined for every $p \in E$, and the minimum of $I(p)$ and $m(p)$ gives the grey-level $I'(p)$.

As described above, the filter $RM[k, h \times w]$ can be implemented with a memory buffer consisting of two intermediate images between the original one I and the filtered one I': first the extension of image I to $E \cup E'$, then the image of all values f_q for $q \in E \cup E''$. This buffer and the operations leading from the original image I to the filtered image I' are illustrated in Figure 5.

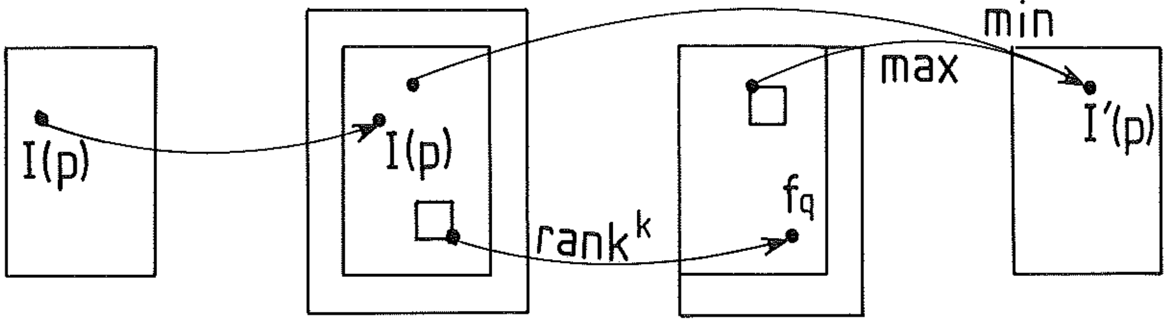

Figure 5. *Decomposition of the filter into stages.*

However in a sequential implementation of the filter we do not need at every time this whole buffer, but only a horizontal slice of it, high enough to contain a row of windows $W(q)$, which will be displaced sequentially from top to bottom. It will be an array of height h and of width equal to that of $E \cup E'$ plus that of $E \cup E''$, in other words $2N + 3(w - 1)$, where N is the width of the grid E. This has two advantages: a reduced memory, and the possibility to apply the filter to images of any height. We will call this slice the *window array*. The process of reading the input image, moving the window array, computing of the values f_q and $m(p)$ over a row, and writing the output image, is performed sequentially row by row.

The window array (of height h and width $2N + 3(w - 1)$) can be laterally divided into two parts (see Figure 6). The *left part* covers the extension of the input image I to $E \cup E'$, has width $N + 2(w - 1)$, and can be subdivided into a *left margin* of width $w - 1$, an *input image portion* of width N, and a *right margin* of width $w - 1$. The *right part* covers the intermediate image consisting of values f_q for points q inside $E \cup E''$, has width $N + w - 1$, and can be subdivided into an *main portion* of width N and a *right margin* of width $w - 1$.

The following two facts concerning the two parts of the window array follow from Figure 6. First, to any point of the right part corresponds a unique point of the left part, namely the point on the same row, but located $w - 1$ points further from its respective left border. In particular the points of the left margin of the left part do not correspond to points of the right part. Second, at the beginning of the scanning of the input image (when

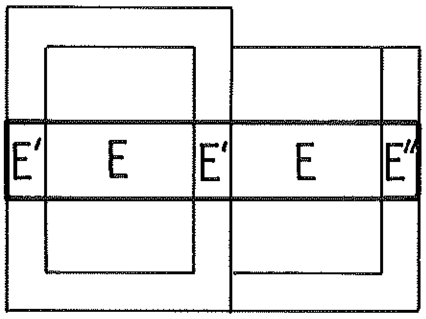

Figure 6. *Decomposition of the window array.*

we read its first row), the $h-1$ top rows of the right part are idle; it is only when the window array is completely below the top part of frame E' (in other words when at least h rows of the input image I have been read) that the right part of that array can be completely filled with values f_q.

The filter $RM[k, h\times w]$ is implemented by the following algorithm. One takes a window array of height h and width $2N+3(w-1)$ which is initially filled with the constant grey level $grl_{E'}$ associated to points of E'. We repeat the following sequence of operations, which correspond to a vertical scan of image I, the corresponding calculation of rank and max functions, and the writing of the resulting lines of the filtered image I'.

First, we must shift the window array downwards by one unit on the buffer; this corresponds to shifting all rows of that array by one unit upwards, and liberating the last row. In other words, for $i = 1$ to $h-1$ the i-th row is replaced by the $i+1$-th row, and the h-th row is free. In fact, in our PASCAL implementation, we do not interchange the contents of these rows, but we have a labels array associating to $i = 1, \ldots, h$ the physical position of the row in the window array corresponding to the i-th row in the algorithm; then we have only to interchange labels and find a free row whose position will be a new label associated to h, in other words to the last row of the window array.

Second, in the left part of the window array, the last row must correspond to a new row of the input image I extended to $E \cup E'$. If I is not completely read, we get a new line of it, and write it on the corresponding portion of the last row, namely the input image portion of the left part. If I is completely read, we must read a row of the bottom part of E', that is a row with constant grey-level $grl_{E'}$. In our PASCAL implementation, we do not copy anything in this case, but we associate with h the label 0, which corresponds to a supplementary row added to the window array, whose left part is already set to the constant grey-level $grl_{E'}$.

Clearly this process of shifting the rows of the window array and reading the input image is performed provided that beforehand we have not reached the end of I (if $h = 1$) or less than $h-1$ rows of the bottom part of E' have been read (if $h > 1$). Otherwise we should already have stopped.

Third, the left part of the window array contains $N+w-1$ rectangular $h\times w$ windows. For each one of them we compute the k-th rank of its grey-levels, and this value is given to the point in the right part of the window array corresponding to the bottom right corner of that window. In other words, the result corresponding to the i-th rectangular window in the left part is given as grey-level to the i-th point on the last row of the right part.

If we have not yet read at least h rows of the input image I, then there is still a portion of the right part which has not been filled. We have then nothing more to compute at this stage of the scan. On the other hand, if we have read at least h rows of the input image I, then the whole right part has been filled, and there is a fourth task at hand. The right part of the window array contains N rectangular $h \times w$ windows. For each one of them we compute the maximum of its grey-levels, and this value is to be compared to the grey-level of the point in the left part of the window array corresponding to the top left corner of that window. In other words, the result corresponding to the i-th rectangular window in the right part is compared to the grey-level of the $i + w - 1$-th point on the first row of the left part. We take the minimum of these two grey-levels, and the row of N values obtained in this way becomes a new row of grey-levels in the filtered image I'.

This sequence of three or four operations is continued, as we said above, as long as at the beginning of it we have not reached the end of the input image I or have not read $w - 1$ rows of E' after the end of I. The second, third, and fourth operations are illustrated in Figure 7.

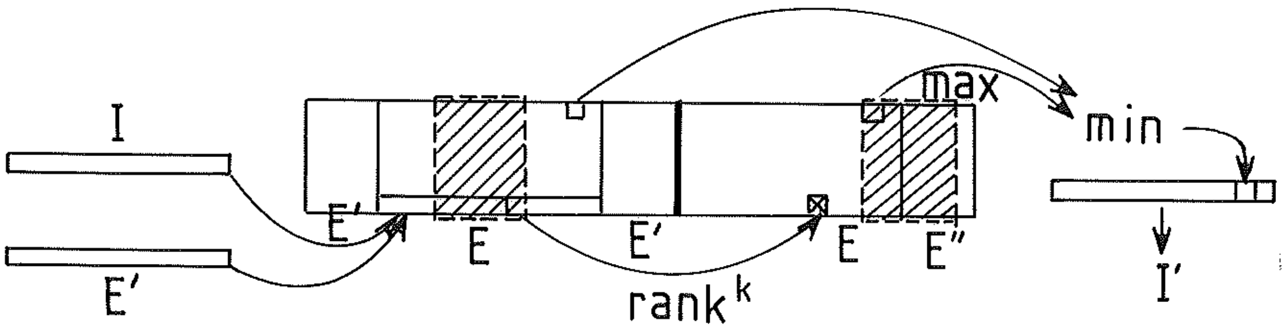

Figure 7. *Computations in the window array.*

In order to know whether we have already read h rows of the input image I (for deciding if we have to apply the fourth operation) or $h - 1$ rows of E' after the end of I (for deciding if the algorithm must stop), we take two counters *up* and *down* whose values are updated at each step.

The computation of the k-th rank (in the third operation) or of the maximum (in the fourth one) of grey-levels in $h \times w$ windows is done by the algorithm of Huang et al. [3]. Let k be the required rank. For any window X and grey-level g, write $lower_X(g)$ and $equal_X(g)$ for the number of occurences in X of grey-levels respectively smaller than g and equal to g. We build an histogram of grey-levels in the first window X_0. By scanning that histogram for g increasing from the smallest grey-level to the largest one, we can iteratively compute $lower_{X_0}(g)$ as the sum of $equal_{X_0}(g')$ for $g' < g$, and we do so until we have $lower_{X_0}(g) < k \leq lower_{X_0}(g) + equal_{X_0}(g)$. This gives the required k-th rank grey-level g_0 in X_0. The k-th rank in the other windows is built iteratively by the following modification of the histogram. Suppose that we have the histogram of grey-levels in the i-th window X_i and the resulting grey-level g_i for k-th rank grey-level; we know also the value of $lower_{X_i}(g_i)$. The next window X_{i+1} is obtained from X_i by deleting its leftmost column and by adding a new rightmost column. For every grey-level g in the removed leftmost column, we must decrease by 1 the value of $equal_{X_i}(g)$, and also the value of $lower_{X_i}(g_i)$ if $g < g_i$; for every grey-level g' in the added rightmost column, we must increase by 1 the value of $equal_{X_i}(g')$, and also the value of $lower_{X_i}(g_i)$ if $g' < g_i$. This gives then the histogram for X_{i+1}, and the value of $lower_{X_{i+1}}(g_i)$. If $lower_{X_{i+1}}(g_i) < k \leq lower_{X_{i+1}}(g_i) + equal_{X_{i+1}}(g_i)$, then the resulting grey-level g_{i+1} is g_i; otherwise we move up or down in the histogram until we obtain $lower_{X_{i+1}}(g) < k \leq lower_{X_{i+1}}(g) + equal_{X_{i+1}}(g)$, which gives then the value for g_{i+1}.

II.2. The PASCAL program

Columns are numbered starting from 0, and so for an image $N = 256$ pixels wide, they are numbered from 0 to `maxcol=255`. We choose a window array of length 640, leading to the constant `arrwm=639`. The rectangular windows have their width and height bounded by `maxwwd=32` and `maxwht=32` respectively, with the further constraint that their product cannot exceed `maxwsz=256`. Grey-levels range between `mingrl=0` and `maxgrl=255`.

The actual width and height of the windows (chosen by the user) are given by the variables `wwd` and `wht`, and their product by `wsz`. We define also `wwdm` as `wwd-1`, and the width of the left part of the window array is represented by `arrhalf`, which is equal to `maxcol+wwd+wwdm`.

The two variables `up` and `down` play a central role in the execution of the program. Their value is initialized to 0. Afterwards `up` gives the number of lines of the input image which have already been read, but when this number exceeds `wht`, `up` remains equal to `wht`. The computation of window maxima and the writing of rows of the output image is done only when we have `up=wht`. On the other hand `down` gives the number of lines of the frame which have been read after the end of the input image. The program stops with the input image file at end and `down=wht-1`.

The grey-level of the frame is given by `bordergrl`, which is set to `mingrl` in the 'minus' variant, and to `maxgrl` in the 'plus' variant.

The window array is represented by the variable `windarr`, which is a `packed array [0..maxwht,0..arrwm] of mingrl..maxgrl`. Note that the rows of the window array are numbered starting from 1, contrarily to columns. In fact the rows of `windarr` do not correspond directly to the rows of the window array in the algorithm; to the i-th row of the abstract window array ($i = 1, \ldots$) corresponds the row indexed `no[i]` of `windarr`, where `no` is an `array [1..maxwht] of 0..maxwht`. The row of `windarr` indexed 0 corresponds to a row entirely contained in the frame, and so all entries of its left part keep the value `bordergrl` throughout the execution of the program. The rows indexed 1 to `wht` correspond to rows of the input image, and the input image portion of their left part is filled with values from the input image. In other words `no[i]=0` when the i-th row is in the frame, otherwise `no[i]` lies in the range `1..wht`.

The variable `new` is the index of the last row of the window array, in other words `new=no[wht]`; we use also `first=no[1]`.

The rows of the input and output images are accessed by the variable imrow, which is a packed array [0..maxcol] of mingrl..maxgrl. These images are sequential files (called infile and outfile respectively) whose elements are such rows.

The chosen rank for the filter is given by rnk. Inside each window we compute the grey-level rnkgrl having the required rank (rnk in the left part of the window array, and wsz in the right part). As said above, this computation is done by the algorithm of Huang et al., which builds a grey-level histogram of the leftmost window and updates it along the row. This histogram is represented by the variable histo, which is a packed array [mingrl..maxgrl] of 0..maxwsz. The number of grey-levels in the window which are smaller than rnkgrl is given by lower.

The other variables used in the program (see the VAR declaration in the code) are auxiliary.

Let us now describe what the program and its procedures do. The reader should refer to the code at the end of this section. There are five minor procedures: choiceofwsize, choiceofrnk, choiceofbordergrl, makehisto, and movehisto. The first three interactively allow the user to choose the values of wwd, wht, rnk, and bordergrl. The code of these three procedures is self-explanatory and needs no further comment.

The last two minor procedures makehisto and movehisto are used to compute in a given window, by the above-mentioned algorithm of Huang et al., the grey-level rnkgrl having a certain rank myrnk, and the number lower of grey-levels smaller than rnkgrl. (NB: The variable myrnk will be rnk and wsz respectively on the left and right parts of the window array.) The procedure makehisto is used for the leftmost window. It makes the grey-level histogram histo of the window by scanning it, and initializes rnkgrl to mingrl, with lower set to 0; as long as rnkgrl is too small for the rank myrnk (that is, lower+histo[rnkgrl]<myrnk), it is increased and lower is modified accordingly. The procedure movehisto is used for following windows. The leftmost column of the previous window is removed, and the right column of the new window is added, leading to corresponding modifications of the histogram histo and the number lower; if the value of rnkgrl is too large for the rank myrank (that is, lower>=myrnk), it is decreased and lower is modified accordingly, until the right value is reached; otherwise we eventually increase rnkgrl as we did in makehisto.

The four major procedures are init, inwindow, rankwindow, and maxout. The first one, init, initializes the values of all variables. In particular it calls the procedures choiceofwsize, choiceofrnk, and choiceofbordergrl. Note that all entries of the indexing array no are set to 0 (corresponding to rows entirely contained in the frame), since we have not yet read any row of the input image, and that the left part of the array windarr is filled with the greylevel bordergrl. Note also the VAX/VMS nonstandard PASCAL commands open used to allow the processing of file variables.

The procedure inwindow corresponds to the operation of shifting the window array downwards by one unit, and reading one new row of the input image or the frame. We must in particular update the values of up and down, and find a value for the index new of the row of windarr corresponding to the new row. If up<wht, then only up rows of the input image have been read, and they are put in the rows 1 to up of windarr; in this case we increase up by one, and the new value of up is the index of an iddle row of windarr; we

choose this value for new. If we are at the end of the input file, then we increase down by one, and the next row will be completely inside the frame; we set thus new to 0. If it is not the case, then we read a new row of the input file, and copy it inside the left part of the row of windarr indexed new. Finally we update the indexing array no.

The procedure rankwindow finds (using makehisto and movehisto) the grey-level having rank rnk inside each window in the left part of windarr, and copies it in the corresponding place on the right part of the row of windarr indexed new.

The procedure maxout finds (using again makehisto and movehisto) the maximum grey-level inside each window in the right part of windarr, compares it with the grey-level of the corresponding point in the left part of the row of windarr indexed no[1], and copies their minimum in a new row of the output file.

Having described all procedures, we come now to the main body of the program rankmax. We first call init. Afterwards, as long as we have not finished, *i.e.*, either we have not reached the end of the input file (for wht=1), or down<wht-1 (for wht>1), we call inwindow, rankwindow, and if up=wht also maxout.

We have also written a program difcont, which subtracts two images, and linearly enhaces the contrast of the difference image in order to fill the whole grey-level range. It will be used to show the narrow peak features in an image which are removed by the rank-max filter.

We reproduce the code of the two PASCAL programs starting on next page.

```
PROGRAM rankmax(input,output,infile,outfile);
{logical names 'inf' 'outf' for 'infile' 'outfile'}

CONST
maxcol = 255;
arrwm  = 639;
maxwwd = 32;
maxwht = 32;
maxwsz = 256;
mingrl = 0;
maxgrl = 255;

TYPE
wwdrng = 0..maxwwd;
whtrng = 0..maxwht;
wszrng = 0..maxwsz;
arrno  = array [1..maxwht] of whtrng;
grl    = mingrl..maxgrl;
colno  = 0..maxcol;
row    = packed array [colno] of grl;
arwrng = 0..arrwm;
whtarr = packed array [whtrng,arwrng] of grl;
hstgrm = packed array [grl] of wszrng;

VAR
wwd, wwdm, jj : wwdrng;
wht, up, down, new, ii, uu, first : whtrng;
wsz, rnk, lower : wszrng;
no : arrno;
bordergrl, thisgrl, rnkgrl : grl;
imrow : row;
infile, outfile : file of row;
j, arrhalf : arwrng;
windarr : whtarr;
histo : hstgrm;
```

```
PROCEDURE choiceofwsize;
VAR
width, height : integer;
BEGIN
REPEAT
  writeln('Enter window dimensions, width (1..',
     maxwwd:3,'), height (1..', maxwht:3, '),');
  writeln('with width * height not larger than', maxwsz:4, ':');
  writeln('width=');
  readln(width);
  writeln('height=');
  readln(height);
  UNTIL ((width>0) AND (width<=maxwwd) AND (height>0)
        AND (height<=maxwht) AND (width*height<=maxwsz));
wwd:= width;
wht:= height;
wsz:= wwd*wht;
END;

PROCEDURE choiceofrnk;
VAR
rank : integer;
BEGIN
REPEAT
  writeln('Enter rank (1=MIN ..',wsz:3,'):');
  readln(rank);
  UNTIL ((rank>0) AND (rank<=wsz));
rnk:= rank;
END;

PROCEDURE choiceofbordergrl;
VAR
answer : char;
BEGIN
REPEAT
  writeln('Enter choice of border environment (+,-):');
  readln(answer);
  UNTIL ((answer='+') OR (answer='-'));
IF answer='+' THEN bordergrl:= maxgrl ELSE bordergrl:= mingrl;
END;
```

```
PROCEDURE makehisto(myj: integer; myrnk: wszrng);
BEGIN
FOR thisgrl:= mingrl TO maxgrl DO histo[thisgrl]:= 0;
FOR ii:= 1 TO wht DO
  BEGIN
  uu:= no[ii];
  FOR jj:= 0 TO wwdm DO
    BEGIN
    thisgrl:= windarr[uu,myj+jj];
    histo[thisgrl]:= histo[thisgrl]+1;
    END;
  END;
lower:= 0;
rnkgrl:= mingrl;
WHILE lower+histo[rnkgrl]<myrnk DO
  BEGIN
  lower:= lower+histo[rnkgrl];
  rnkgrl:= rnkgrl+1;
  END;
END;

PROCEDURE movehisto(myj: integer; myrnk: wszrng);
BEGIN
FOR ii:= 1 TO wht DO
  BEGIN
  uu:= no[ii];
  thisgrl:= windarr[uu,myj-1];
  IF thisgrl<rnkgrl THEN lower:= lower-1;
  histo[thisgrl]:= histo[thisgrl]-1;
  thisgrl:= windarr[uu,myj+wwdm];
  IF thisgrl<rnkgrl THEN lower:= lower+1;
  histo[thisgrl]:= histo[thisgrl]+1;
  END;
IF lower>=myrnk
  THEN REPEAT
         rnkgrl:= rnkgrl-1;
         lower:= lower-histo[rnkgrl];
       UNTIL lower<myrnk
  ELSE WHILE lower+histo[rnkgrl]<myrnk DO
         BEGIN
         lower:= lower+histo[rnkgrl];
         rnkgrl:= rnkgrl+1;
         END;
END;
```

```
PROCEDURE init;
BEGIN
open(infile,'inf',old);
reset(infile);
open(outfile,'outf',new,,sequential,fixed);
rewrite(outfile);
up:= 0;
down:= 0;
choiceofwsize;
wwdm:= wwd-1;
arrhalf:= maxcol+wwd+wwdm;
choiceofrnk;
choiceofbordergrl;
FOR ii:= 1 TO maxwht DO no[ii]:= 0;
FOR ii:= 0 TO maxwht DO
  FOR j:= 0 TO arrwm DO windarr[ii,j]:= bordergrl;
END;

PROCEDURE inwindow;
BEGIN
IF up<wht
  THEN BEGIN
       up:= up+1;
       new:= up;
       END
  ELSE new:= no[1];
IF EOF(infile)
  THEN BEGIN
       down:= down+1;
       new:= 0;
       END
  ELSE BEGIN
       imrow:= infile^;
       get(infile);
       FOR j:= 0 TO maxcol DO windarr[new,wwdm+j]:= imrow[j];
       END;
FOR ii:= 1 TO wht-1 DO no[ii]:= no[ii+1];
no[wht]:= new;
END;
```

```
PROCEDURE rankwindow;
BEGIN
makehisto(0,rnk);
windarr[new,arrhalf]:= rnkgrl;
FOR j:= 1 TO maxcol+wwdm DO
  BEGIN
  movehisto(j,rnk);
  windarr[new,arrhalf+j]:= rnkgrl;
  END;
END;

PROCEDURE maxout;
BEGIN
first:= no[1];
makehisto(arrhalf,wsz);
thisgrl:= windarr[first,wwdm];
IF rnkgrl<thisgrl THEN imrow[0]:= rnkgrl
                  ELSE imrow[0]:= thisgrl;
FOR j:= 1 TO maxcol DO
  BEGIN
  movehisto(arrhalf+j,wsz);
  thisgrl:= windarr[first,wwdm+j];
  IF rnkgrl<thisgrl THEN imrow[j]:= rnkgrl
                    ELSE imrow[j]:= thisgrl;
  END;
outfile^:= imrow;
put(outfile);
END;

BEGIN
init;
WHILE (NOT EOF(infile)) OR (down<wht-1) DO
  BEGIN
  inwindow;
  rankwindow;
  IF up=wht THEN maxout;
  END;
END.
```

```
PROGRAM difcont(output,hifile,lofile,outfile);
{logical names 'hif' 'lof' 'outf' for 'hifile' 'lofile' 'outfile'}

CONST
maxcol = 255;
mingrl = 0;
maxgrl = 255;
grlrng = 255;

TYPE
colno = 0..maxcol;
grl   = mingrl..maxgrl;
row   = packed array [colno] of grl;

VAR
jj : colno;
mindif, maxdif, dif, difrng : integer;
hirow, lorow, outrow : row;
hifile, lofile, outfile : file of row;
```

```
BEGIN
open(hifile,'hif',old);
reset(hifile);
open(lofile,'lof',old);
reset(lofile);
open(outfile,'outf',new,,sequential,fixed);
rewrite(outfile);
maxdif:= mingrl;
mindif:= maxgrl;
WHILE NOT (EOF(hifile) OR EOF(lofile)) DO
  BEGIN
  hirow:= hifile^;
  get(hifile);
  lorow:= lofile^;
  get(lofile);
  FOR jj:= 0 TO maxcol DO
    BEGIN
    dif:= hirow[jj]-lorow[jj];
    IF dif< mindif THEN mindif:= dif;
    IF dif> maxdif THEN maxdif:= dif;
    END;
  END;
writeln('minimum difference=',mindif);
writeln('maximum difference=',maxdif);
IF maxdif>mindif THEN difrng:= maxdif-mindif ELSE difrng:= 1;
reset(hifile);
reset(lofile);
WHILE NOT (EOF(hifile) OR EOF(lofile)) DO
  BEGIN
  hirow:= hifile^;
  get(hifile);
  lorow:= lofile^;
  get(lofile);
  FOR jj:= 0 TO maxcol DO
    BEGIN
    dif:= hirow[jj]-lorow[jj];
    outrow[jj]:= mingrl+(((dif-mindif)*grlrng) DIV difrng)
    END;
  outfile^:= outrow;
  put(outfile);
  END;
END.
```

III. Applications to X-ray angiographic images

We have described in Chapter I the rank-max filter $RM[k, W]$ and its main properties. In Chapter II we have given its PASCAL implementation for rectangular $h \times w$ windows. We will now give examples of its application in the processing of digitized X-ray arterial images.

In Section I.1 we said that an advantage of the rank-max filter over the min-max filter is that, as the rank increases, it becomes less sensitive to small holes in peaks and ridges, for example dark speckle noise. Another advantage is that it produces less artefacts. We assume that each window $W(p)$ is the translate $p + B$ of a fixed template B. Suppose that we have a peak or ridge portion X which is large enough to contain a translate of the window B, but whose outline does not match that of B (see Figure 8). Then with the min-max filter only the part X_B of X which can be filled by translates of B will be preserved, and $X - X_B$ will appear in the difference between the original and filtered image. But this difference image is supposed to show peaks and ridges narrower than B, and so we have an artefact. Now if we take the filter $RM[k, W]$ for a value of k a sufficiently large, many points of $X - X_B$ belong to a translate of B having less than k points outside X, and so the artefact in the difference image will be greatly reduced.

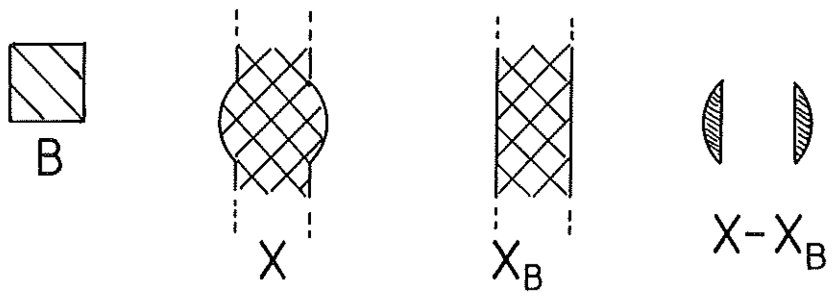

Figure 8. *Artefact produced by the min-max filter in the difference image*

Let us now give a few examples of the application of the rank-max filter $RM[k, h \times w]$ to angiographic images. We used digital images on a 256×256 grid with 256 grey-levels (this fact is reflected in the choice of constants of our PASCAL program, see Section II.2). The filter was applied with square windows ($h = w$), and the 'plus' variant was adopted for the choice of grey-level in the frame surrounding the grid (see Section I.3), in order to minimize changes along the border.

First, by taking very small windows (2×2), one can eliminate a large proportion of bright speckle noise (isolated peaks) or scratches. This is the case indeed for the min-max filter $RM[1, 2 \times 2]$, which eliminates all peaks too small to contain *four* mutually adjacent points. We have taken the less active filter $RM[2, 2 \times 2]$, which eliminates all peaks too small to contain *three* mutually adjacent points. In Plate 1 we show an image with very bright thin horizontal scratches. As seen in Plate 2, most of them (those one point wide) are eliminated by the filter. In Plate 3 we show another image with bright horizontal noise artefacts (although less conspicuous than in Plate 1), and a close look at Plate 4 shows that they are reduced. This is more visible if we enhance ridges by taking the difference between the image and the one obtained by a min-max filter with 16×16 windows. Plates 5 and 6 show this enhancement applied to Plates 3 and 4 respectively, and the reduction of horizontal artefacts becomes clear.

The second application of rank-max filters is, as we have just seen with Plates 5 and 6, the extraction and enhancement of blood vessels. We take relatively large windows (16×16), and the filter $RM[k, 16 \times 16]$ will eliminate most blood vessels, since they are ridges narrower than 16 points. Then the difference between the original and the filtered image shows these blood vessels. (NB: We linearly enhace the contrast of the difference image in order to fill the whole grey-level range —see the program `difcont` at the end of Chapter II). This is illustrated for $k = 1$ in Plates 5 and 6, which correspond to Plates 3 and 4. Clearly Plate 6 gives cleaner result, and so we can safely restrict our enhancements to Plate 4. We try two other small values of k, 2 and 7; this gives Plates 7 and 8. A comparison of Plates 6, 7, and 8 shows that an increase of the rank k leads to a a diminution of background texture in the blood vessel enhancement.

The improvement due to the bright noise reduction by $RM[2, 2 \times 2]$ and to the increase of k in the enhancement $\mathbf{id} - RM[k, 16 \times 16]$ can be quantitatively estimated by comparing the compression ratio when these images are coded. We have translated these images into Postcript with a compression by run-length encoding, and the size of the processed images decreases monotonically from Plate 3 to Plate 8.

Other examples are given in Plates 9 to 12. Here we first apply $RM[3, 3 \times 3]$ to Plate 3 (see Plate 9), and then enhance blood vessels with $\mathbf{id} - RM[k, 16 \times 16]$, where we take successively $k = 1$, 3, and 9 (Plates 10, 11, and 12). The compression ratio of the run-length encoding for these four plates is yet stronger than the one for Plates 4, 6, 7, and 8 respectively.

We have thus outlined two stages in the extraction of blood vessels in digitized X-ray angiographic images. First a reduction of bright speckle noise by $RM[a, s \times s]$ for a small s (say, $s = 2$) and a of the order of s, then an enhancement by $\mathbf{id} - RM[k, \ell \times \ell]$, where ℓ is relatively large and k can be chosen in the range between 1 and ℓ.

We have compared the results obtained by the rank-max filter with those given by its particular case, the min-max filter, the advantage being in a reduction of background noise. Of course, if we take too large a rank k, very small blood vessels will not be preserved in the difference image, but we can already obtain interesting results with small values for k. We can also compare them with what we get by the two methods given in [1]. The 'basic method' (Section II of [1]) uses a smoothing followed by a min filter, and it produces in the difference image artifactual ridges corresponding to ramps in the original image. The 'skeleton-controlled' method (Section III of [1]) avoids this type of artefact, but it is based on complicated and time-consuming skeletonization and skeleton filtering procedures; moreover it tends to eliminate small or fuzzy portions of blood vessels, while leaving a certain amount of textural noise artefacts. By relaxing the skeleton-filtering stage, the first defect (loss of blood vessels) is reduced, but the second one (textural noise) is augmented.

This shows that a simple, but well-conceived filter, can give nice results, while using only limited computing resources.

References

[1] P.A. Devijver, C. Ronse, P. Haaker, E. Klotz, R. Koppe, R. Linde: Pseudomask technique for digital subtraction angiography (DSA). *Mustererkennung 1984*, Informatik-Fachberichte, Vol. 87, Springer-Verlag (1984), pp. 230-236.

[2] H.J.A.M. Heijmans, C. Ronse: The algebraic basis of mathematical morphology; part I: dilations and erosions. *CWI Report AM-R8807, PRLB Manuscript M248* (1988).

[3] T.S. Huang, G.J. Yang, G.Y. Tang: A fast two-dimensional median filtering algorithm. *IEEE Trans. Acoustics, Speech, & Signal Processing*, Vol. ASSP-27, no. 1, pp. 13–18, 1979.

[4] Y. Nakagawa, A. Rosenfeld: A note on the use of local min and max operations in digital picture processing. *IEEE Trans. Systems, Man, & Cybernetics*, Vol. SMC-8, no. 8, pp. 632–635, 1978.

[5] C. Ronse: Erosion of narrow image features by combination of local low rank and max filters. *Proceedings of the Second International Conference on Image Processing and its Applications*, London (1986), pp. 77-81.

[6] C. Ronse: Order-configuration functions: mathematical characterizations and applications to digital signal and image processing. *Information Sciences* (to appear).

[7] C. Ronse, H.J.A.M. Heijmans: The algebraic basis of mathematical morphology; part II: openings and closings. (In preparation).

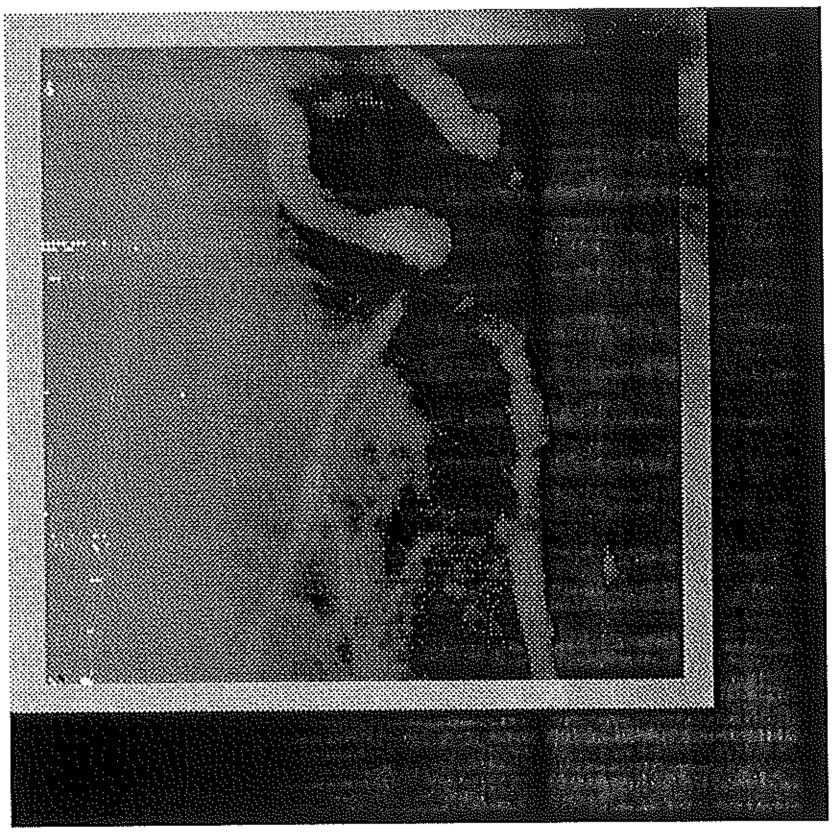

Plate 1.
Original image **A**.

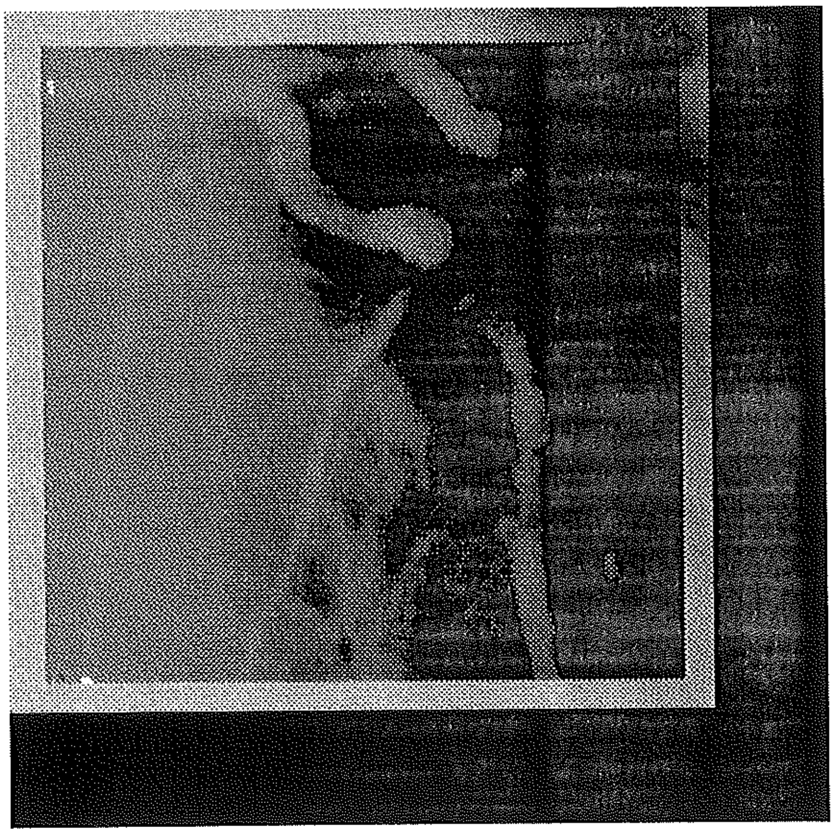

Plate 2.
$\mathbf{A}' = RM[2, 2 \times 2](\mathbf{A})$.

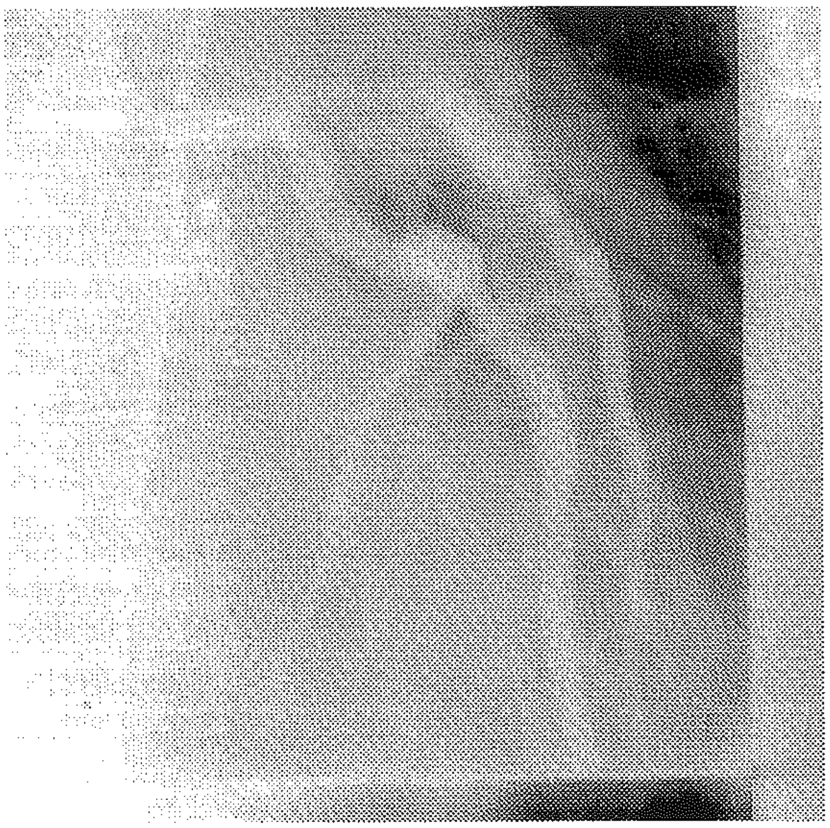

Plate 3.
Original image **B**.

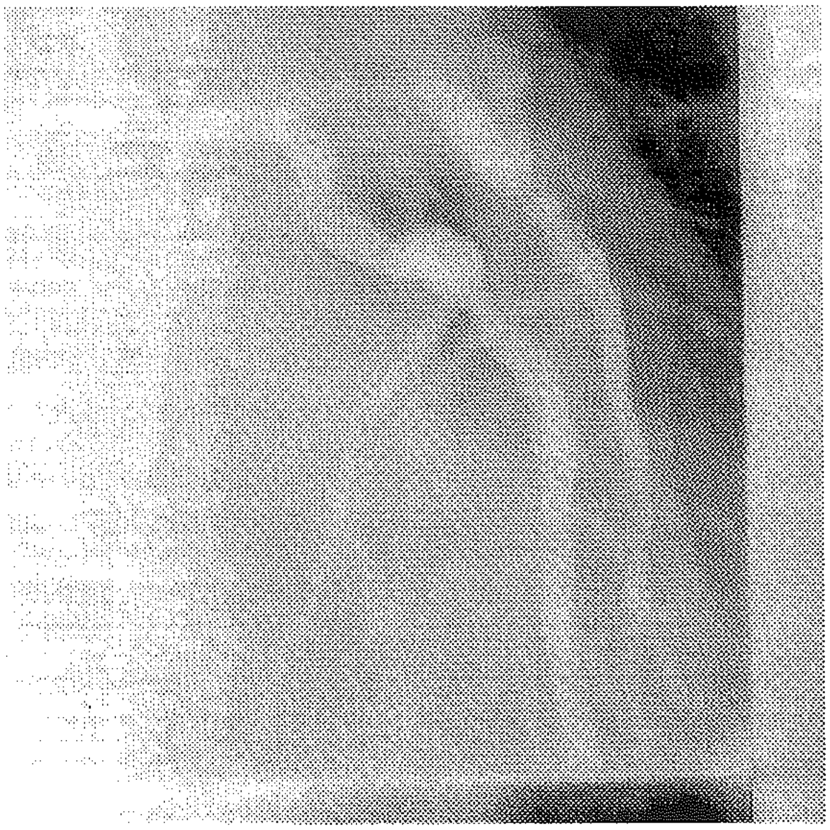

Plate 4.
$\mathbf{B}' = RM[2, 2 \times 2](\mathbf{B})$.

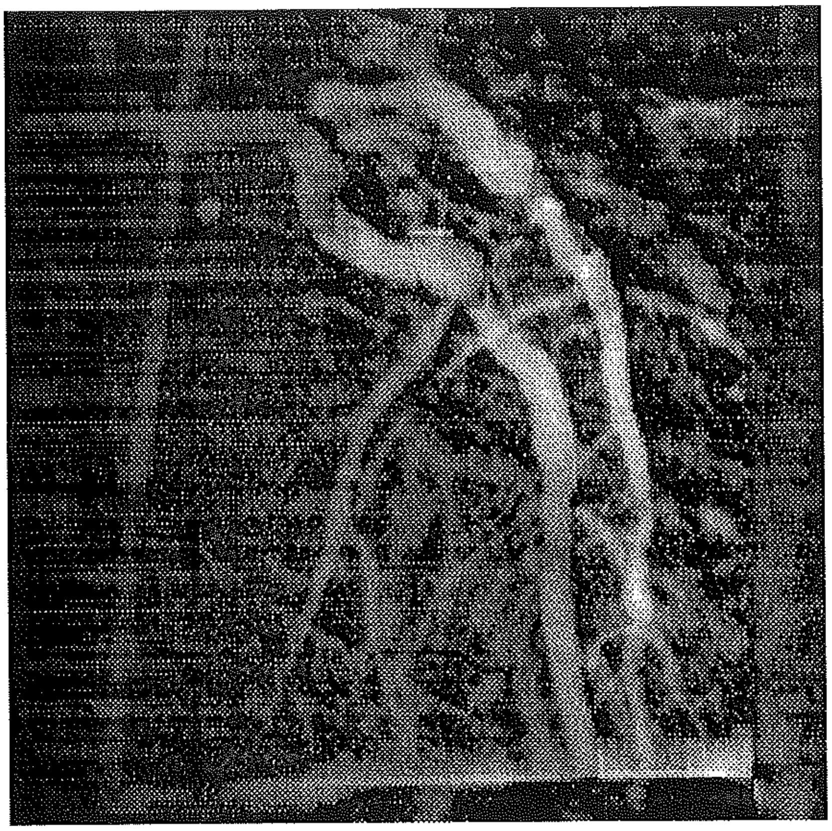

Plate 5.
$\mathbf{C}_1 = \mathbf{B} - RM[1, 16 \times 16](\mathbf{B})$.

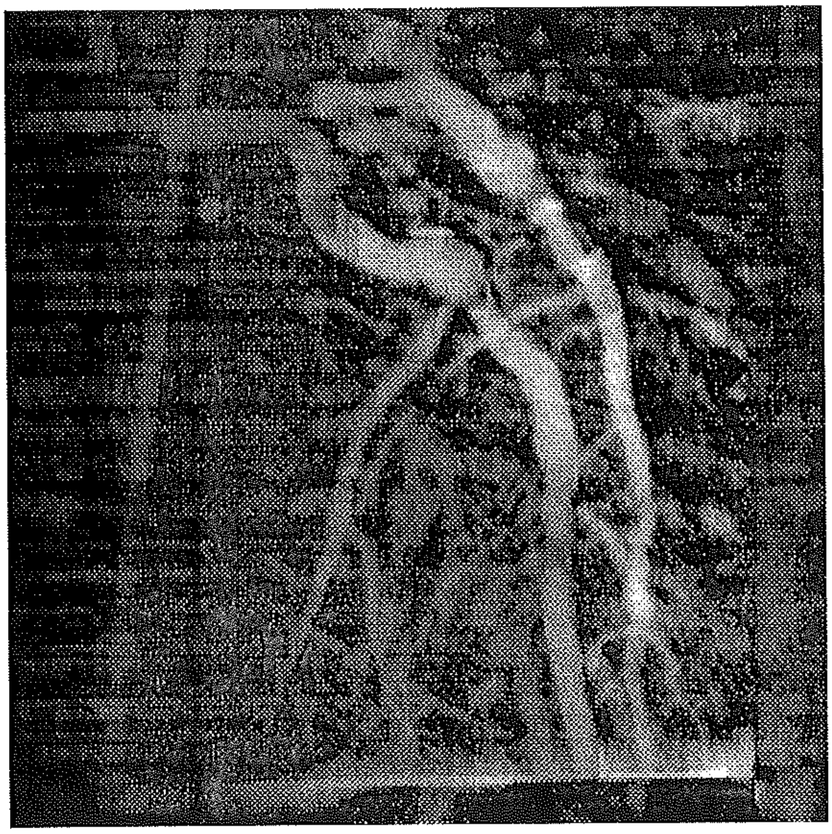

Plate 6.
$\mathbf{C}'_1 = \mathbf{B}' - RM[1, 16 \times 16](\mathbf{B}')$.

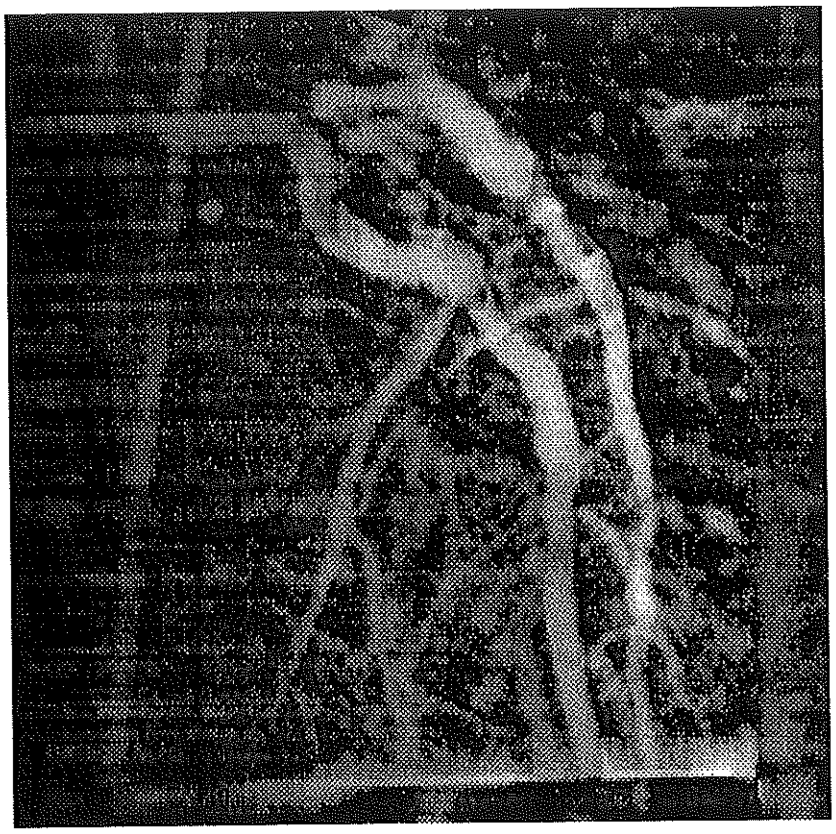

Plate 7.
$\mathbf{C}'_2 = \mathbf{B}' - RM[2, 16 \times 16](\mathbf{B}')$.

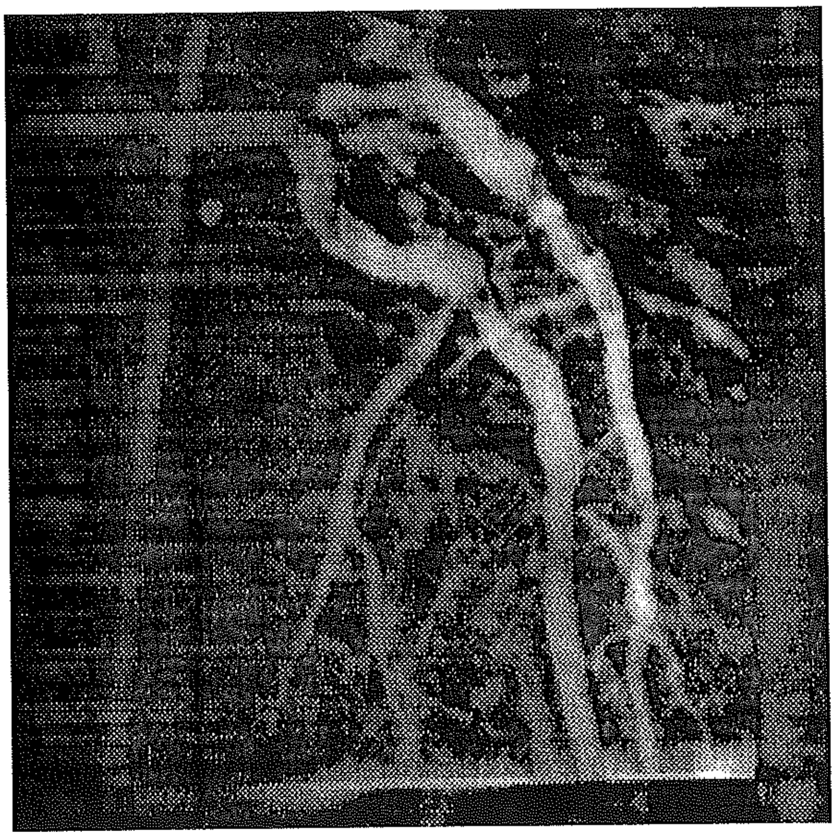

Plate 8.
$\mathbf{C}'_7 = \mathbf{B}' - RM[7, 16 \times 16](\mathbf{B}')$.

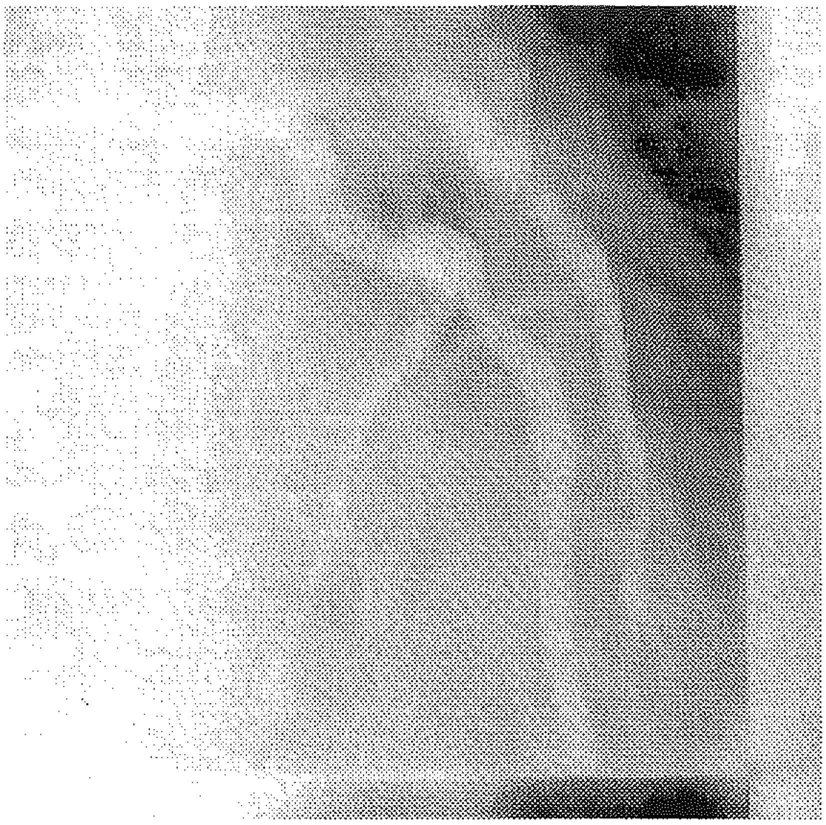
Plate 9.
$\mathbf{B}'' = RM[3, 3 \times 3](\mathbf{B})$.

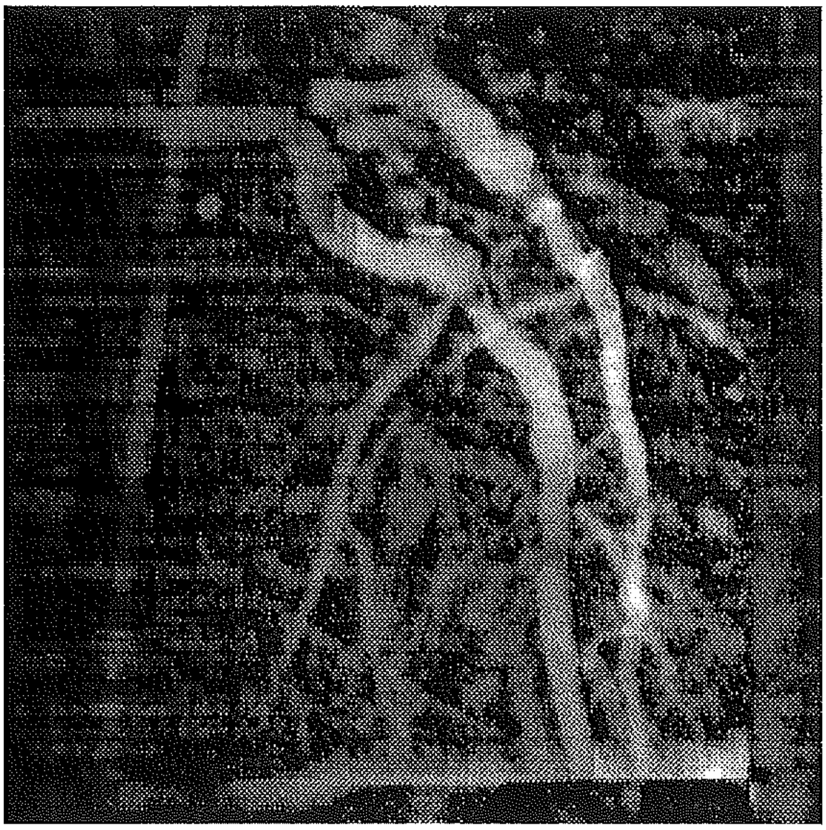
Plate 10.
$\mathbf{C}_1'' = \mathbf{B}'' - RM[1, 16 \times 16](\mathbf{B}'')$.

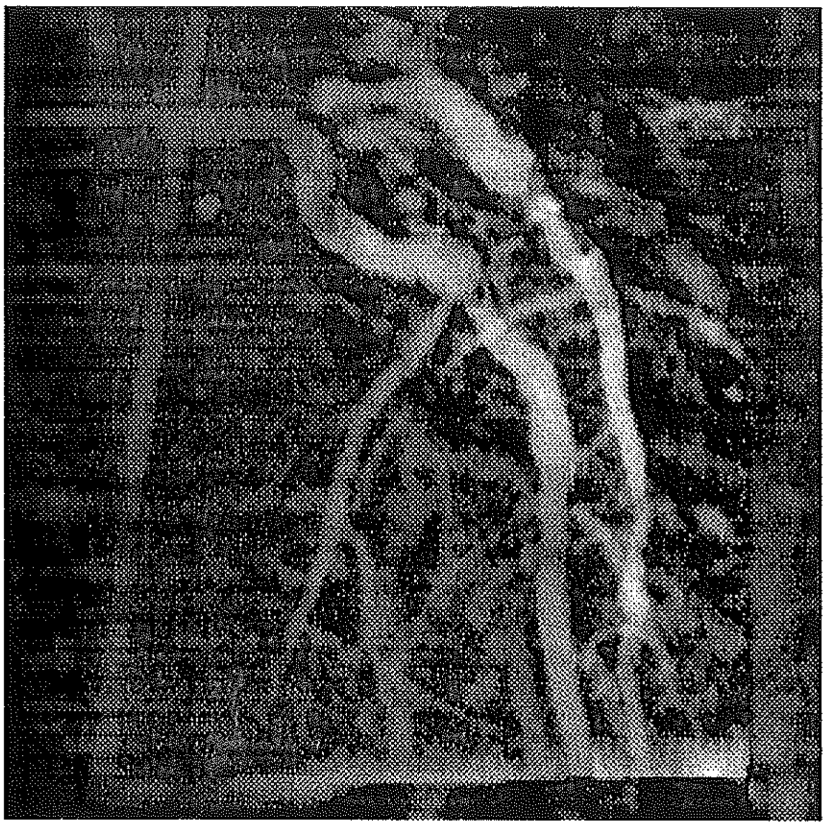

Plate 11.
$\mathbf{C}''_3 = \mathbf{B}'' - RM[3, 16 \times 16](\mathbf{B}'')$.

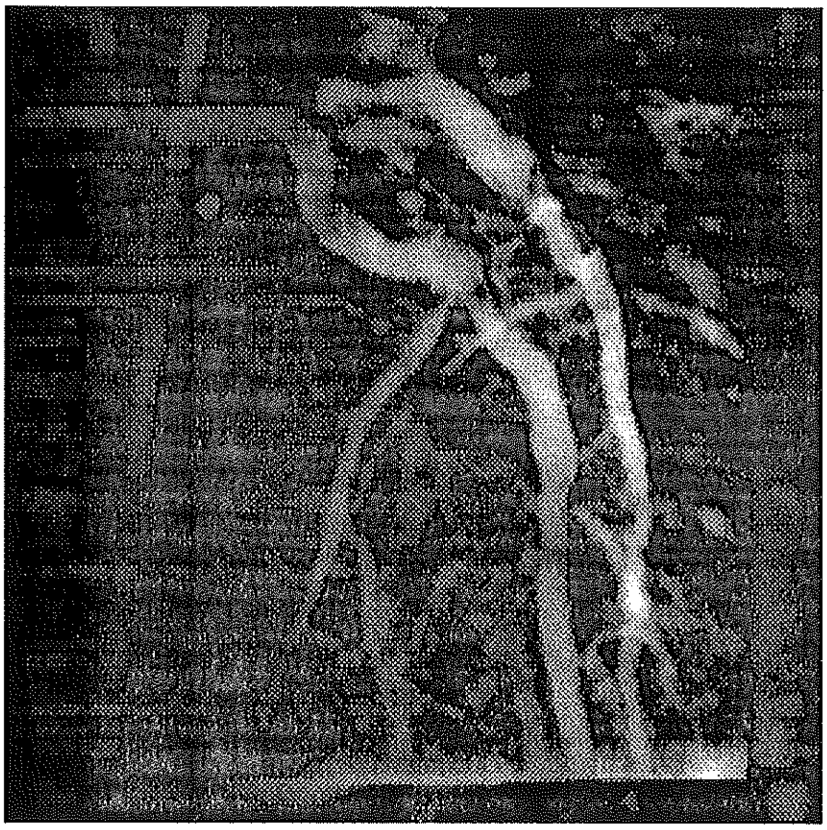

Plate 12.
$\mathbf{C}''_9 = \mathbf{B}'' - RM[9, 16 \times 16](\mathbf{B}'')$.