Microsc. Microanal. Microstruct. **6** (1995) 159–177 APRIL **1995**, PAGE 159

Classification
Physics Abstracts
41.80 — 42.30

Review paper

Image Algebra for Electron Images

Peter W. Hawkes

CEMES - LOE du CNRS, B.P. 4347, F-31055 Toulouse Cedex, France

(Received February 27; accepted April 10, 1995)

Résumé. — Une partie toujours en augmentation des algorithmes utilisés dans le traitement des images a été exprimée à l'aide de l'algèbre de l'image. Les aspects de cette algèbre les plus intéressants pour l'optique électronique sont présentés et son adoption systématique est prônée.

Abstract. — An ever-increasing proportion of the algorithms employed in image processing are being translated into the language of image algebra. The aspects of this algebra of particular interest in electron optics are presented and the advantages of adopting it as a lingua franca are described.

1. Introduction

For many years, images were processed in a piecemeal fashion in the many and varied fields in which the computer and the image were brought together: electron microscopy of course but also forensic science and medicine, the textile industry and agriculture, astronomy and military reconnaissance. Each field introduced its own vocabulary, adopted its own notation and wrote its own programs, with the result that there was much duplication of effort and the transfer of procedures from one domain to another was not easy. The authors of the principal treatises on digital image processing have attempted to impose some pattern on this mass of disparate material and four broad types of activity have emerged: acquisition and coding; enhancement; restoration; and image analysis and pattern recognition. We shall explain later what is meant by these headings, which are adopted here.

In the 1980s, a real attempt was made, encouraged by the US Air Force, to construct a theoretical framework into which as many as possible - ideally all - of the image processing techniques could be fitted in some standard way. A computer language or "dialect" could then be created for representing and implementing the techniques in a standard way; it was hoped that all the different specialists would adopt this language, thereby providing a direct gateway from the algorithms of one specialist area to those of another. The aim can be stated simply: find a mathematical structure, as simple as possible, making only modest demands on the mathematical sophistication of the user and remaining as close as possible to familiar notions, in terms of which any image manipulation can be expressed.

The result was a number of Image Algebras [11, 25, 38, 47, 51], of which we describe only one here, the image algebra first presented at length by Ritter, Wilson and Davidson [51]. This algebra satisfies all the conditions listed above. It is still young, however, and is in continual evolution, extension and improvement. We confine the following introduction to the basic ideas, in the hope that the authors of new ideas in electron image processing will be tempted to express their work in the vocabulary and notation of this "electron image algebra".

Before embarking on details of the algebra, it seems helpful to give a brief idea in qualitative language of the material to be discussed. The closing paragraphs of this introduction will thus be repeated in different terms in subsequent sections.

Most image processing algorithms are described at *pixel* level. An extremely simple procedure for smoothing images, for example, consists in replacing the grey level at each pixel of the image by the average of the grey levels of the pixel in question and of those of its immediate neighbours. A most important feature of image algebra is that all the basic operations are applied to *images* and the rules, which usually must be applied at pixel level, are relegated to a secondary position. We shall see that we need to consider a host of different kinds of images, all of which are two-dimensional arrays of some kind and that it is therefore important to be very clear about the kind of image being treated at any given time - the mathematics will be organized in such a way as to emphasize this.

The operations that we shall need are reassuringly familiar: addition and multiplication and their duals or inverses, subtraction and division. In addition, we shall need the operation of max (sup) and min (inf). Considerable care is needed when setting up the formal structure to ensure that the definitions of these operations are free of self- contradictions and other traps but here we shall only need their simple, everyday meanings. If we compare two images at a given point, for example, or compare the grey-level values at neighbouring pixels of an image, then "max" returns the largest grey level of those compared, or the set of pixels with this largest grey level if several have this largest value.

Surprisingly, this is all that is needed: a rather general definition of what we mean by an image and some care in defining these basic and familiar operations. Thus armed, all known linear and nonlinear image processing procedures can be written compactly in the formalism of image algebra.

One final point needs to be made before we examine the algebra in more detail: as I have presented it above, image algebra appears to be no more than a convenient and reasonably transparent way of rewriting material already known and this was indeed the original ambition of its inventors. If it were no more than that, it would be of moderate interest but probably no more than moderate. In fact, the rather different way of thinking about image processing that image algebra calls for has proved immensely fruitful. Very new ideas have emerged, amazing relations between fields that had hitherto had no apparent connection have been established and the catalytic role of the algebra has been immense. It is really for this reason that I felt that it was worth writing an account specifically aimed at the electron image processing community.

2. Types of Images

The images considered here are all essentially two-dimensional, though we may be led to consider higher-dimensional structures. This is not an intrinsic limitation of the algebra but will more than suffice for our present purposes. One-dimensional images are simply regarded as images with a single row or column and it will be convenient to illustrate many of the operations in terms of such "degenerate" images, for clarity of presentation.

An image a is a set of coordinate values and of the values of the image at each coordinate. We usually think of an image as a discrete pattern of grey levels, the image having been divided up into small zones, usually squares or rectangles, called pixels; some numerical measure of greyness, between black and white, will have been associated with each pixel. This is the simplest kind of image and we write

$$a = \{((i,j), a(i,j)) | i = 0,1,...,I-1, j = 0,1,..,J-1\} \tag{1}$$

$$a(i,j) \text{ is real (or an integer).}$$

This tells us that the image a is a *set* of positions (i,j) and of grey-level values at each of these positions. We have also specified that the image is rectangular, with $I \times J$ pixels and that the grey-levels are real numbers (or integers, since they are often coded as an 8-bit number, $0-255$). A similar definition is in fact suitable for *all* images and we in general write

$$\begin{array}{l} a = \{(x, a(x)) | x \text{ has the form...}\} \\ a(x) \text{ is of the following nature...} \end{array} \tag{2}$$

We now consider the most common extensions of the elementary image (1). The pixel values of a measured image will usually be real numbers ($a(x) \in \mathbb{R}$) or integers ($a(x) \in \mathbb{Z}$) or perhaps real numbers with a limited range: for some applications, it is convenient to make "black" equal to zero and "white" equal to unity, for example. But as soon as we perform a Fourier transform, the pixel values are likely to become complex; we then have $a(x)$ is complex or ($a(x) \in \mathbf{C}$).

Another type of image has more than one value associated with each pixel. Two obvious examples are colour images, and scanning electron microscope images. In the former, we have values corresponding to each of the primary colours and so three values will be associated with each pixel. In the latter, several signals may be collected from each object element and each of these generates an image: the secondary electron image, the backscattered electron image and so on. In this case, we speak of a *multi-valued image*. When we come to specify $a(x)$, we write

$$a(x) = (a_1(x), a_2(x), a_3(x), ..., a_n(x)) \tag{3}$$

for an n-valued image and we have to specify the nature of each of the $a_i(x), i = 1,2,...,n$; some may be integers, others real numbers. Some may be codes or symbols: if we have identified the presence of a particular chemical element, for example, it may be useful to regard it as one of the "values" of the image, labelling it with its atomic number say. Although these may look like integers in a restricted range, it is of course imperative to specify that they are in reality codewords, to which arithmetic operators such as addition are not to be applied !

The next common degree of complexity is represented by *vector-valued images*. In analytical electron microscopy, we associate a set of values with each pixel when we measure the energy-loss spectrum there. It is not sensible to regard this a multi-valued image, where the different values are associated with different kinds of signals. Figure 1, which shows these various situations schematically, illustrates the difference. In a vector-valued image, an ordered set or vector is associated with each pixel, the elements of the vector being the (sampled) energy-loss spectrum in the EELS example. We thus have

$$a = \{(x, a(x)) | x \in ...\} \tag{4}$$

$$a(x) \text{ is a vector with } N \text{ real elements.}$$

The elements may of course be complex or integers, or belong to any other "value set".

The final common image type is so important that it is given a special name: we shall repeatedly encounter *templates* below. These are *image-valued images*, an entire image being associated with each pixel. Numerous image-processing operations call for such images and there has been a tendency to think of them as mathematical constructs, but in fact they occur naturally, notably in the STEM and indeed in all instruments in which object and image are not conjugates. In the STEM, each object-element gives rise to a two-dimensional current distribution in the detector plane and this is usually reduced to a very small number of numerical values by capturing part of the current on different detectors: a central (circular) bright-field detector (in practice replaced by a dispersive element), an annular detector or perhaps a quadrant detector. But in principle, and in a few instruments in practice as well, the current distribution from each object- element can be recorded as an image with the result that an entire image is associated with every object-element. The STEM generates templates! [32, 33]. Another common example is furnished by the point-response function, which represents the image of an object-element (originally, a point) and will vary with the position of the object-element unless the imaging is assumed to be isoplanatic.

In other to describe a template, we need *two* coordinate sets, one for the pixels of the basic image and another for the images associated with each pixel of the basic image. In Figure 1d, the basic image might be 64×64 pixels and the coordinate set Y would be of the form $\{(i,j)|i = 0,...,63, j = 0,...,63\}$. The images associated with individual pixels might then be much smaller and belong to a coordinate set X of the form $\{(p,q)|p = 0,1,2,3, q = 0,1,2,3\}$. For each point y belonging to Y, a template t will be an image: $t(y)$ is an image on X. To avoid having to describe the pixel-values of these images by $(t(y))(x)$, we write

$$t_y \equiv t(y) \tag{5}$$

so that a template t consists of the images

$$t_y = \{(x, t_y(x)|x \in X, t_y(x) \in F\} \tag{6}$$

in which F denotes the value set to which the image values belong - real numbers, integers, complex numbers or even vectors or more complicated structures. It sometimes happens that the image-values, $t_y(x)$ above, are themselves images and we are then confronted with template-valued templates ! This is fortunately not common.

Before leaving templates, a little more vocabulary is convenient. Y is often called the "target domain" of t while X is called its "range space". A point y is then the "target point" of t and the $t_y(x)$ are sometimes known as the "weights" of t at y.

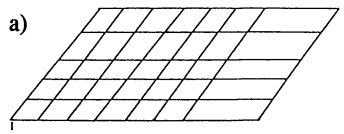
Templates may be translation-invariant or variant. When invariant, the weights $t_y(x)$ are the same for all target points y and the template can be represented simply by a single pattern. It is then helpful to note that $t_y(x)$ is of the form $t(y - x)$.

3. Operations on Images

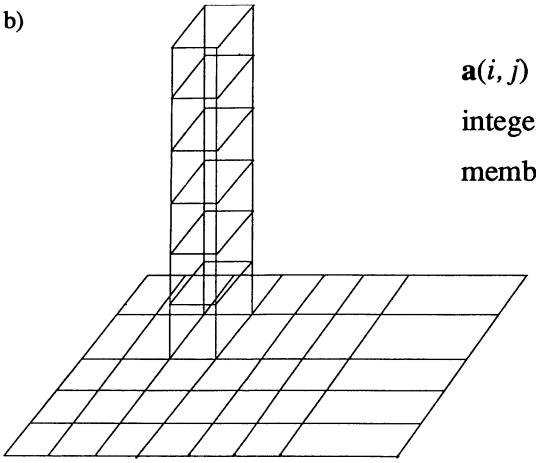
The three basic operations are very familiar: addition, multiplication and maximum. In what follows, we consider real-valued images; the operations are not of course restricted to $F = \mathbb{R}$ but some thought is needed if the value set F is, for example, the integers $0 - 255$ (corresponding to a grey-level image coded on eight bits); any of these operations is more than likely to give a result that does *not* belong to F !

Addition $(+)$, multiplication $(*)$ and maximum $(\vee)$ have their everyday meanings. Let a and b be real-valued images defined on X. Then $a + b$ is a new image, c, such that

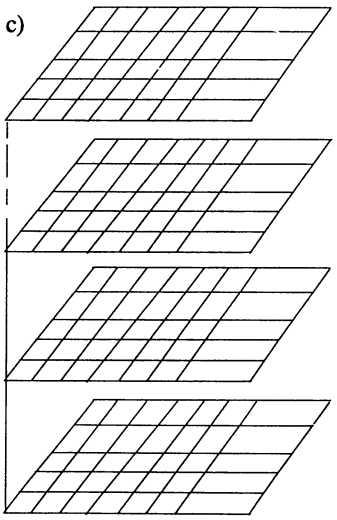
$$a + b = c, c = \{(x, c(x)|(c(x) = a(x) + b(x), x \in X\} \tag{7}$$



$\mathbf{a} = \{i, j, \mathbf{a}(i, j)\}$; $\mathbf{a}(i, j)$ is an integer or a real number or a complex number



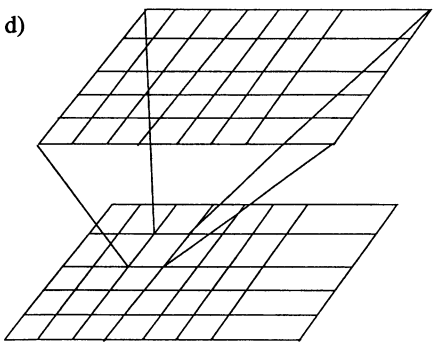
$\mathbf{a}(i, j)$ is a vector, the elements of which may be integers, real numbers, complex numbers or members of some other value set



$\mathbf{a} = \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \ldots \mathbf{a}_n\}$;

$\mathbf{a}$ is a multi-valued image,

$\mathbf{a}(i, j) = \{\mathbf{a}_1(i, j), \mathbf{a}_2(i, j), \mathbf{a}_3(i, j), \ldots \mathbf{a}_n(i, j)\}$



$\mathbf{a}$ is a template

$\mathbf{a}(i, j)$ is an image

$\mathbf{a}(i, j) = \{k, l, \mathbf{a}_{ij}(k, l)\}$

Fig. 1. — The different types of image. a) Scalar-valued image. b) Vector-valued image. c) Multi-valued image. d) Image-valued image, or template.

and likewise for $a * b$ and $a \vee b$. Subtraction, division and minimum are obtained in terms of these definitions. Exponentiation and logarithm are also needed:

$$a^b = (x, c(x) | c(x) = a(x)^{b(x)} \quad \begin{array}{l}\text{unless } a(x) = 0,\\ \text{in which case } c(x) = 0, x \in X\}\end{array} \tag{8}$$

$$\log_a b = \left\{(x, c(x) | c(x) = \log_{a(x)} b(x), x \in X\right\} \tag{9}$$

and it is assumed here, to avoid digression, that $a(x)^{b(x)}$ is real in a^b and that $a(x)$ and $b(x)$ are both greater than zero in the logarithm. Note that the base of the logarithm may be an image, a much more general definition than the common one in which the basis is typically 10 or e or 2; nevertheless, such general definitions are rarely needed and in practice, $a(x) = e$ for all x in X.

Any function may be applied to an image. Thus sin a, for example, is an image

$$\sin a = \{(x, \sin a(x)) | x \in X\} \tag{10}$$

and in general

$$f(a) = \{(x, f(a(x))) | x \in X\} \tag{11}$$

The most commonly used function is the characteristic function with respect to some set S, denoted χ_S, which is a compact way of noting whether or not some quantity belongs to that set:

$$\begin{array}{lll}\chi_S(a) & = \{(x, \chi_S(a(x))) | x \in X\} & \\ & = c = \{x, c(x) | c(x) & = 1 \text{ if } a(x) \in S \\ & & = 0 \text{ otherwise}\}\end{array} \tag{12}$$

When the image is multi-valued, the operator may not be the same for all the image strata. We then require the set of operations, $o = \{o_1, o_2, ...o_n\}$; the operation a o b, where a and b are n-valued images, will now return the n-valued image c, where

$$c(x) = (a_1(x) o_1 b_1(x), a_2(x) o_2 b_2(x), ...a_n(x) o_n b_n(x))$$

This only allows combinations of corresponding strata of images, however: there are no terms of the form $a_2(x) o_{21} b_1(x)$, for example. This is a weakness for, if a_1, a_2 are the real and imaginary parts of a complex-valued image, we shall need to be able to combine different strata of the multi-valued image. A generalization is described in [30], in which a new law of composition is defined:

$$c(x) = (a(x) o_1 b(x), a(x) o_2 b(x), ...a(x) o_n b(x))$$

The generalization to operations between multi-valued images and multi-level templates is straightforward and although this has not yet found practical application, it will be needed in connection with morphological spectra and the associated expansions and developments envisaged by Haralick *et al.* [27]; we return to this briefly below.

As we know, templates are images of a particular kind and there seems no need to discuss them separately. In practice, operations between an image and a template usually require *two* of the basic operators just listed and it is very convenient to have a compact notation for such combinations. It has therefore become usual to speak of "generalized products" between images and templates though the word "product" is not always strictly appropriate. These products differ from one another in the choice of pair of operators. The simplest is the generalized convolution (strictly, the generalized backward convolution), denoted by the symbol $\oplus$. The generalized convolution between an image a and a template t is given by

$$a \oplus t = b, b = \left\{(y, b(y)) | b(y) = \sum_x a(x) t_y(x), y \in Y\right\} \tag{13}$$

We return to and illustrate this below but we first mention some of the other generalized products that have proved important.

The (backward) additive maximum is defined by

$$a \boxtimes t = \left\{ (y, b(y)) | b(y) = \bigvee_x a(x) + t_y(x), y \in Y \right\} \tag{14}$$

and the (backward) multiplicative maximum

$$a \circledvee t = \left\{ (y, b(y)) | b(y) = \bigvee_x a(x) t_y(x), y \in Y \right\} \tag{15}$$

These definitions will become much easier to understand once we have shown them in use in practice. But even now, we can appreciate an extremely interesting aspect of them: the original image a is defined on X but the coordinate set Y on which the result is defined may be different. Thus template operations can - if necessary - be used to generate images of a new type from a basic image. Each of these operations has a "forward" version, which we define without comment:

$$t \oplus a = \left\{ (y, b(y)) | b(y) = \sum_x a(x) t_x(y), y \in Y \right\} \tag{16}$$

$$t \boxtimes a = \left\{ (y, b(y)) | b(y) = \bigvee_x a(x) + t_x(y), y \in Y \right\} \tag{17}$$

$$t \circledvee a = \left\{ (y, b(y)) | b(y) = \bigvee_x a(x).t_x(y), y \in Y \right\} \tag{18}$$

4. Image Processing

The four themes of digital image processing are now considered in more detail. We here part company with the major and authoritative texts on image algebra, upon which we have leaned heavily in preparing this account. There, the various image–image, image–template and template–template operations are examined in turn. Here, we have preferred to examine the usual activities of image processing and to show that image algebra has something, and often much, to offer in each category.

4.1 IMAGE ACQUISITION AND CODING. — The raw image must be digitized before transfer to the computer, which involves sampling and quantization. During sampling, the quasi-continuous current density distribution in the image plane of the microscope is broken up into discrete elements, typically squares, and the mean value in each element is recorded. The result is a two-dimensional array, or matrix of grey-level values, usually 64×64, 128×128 or in general $2^n \times 2^n$. Quantization is the replacement of the quasi-continuous range of grey levels between black and white by a discrete set of grey levels, often coded on eight bits and hence numbered $0 - 255$.

The image in discrete form therefore occupies a considerable volume of computer memory and transfer between computer centres will be time-consuming, at least with present-day technology.

Ways of coding images more efficiently have therefore been explored and among the many possibilities, transform coding and vector coding have emerged as among the most efficient. Both are lossy, in the sense that the image reconstructed from the coded image is not identical with the original; in the case of transform coding, this is intrinsic to the method, while for vector coding, it is inevitable to keep the codebook of reasonable size. We shall consider only these two approaches here.

Transform coding is simply explained: a discrete transform such as the discrete Fourier transform of the image is obtained and the high-frequency elements of the transform are discarded beyond a certain spatial frequency, at which little or no useful signal is thought to survive. Particularly efficient transforms for this purpose are known [3, 44 or Sects. 71.4 of 36]. Vector coding starts from the idea that there is some compromise to be found between codebook size and the number of pixels associated with a given codeword. We could, for example, associate a single codeword with every possible group of four pixels (Fig. 2), so that a 64×64 image would need only 16×16 codewords but the latter would be long and numerous. The possible combinations of four pixel-values are not equally likely to occur, however, so by allowing only the most probable, both the number and length of the codewords can be reduced, at the expense of accepting a lossy coding.

Image algebra in coding theory has not attracted much attention. However, the image algebra representation of discrete transforms, the Fourier transform especially, have been very thoroughly studied, notably by Gader [48], who shows that such transforms may be expressed naturally as an image–template operation, the template being characteristic of the particular transform. For the discrete Fourier transform, we know that

$$\hat{a}(u,v) = \sum_{k=0}^{n-1} \sum_{j=0}^{m-1} a(j,k) \exp\{-2\pi i(ju/m + kv/n)\} \tag{19a}$$

with inverse

$$a(j,k) = \frac{1}{mn} \sum_{v=0}^{n-1} \sum_{u=0}^{m-1} \hat{a}(u,v) \exp\{2\pi i(uj/m + vk/n)\} \tag{19b}$$

The template f, the elements of which are defined by

$$f_{(u,v)}(j,k) = \exp\{-2\pi i(ju/m + kv/n)\} \tag{20}$$

will thus yield the Fourier transform or its inverse in terms of the operator $\oplus$ defined above (13):

$$\hat{a} = a \oplus f \tag{21a}$$

with inverse

$$a = \frac{1}{mn}(\hat{a} \oplus f^*) \tag{21b}$$

Transform coding consists in suppressing part of $\hat{a}$ corresponding to higher spatial frequencies. The more complicated case of the Karhunen-Loève transform, in which the elements of f are defined in terms of the image being coded, is considered by Ritter *et al.* [49].

Vector coding has been recognized to be a type of mapping not hitherto analysed, namely, an image–template mapping [34]. If we think of the pixel-clusters described by the various codewords as images, we see that the vector-coded image (16×16 in the example mentioned above) is in fact an image-valued image, or template. It is of course a variant template. The operation of replacing the basic image by its vector- coded version is equivalent to mapping an image into a template.

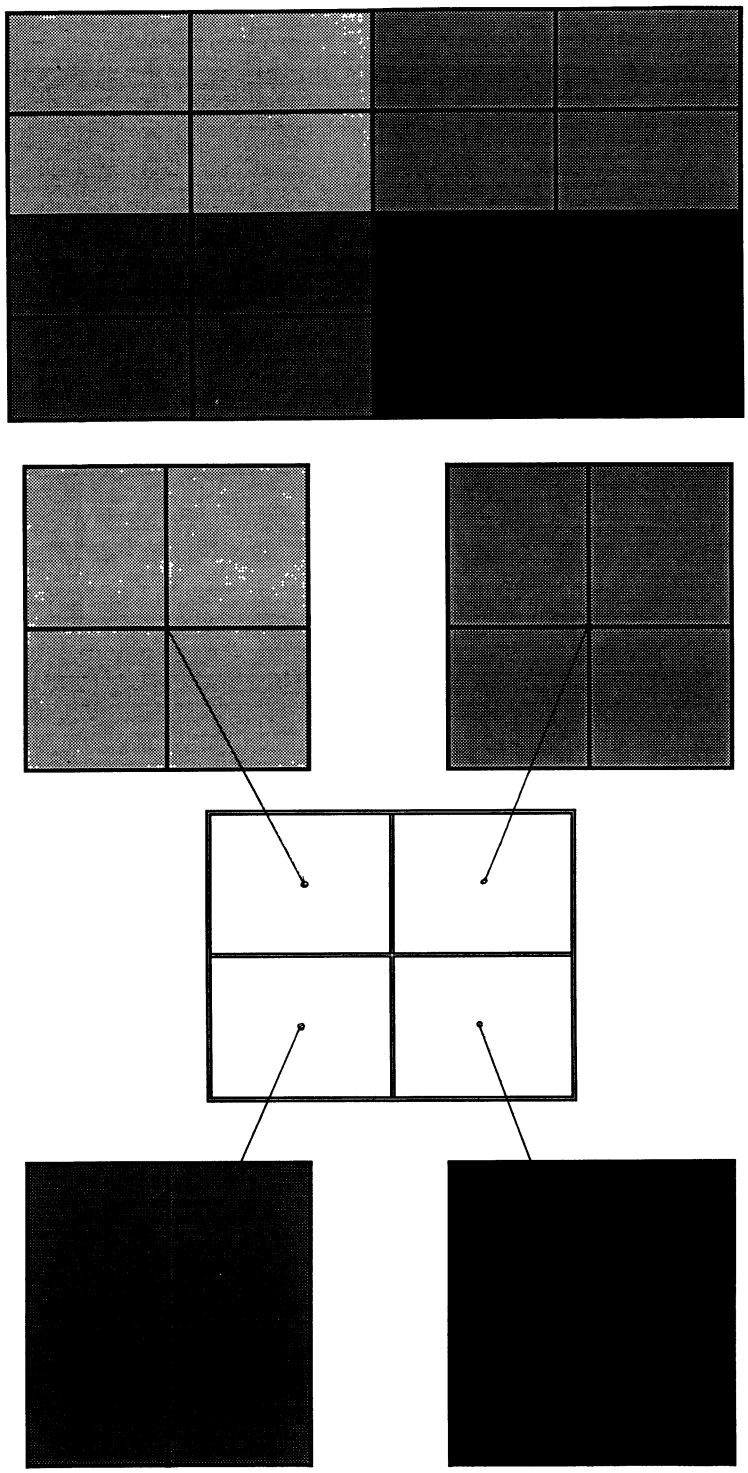


Fig. 2. — Vector coding. A single codeword is given to the group of four grey levels associated with each cluster of four pixels (above). Each group of four grey levels is regarded as an image. After coding (below), the image has one quarter the number of pixels but an image is associated with each. Vector coding is thus an image–template mapping.

The implications of this observation remain to be investigated in detail but it is already clear that the considerable volume of theory dealing with templates immediately becomes available for vector-coded images. We note in passing that this mapping generates a *parametrized* template, since the pixel-values (images) of the template are determined by those of the original image. We shall meet other examples of parametrized templates in connection with image enhancement and adaptive filtering.

4.2 ENHANCEMENT. — An important branch of activity in image processing consists of *enhancing* images, that is, of making them sharper, less blurred, more contrasty, in short, better adapted to the visual response: small contrast differences, such as steps, may need to be emphasized. The special feature of enhancement is that we do not enquire why the image quality is inadequate: we simply accept that there is something wrong and attempt to remedy matters, often on-line, in the case of much medical imagery for example and of such devices as the Synoptics "SysTEM".

The methods employed fall into several groups. The simplest and oldest methods consist of replacing each pixel-value of the image by a weighted sum of the pixel-value in question and those of its nearest neighbours.

If we simply take the average of all these pixel-values, we can expect to obtain a less noisy image but at the cost of some loss of resolution. Different patterns of weights allow us to accentuate small differences in particular directions or in all directions. All these choices are *linear* operations and have the form of a convolution between the original image and a discrete function, usually much smaller than the image, representing the pattern of weights. If we regard this pattern of weights as the weights of an invariant template, t, then *all* linear convolutional filters can be written as image–template convolutions of the form $a \oplus t$.

Nice though invariant templates are to understand and manipulate, the possibility of allowing t to be variant solves a perennial inconvenience of convolutional filters at the practical level: what is to be done near the edges of the image ? An edge pixel has only five neighbours, not eight and a corner pixel only three. These exceptional pixels can be included by making the template the same (invariant) for all pixels in the interior of the image but defining it differently when the target pixel lies on an edge or at a corner.

The next large group of methods of enhancing images is based on the operations of mathematical morphology [6, 12, 26, 39, 68]. These are all intrinsically nonlinear, though the basic operations can be regarded as a combination of a linear operation (convolution) and (non-linear) thresholding [42].

The basic morphological operations are *erosion* and *dilation* of an image by a *structuring element*, which may be chosen freely. All other morphological operations may be written in terms of these basic operations.

Dilation and Erosion

We now define a template t in such a way that it represents the structuring element. For this, we simply set

$$t_y(x) = \begin{cases} 0 & \text{if } x \in B'_y \\ -\infty & \text{otherwise} \end{cases} \tag{22}$$

The more general case in which the structuring element is not flat, or in other words, has "grey levels" is almost as simple. We then write

$$t_y(x) = \begin{cases} g(y-x) & \text{if } x \in B'_y \\ -\infty & \text{otherwise} \end{cases} \tag{23}$$

in which g denotes the grey level pattern of the structuring element. But what is B'_y, which appears in the definitions? B is simply the structuring element, B_y is the same structuring element translated through y and the prime indicates that B_y must be reflected about its origin. It is then easy to show that dilation is represented by

$$a \boxtimes t \; : \quad \text{dilation of } a \text{ by } B \tag{24}$$

and erosion by

$$a \boxtimes t^* \; : \quad \text{erosion of } a \text{ by } B \tag{25}$$

where t^* is the "additive conjugate" of t,

$$t^*_x(y) = \{t_y(x)\}^* = \begin{cases} -t_y(x) & \text{if this is finite} \\ -\infty & \text{if } t_y(x) = \infty \\ \infty & \text{if } t_y(x) = -\infty \end{cases} \tag{26}$$

The familiar combinations of dilation and erosion, opening and closing, are obvious extensions of these elementary operations.

In everyday morphology, the shape and grey-level pattern of the structuring element are invariant but adaptive morphology is also beginning to be studied. In fact, it should already be obvious that in the image algebraic representation, the difference is trivial: in one case, the template that represents the structuring element is invariant while in the other, it is not. An immense range of extensions becomes apparent, few of which have yet been explored in depth. The template may vary as a function of the target point in some predetermined way, as we have already seen in connection with convolutional filtering based on the operation $\oplus$. Again, it may vary with the target point not as a function of the location of the latter but guided by the grey-level at that point or by some property of the grey-levels in the neighbourhood of the target point. This second case, in which we use a parametrized template, was envisaged early on in the development of image algebra and is described in the first full presentation [51]; we have already met an example in vector coding. Davidson, to whom we owe the most sensitive explanation of the relation between image algebra and mathematical morphology [8, 9], emphasizes this point.

Another important aspect of the image processing operations associated with the operator $\boxtimes$ concerns the relation to matrix–vector and matrix–matrix products. Suppose that we map an image a into a vector $\nu(a)$ by simply placing the grey-levels in a single line, row after row, so that

$$\nu(a) = (a(x_1), a(x_2), ..., a(x_n)) \tag{27}$$

(Fig. 3). We likewise map a template t into a matrix $\Psi(t)$, each row representing the image associated with a particular pixel:

$$(\Psi(t))_{ij} = t_{y_j}(x_i) \tag{28}$$

Then it can be shown that

$$\begin{aligned} &\nu(a \boxtimes t) = \nu(a) \otimes \Psi(t) \\ &\nu(a \vee b) = \nu(a) \vee \nu(b) \\ &\Psi(t \boxtimes s) = \Psi(t) \otimes \Psi(s) \\ &\nu(t \vee s) = \Psi(t) \vee \Psi(s) \end{aligned} \tag{29}$$

in which a and b are images and s and t are templates. The matrix multiplication indicated by the symbol $\otimes$ does not, however, obey the familiar rule

$$\sum_j a_{ij} b_{jk} \text{ or } \sum_j a_{ij} b_j$$

but is defined by

$$\begin{aligned} \nu(a) \otimes \Psi(t) &= \bigvee_j \nu(a)_j + \Psi(t)_{jk} \\ \Psi(t) \otimes \Psi(s) &= \bigvee_j \Psi(t)_{ij} + \Psi(s)_{jk} \end{aligned} \tag{30}$$

This harmonization of the linear and nonlinear filters is a particularly impressive achievement of image algebra.

Despite the volume of the morphological literature, already vast, whole tracts of mathematical morphology remain unexplored or, at best, charted in a preliminary fashion. An important question concerns the morphological analogues of mean-squares filters, such as the Wiener filter. Considerable thought is being given to these, notably by Dougherty [13-16] and recently by Haralick *et al.* [27]. Here, we meet the notion of spectral decomposition, based not on linear operations as in the case of the Fourier transform but on nonlinear operations; a set of structuring elements then replaces the orthogonal sets of basis functions of the various linear transforms. So far, most of the analysis has been expresed in the formal set-theoretic language traditionally adopted in the morphological literature but past experience suggests that the image algebraic formulation will not only shed new light on what has already been achieved but also indicate fresh lines of investigation. Furthermore, the transition to grey-level images (and structuring elements) should be straightforward in image algebra (we note that the major part of Haralick *et al.* [27] is confined to binary images). For a good list of earlier (but still very recent) papers on these matters, see those by Cuciurean-Zapan and Dougherty [7] and Dougherty [14].

Another current research topic is the incorporation of the ideas of fuzzy set theory into image algebra and hence into such applications as morphology and neural network theory. We cannot go into detail here and refer to [10, 16, 62-64, 66], where recent thinking on these questions is to be found.

Morphological filters are not the only nonlinear methods of enhancing images. A very old procedure is still among the best for a certain class of images, those that are too light or too dark for the eye to be able to appreciate them. This is *histogram equalization* (or more generally, *modification*), whereby we first establish the histogram showing how many pixels of the image have each grey level, typically between 0 and 255. The eye finds images easiest to see when this histogram is reasonably broad and flat, the grey levels in the image being spread over the whole range from black to white. If, however, the grey levels are clustered in a narrow range, the eye will be unable to discern detail even if it is present. The remedy is simple and dramatically effective: the grey levels are redistributed by the computer so as to fill the whole range reasonably uniformly.

There are several ways of generating the grey-level histogram h of an image a. Let us denote the set of grey levels by $G = \{0, 1, ..., g\}$, where g is often 255. We define a template t such that

$$t_x^{(a)}(y) = \begin{cases} 1 & \text{if } y = a(x) \\ 0 & \text{otherwise} \end{cases} \tag{31}$$

This is another example of a parametrized template, since its values depend on a parameter, in this case the grey level of image a at x. Then $h = t^{(a)} \oplus 1$ will be the histogram of a, where 1 denotes a uniform image all of whose grey levels are unity; we see immediately that $h(k) = \sum_x t_x^{(a)}(k)$ is the number of pixels with the value $a(x) = k$. The histogram is then used to map the original grey levels into new grey levels according to the various schemes that have been proposed for this purpose [22, 45].

The final group of enhancement procedures that we mention here are generalizations of the *median filter*, which can often be relied on to reduce image noise without destroying fine detail,

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \Rightarrow \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{21} & a_{22} & a_{23} & a_{31} & a_{32} & a_{33} \end{pmatrix}$$

r	s
q	p

$\Downarrow$ ..... for a template 3 × 3

$$\begin{pmatrix} p\ q * s\ r \ldots\ldots \\ * p\ q * s\ r \ldots\ldots \\ * * p * * s \ldots\ldots \\ * * * p\ q * \ldots\ldots \\ * * * * p\ q \ldots\ldots \\ * * * * * p \ldots\ldots \\ \ldots\ldots \end{pmatrix}$$

Fig. 3. — Above: mapping an image into a row vector; below: mapping an invariant 3 × 3 template with origin at p into a matrix.

unlike the convolutional filters. The principle is very simple: the grey levels at the pixel in question and its immediate neighbours are written in ascending order and the grey level of the pixel being treated is replaced by the median (the fifth member of the list if there are nine members, Fig. 4). Generalized forms of this have been tested, in which the grey-level at the pixel in question is replaced by some other member of the list than the median (rank-order filters) or in which the grey levels of the neighbours are weighted or otherwise modified before selecting the median. Such filters can be represented either as an iterative sequence of max and min comparisons or by introducing a new operator, "median", which essentially selects a particular member of an (ordered) list [40, 41]. It has to be said that the attempt to represent median (and related) filters in image algebra has revealed, if not a weakness, at least a source of awkwardness. A reason for this is that operations such as max return a grey-level value, which is of course the highest value among the set of pixels to which the operator is applied, but do not tell us which pixel(s) had this highest value. In other words, the algebra lacks an operator based on max, for example, that returns a (binary) image indicating the site(s) of the pixel(s) attaining the maximum. Since we commonly apply max to a sequence of windows placed over a large image, this missing operator would in fact return not a (binary) image but a (binary) template. Various ways of remedying this situation are currently being examined [35]. We note in conclusion that we have enclosed the

word "binary" in parentheses above because two possibilities are being envisaged: either a binary image (or template) is generated, which simply signals the sites of pixels having the maximum value found by max, or a grey-level image (or template) with, for example, the current grey-level value at pixels having the maximum value and zero elsewhere. The relation between these alternatives is of course trivial but it is not yet clear which emerges more naturally.

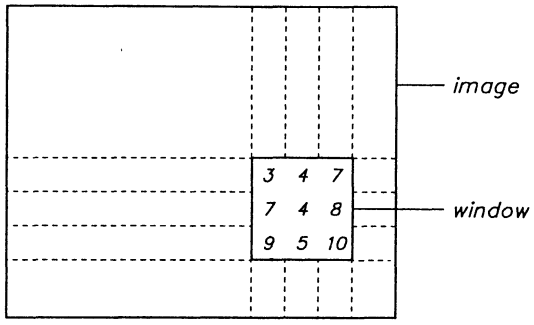


Fig. 4. — The action of a median filter on a section through an image. The grey-level value at each pixel is replaced by the median of the grey levels at the pixel and its nearest neighbours. In this example, the 4 is replaced by 7.

A very ingenious way of unifying many of these procedures - morphological operations, rank-order filters, convolutional filters - has been devised by Wilson [72], who introduces the notion of a "floating stack array". The idea is simple: a grey-level image in two dimensions (Wilson discusses the more general case of an n-dimensional image, thus including higher dimensional morphology for example) is mapped into a five-dimensional *binary* space, which is called a "floating stack array". Various sections of this array yield the different types of filter. The new dimensions introduced by the mapping are (1) threshold decomposition of the grey-level image, which is replaced by a stack of binary images; (2) the extent of a window, which will be used to characterize that of a convolutional filter or a structuring element; (3) the stack decomposition of the weights of the convolutional filter or the grey levels of a structuring element when the latter is not flat. We shall not reproduce Wilson's account here, but simply confirm that all the various types of filter mentioned above can be obtained by cutting appropriate sections through the new five-dimensional stack array.

4.3 RESTORATION. — Restoration of an image provides us with a more informative image or with information about its contents not directly visible by exploiting all available knowledge about the image-forming process. Common examples are three-dimensional reconstruction, Wiener filtering and the extraction of the phase and amplitude of the wavefunction from a focal series or from some other set of images of the same specimen. These are all easy to express in terms of image algebra. Saxton [53] has recently demonstrated that the techniques for extracting phase and amplitude from focal series that van Dyck and colleagues have been developing during the past few years, surveyed in [17], are in fact not different in essence from the early proposals of Schiske [54, 55]. It has to be said that, here too, the resemblance was obscured by the choice of notation and vocabulary. An instructive example of the benefit of expressing restoration procedures in terms of image algebra is the Gerchberg-Saxton algorithm [23, 24], which extracts the amplitude and phase of the wavefunction from the image and diffraction pattern of the same specimen area.

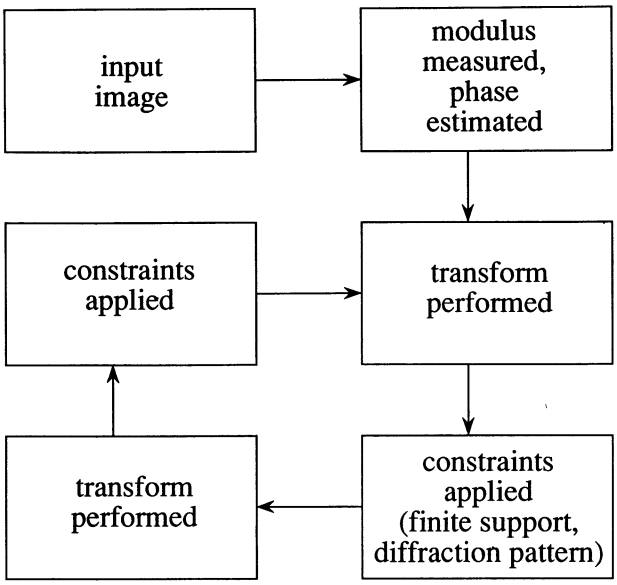


Fig. 5. — The phase problem. The sequence of iterations in the Gerchberg-Saxton and related (Fienup) procedures.

We are not concerned here with the practical aspects of this algorithm and the problems associated with it; for discussion of these, see Saxton [52] or section 74.3 of [36].

The algorithm is shown schematically in Figure 5. The complex-valued wavefunction at the image, of which only the modulus can be measured, is denoted by i and the complex-valued wavefunction at the diffraction pattern by d. We regard each of these as a two-valued image, $i = (i_a, i_\varphi)$ and $d = (d_a, d_\varphi)$ in which i_a and d_a are the (known) moduli and i_φ and d_φ are the (unknown) phases. It is helpful to introduce $m_i = (i_a, 0)$, $m_d = (d_a, 0)$. We also need an image with unit phase, $e = (0, 1)$.

The first estimate $i^{(1)}$ of the true image wavefunction i is then

$$i^{(1)} = (i_a, i_\varphi^{(1)}) \tag{32a}$$

in which $i_\varphi^{(1)}$ is estimated or guessed. The first estimate of the diffraction pattern wavefunction is then

$$d^{(1)} = tFi^{(1)} + m_d \tag{32b}$$

in which F, F^{-1} denote the Fourier transform and its inverse. This leads to

$$i^{(2)} = tF^{-1}d^{(1)} + m_i \tag{32c}$$

and in general

$$\begin{aligned} d^{(n)} &= tFi^{(n)} + m_d \\ i^{(n+1)} &= tF^{-1}d^{(n)} + m_i \end{aligned} \tag{33}$$

This example [28, 29] illustrates strikingly our earlier observation to the effect that image algebraic representation often suggests new approaches or promising extensions of the algorithm in question. For the above sequence, we are immediately tempted to enquire whether some other

form of t might be interesting and whether some other operation than the Fourier transform might be introduced. The former includes an early suggestion of Gassmann [21] and the latter points in the direction of Fienup's extensions [18, 19]. For discussion of some other aspects of restoration, see [29].

4.4 ANALYSIS. — In electron microscopy, and indeed in microscopy in general, image analysis has been largely confined to mathematical morphology. We have seen that the two basic operations, dilation and erosion, are extremely simply expressed in image algebra and all the more elaborate morphological procedures can be written in terms of these. Most of the processors that are offered commercially for use with light microscopes, scanning electron microscopes and transmission microscopes routinely offer the morphological operations.

This is not the only aspect of image analysis, however, and it seems very likely that others will come to be used in the study of electron images and diffraction patterns. Structural description, for example, and measurements of area, perimeter and similar geometric dimensions, estimation of moments and location of the centroids of individual features are all of interest. The study of these in image algebra has been pursued for some years and it is clear that there is no obstacle in principle to the inclusion of all these analytical operations in image algebra. There is, however, so far no definitive account of them and we refer the reader to the work of Shi and Ritter [50, 56-61].

Pattern recognition and classification also form part of image analysis. Here too, the use of image algebra is in its infancy but we draw attention to the fact that the multivariate techniques employed in the preliminary stages of three-dimensional reconstruction, introduced by Frank and van Heel [20, 37], and in elemental mapping by EELS [2] can be straightforwardly expressed in the vocabulary and notation of this algebra [31]. See also [39, 46, 68, 69].

5. Concluding Remarks

In this introductory account of image algebra, only the aspects that are of immediate interest for electron image processing have been discussed since our purpose is not to reproduce the formal presentations of Ritter and colleagues but to draw attention to the usefulness of this approach. In particular, we wish to emphasize that although the algebra began as a means of codifying a mass of disparate material, one of its principal attractions is that new variants on old methods often emerge almost effortlessly. Furthermore, connections between topics that had seemed unrelated may be rendered obvious. We are confident that image algebra has an important part to play in the development and extension of electron image processing.

One last remark: the best way of appreciating the benefits and attractions of image algebra is not to read an account such as this but to try it out for yourself. Versions of ADA, C and FORTRAN that support the needs of image algebra are available (see for example [1, 4, 5, 43, 61, 65, 67, 70, 71, 73], which cover both hardware and software questions).

References

[1] Belisle C.M. and Horner P.A., Image algebra algorithm development environment, *Proc. SPIE* **1769** (1992) 154-165.

[2] Bonnet N., Simova E., Lebonvallet S. and Kaplan H., New applications of multivariate statistical analysis in spectroscopy and microscopy, *Ultramicroscopy* **40** (1992) 1-11.

[3] Clarke R.J., Transform Coding of Images (Academic Press, London & Orlando, 1985).

[4] Coffield P.C., An architecture for processing image algebra operations, *Proc. SPIE* **1769** (1992) 178-189.
[5] Coffield P.C. and Scudiere M.B., A prototype coprocessor for image algebra operations, *Proc. SPIE* **2030** (1993) 327-333.
[6] Coster M. and Chermant J.-L., Précis d'Analyse d'Images (Editions du CNRS, Paris, 1985).
[7] Cuciurean-Zapan C. and Dougherty E.R., Optimal openings for overlapping signal and noise grains, *Proc. SPIE* **2300** (1994) 46-56.
[8] Davidson J.L., Foundation and applications of lattice transforms in image processing, *Adv. Electron. Electron Phys.* **84** (1992) 61-130.
[9] Davidson J.L., Lattice transforms in image processing, *CVGIP: Image Understanding* **57** (1993) 283-306.
[10] Davidson J.L. and Hummer F., Morphology neural networks: an introduction with applications, *Circuits Syst. Signal Proc.* **12** (1993) 177-210.
[11] Dougherty E.R., A homogeneous unification of image algebra, I and II, *J. Imaging Sci.* **33** (1989) 136-143 and 144-149.
[12] Dougherty E.R., An Introduction to Morphological Image Processing (SPIE Optical Engineering Press, Bellingham WA 1992).
[13] Dougherty E.R., Optimal mean-square N-observation digital morphological filters. Part I, Optimal binary filters; Part II, Optimal gray-scale filters, *CVGIP: Image Understanding* **55** (1992) 36-54 and 55-72.
[14] Dougherty E.R., Optimal granulometric-induced filters, *Proc. SPIE* **2300** (1994) 355-366.
[15] Dougherty E.R. and Loce R.P., Efficient design strategies for the optimal binary digital morphological filter: probabilities, constraints and structuring-element libraries. In Mathematical Morphology in Image Processing (E.R. Dougherty Ed.) (Marcel Dekker, New York, Basel & Hong Kong, 1993) 43-92.
[16] Dougherty E.R. and Sinha D., Fuzzy morphology, *Adv. Imaging Electron Phys.* (1996).
[17] van Dyck D., Phase retrieval from electron images, *Adv. Imaging Electron Phys.* (1996).
[18] Fienup J.R., Phase retrieval algorithms: a comparison, *Appl. Opt.* **21** (1982) 2758-2769.
[19] Fienup J.R., Comparison of phase retrieval algorithms, *Adv. Comput. Vision Image Proc.* **1** (1984) 191-225.
[20] Frank J. and van Heel M., Correspondence analysis of aligned images of biological particles, *J. Mol. Biol.* **161** (1982) 134-137.
[21] Gassmann J., Optimal iterative phase retrieval from image and diffraction intensities, *Optik* **48** (1977) 347-356.
[22] Gauch J.M., Investigations on image contrast space defined by variations on histogram equalization, *CVGIP: Graph. Models Im. Proc.* **54** (1992) 269-280.
[23] Gerchberg R.W. and Saxton W.O., A practical algorithm for the determination of phase from image and diffraction plane pictures, *Optik* **35** (1972) 237-246.
[24] Gerchberg R.W. and Saxton W.O., Wave phase from image and diffraction plane pictures, In Image Processing and Computer-aided Design in Electron Optics (P.W. Hawkes Ed.) (Academic Press, London & New York, 1973) 66-81.
[25] Giardina C.R., The universal imaging algebra, *Adv. Electron. Electron Phys.* **67** (1986) 121-182.
[26] Haralick R.M., Sternberg S.R. and Zhuang X.-h., Image analysis using mathematical morphology, *IEEE Trans.* **PAMI-9** (1987) 532-550.
[27] Haralick R.M., Katz P.L. and Dougherty E.R., Model-based morphology: the opening spectrum, *Graph. Models Image Proc.* **57** (1995) 1-12.
[28] Hawkes P.W., A comment on the image algebraic representation of iterative solutions of the phase problem, *J. Optics (Paris)* **22** (1991) 219-222.
[29] Hawkes P.W., Image algebra and restoration, *Scanning Microsc. Suppl.* **6** (1992) 179-184.
[30] Hawkes P.W., Manipulation of multivalued images in image algebra, *J. Math. Imaging Vision* **2** (1992) 83-85.
[31] Hawkes P.W., Reflections on the algebraic manipulation of sets of electron images or spectra, *Optik* **93** (1993) 149-154.
[32] Hawkes P.W., The TEM forms images, the STEM forms templates, Proc. 13th Int. Cong. Electron Microscopy Vol. 1 (Editions de Physique, Les Ulis 1994) 495-496.
[33] Hawkes P.W., The STEM forms templates, *Optik* **98** (1995) 81-84.
[34] Hawkes P.W., Vector quantization, an image-template mapping, *J. Math. Imaging Vision* **5** (1995).
[35] Hawkes P.W., Algebraic representation of median and related nonlinear filters. In *EMAG* 95 (Institute of Physics Publishing, Bristol & Philadelphia, 1995/6).

[36] Hawkes P.W. and Kasper E., Principles of Electron Optics, Vol. 3 (Academic Press, London & San Diego, 1994).

[37] van Heel M. and Frank J., Use of multivariate statistics in analysing the images of biological macromolecules, *Ultramicroscopy* **6** (1981) 187-194.

[38] Huang K.S., Jenkins B.K. and Sawchuk A.A., Binary image algebra and optical cellular logic processor design, *Computer Vision Graph. Im. Proc.* **45** (1989) 295-345.

[39] Jeulin D., Mathematical morphology and materials image analysis, *Scanning Microsc. Suppl.* **2** (1988) 165-183.

[40] Maragos P. and Schafer R.W., Morphological filters - Part I: Their set-theoretic analysis and relations to linear shift-invariant filters; - Part II: Their relations to median, order-statistic, and stack filters, *IEEE Trans.* **ASSP-35** (1987) 1153-1169 and 1170-1184; correction *ibid.* **ASSP-37** (1989) 597.

[41] Maragos P. and Schafer R.W., Morphological systems for multidimensional signal processing, *Proc. IEEE* **78** (1990) 690-710.

[42] Mazille J.E., Mathematical morphology and convolutions, *J. Microscopy* **156** (1989) 3-13.

[43] Murillo J.J. and Wilson J.N., An Ada interpretative system for image algebra, *Proc. SPIE* **1769** (1992) 166-177.

[44] Netravali A.N. and Haskell B.G., Digital Pictures, Representation and Compression (Plenum, New York & London, 1988).

[45] Paranjape R.B., Morrow W.M. and Rangayyan R.M., Adaptive-neighborhood histogram equalization for image enhancement, *CVGIP: Graph. Models Im. Proc.* **54** (1992) 259-267.

[46] Prod'homme M., Coster M., Chermant L. and Chermant J.-L., Morphological filtering and granulometric analysis on scanning electron micrographs: applications in materials science, *Scanning Microsc. Suppl.* **6** (1992) 255-268.

[47] Ritter G.X., Recent developments in image algebra, *Adv. Electron. Electron Phys.* **80** (1991) 243-308.

[48] Ritter G.X. and Gader P.D., Image algebra techniques for parallel image processing, *J. Parallel Distrib. Comput.* **4** (1987) 7-44.

[49] Ritter G.X., Wilson J.N. and Davidson J.L., Data compression of multispectral images, *Proc. SPIE* **829** (1987) 58-64.

[50] Ritter G.X., Wilson J.N. and Davidson J.L., Image algebra application to image measurement and feature extraction, *Proc. SPIE* **1076** (1989) 146-153.

[51] Ritter G.X., Wilson J.N. and Davidson J.L., Image algebra: an overview, *Computer Vision Graph. Im. Proc.* **49** (1990) 297-331.

[52] Saxton W.O., Computer Techniques for Image Processing in Electron Microscopy, *Adv. Electron. Electron Phys. Suppl.* **10** (Academic Press, New York & London, 1978).

[53] Saxton W.O., What is the focus variation method? Is it new? Is it direct? *Ultramicroscopy* **55** (1994) 171-181.

[54] Schiske P., Zur Frage der Bildrekonstruktion durch Fokusreihen. In Electron Microscopy 1968 (D.S. Bocciarelli Ed.) Vol. I (Tipografia Poliglotta Vaticana, Rome 1968) pp. 145-146.

[55] Schiske P., Image processing using additional statistical information about the object. In Image Processing and Computer-aided design in Electron Optics (P.W. Hawkes Ed.) (Academic Press, London & New York, 1973) pp. 82-90.

[56] Shi H. and Ritter G.X., Structural description of images using image algebra, *Proc. SPIE* **1769** (1992) 368-375.

[57] Shi H. and Ritter G.X., Image component labeling using local operators, *Proc. SPIE* **2030** (1993) 303-314.

[58] Shi H. and Ritter G.X., A special class of nonlocal image-template operations, *Proc. SPIE* **2300** (1994) 204-212.

[59] Shi H. and Ritter G.X., A fast algorithm for image component labeling with local operators on mesh connected computers, *J. Parallel Distrib. Comput.* **23** (1994) 455-461.

[60] Shi H. and Ritter G.X., A new parallel binary image shrinking algorithm, *IEEE Trans. Image Proc.* **4** (1995) 224-226.

[61] Shi H., Ritter G.X. and Wilson J.N., Parallel image processing with image algebra on SIMD mesh-connected computers, *Adv. Imaging Electron Phys.* **90** (1994) 353-431.

[62] Sinha D. and Dougherty E.R., Fuzzy mathematical morphology, *J. Visual Commun. Image Represent.* **3** (1992) 286-302.

[63] Sinha P., Sinha D. and Dougherty E.R., Shape detection via fuzzy morphology, *Proc. SPIE* **1904** (1993) 172-182.

[64] Sinha D., Sinha P. and Dougherty E.R., Algorithm development for fuzzy mathematical morphology, *Proc. SPIE* **2030** (1993) 220-230.
[65] Software User Manual for the iac++ Class Library (1994); available by anonymous ftp from the University of Florida Center for Computer Vision and Visualization using ftp.cis.ufl.edu, in pub/ia/documents.
[66] Srivastava R. and Davidson J.L., Fuzzy image algebra neural networks for target classification, *Proc. SPIE* **2300** (1994) 145-156.
[67] Weeks A.R., Myler H.R. and Cinci L.D., An interpretive programming language for image algebra, *Proc. SPIE* **2300** (1994) 180-191.
[68] Tovey N.K., Dent D.L., Corbett W.M. and Krinsley D.H., Processing multi- spectral scanning electron microscopy images for quantitative microfabric analysis, *Scanning Microsc. Suppl.* **6** (1992) 269-282.
[69] Tovey N.K., Hounslow M.W. and Wan J., Orientation analysis and its applications in image analysis, *Adv. Imaging Electron Phys.* **93** (1995) 219- 329.
[70] Wilson J.N., An introduction to image algebra Ada. *Proc. SPIE* **1568** (1991) 101-112.
[71] Wilson J.N., Supporting image algebra in the C++ language, *Proc SPIE* **2030** (1993) 315-326.
[72] Wilson S.S., Floating stack arrays: a unified representation of linear and morphological filters, *Proc. SPIE* **1769** (1992) 332-343.
[73] Yoder M.F., Images and image domains in image algebra Ada. *Proc. SPIE* **1350** (1990) 274-282.