UvA-DARE (Digital Academic Repository)

DIODA: delineation and feature extraction of microscopical objects

Vossepoel, A.M.; Smeulders, A.W.M.; van den Broek, K.

DOI

[10.1016/0010-468X\(79\)90072-2](https://doi.org/10.1016/0010-468X(79)90072-2)

Publication date

1979

Published in

Computer Programs in Biomedicine

[Link to publication](#)

Citation for published version (APA):

Vossepoel, A. M., Smeulders, A. W. M., & van den Broek, K. (1979). DIODA: delineation and feature extraction of microscopical objects. *Computer Programs in Biomedicine*, 10, 231-244. [https://doi.org/10.1016/0010-468X\(79\)90072-2](https://doi.org/10.1016/0010-468X(79)90072-2)

General rights

It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations

If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: <https://uba.uva.nl/en/contact>, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.

DIODA: DELINEATION AND FEATURE EXTRACTION OF MICROSCOPICAL OBJECTS

Albert M. VOSSEPOEL, Arnold W.M. SMEULDERS * and Klaas VAN DEN BROEK **

Dept. of Biomedical Information Processing, * Laboratory for Pathology and ** Department of Histo- and Cytochemistry, Faculty of Medicine, Wassenaarseweg 62, 2333 AL Leiden, The Netherlands

A computer program is described for delineation and measurement of microscopical objects, such as cells and chromosomes, which may have been scanned using absorbance, fluorescence or reflectance microscopy. The quality of the object delineation is optimized through the contour ratio, which is simply computed from the object contour. Geometrical features, like the perimeter and area are computed. A new definition is introduced for the background region, especially suited for the analysis of closely packed objects. This definition is based upon a comparison between total staining material content values resulting from a variety of methods and circumstances. The program is principally intended for use in the on-line real-time environment of a small laboratory computer, but may be used off-line as well. It has facilities for displaying the results and the process by which they are produced. A program module is described for displaying scan data on a bilevel display using an adaptation of the sigma-delta method.

Background level
Object delineation

Cytology automation
Quantitative microscopy

Image processing
Scene segmentation

Integrated optical density
Staining material content

1. Introduction

Quantitative measurements of the amount of staining material in microscopical objects have become of increasing importance in the field of histochemistry and cytochemistry. The objects may be stained by a variety of methods, binding the staining material to cells, cell nuclei or cellular components, such as chromosomes and proteins. Apart from flow measurements, the methods by which the distribution of staining material over the object is measured can be divided in absorbance, fluorescence and reflectance scanning microphotometry. The image has to be sampled at high resolution ($\leq 0.5 \mu\text{m}$) in order to avoid severe errors in the measurement caused by the optical system [1]. For this reason, or if the original object is perishable, an intermediary photographic negative enlargement is sometimes used [2]. The sampling may be performed by a microscope stage scanning system [2], or by a flying spot scanner [3].

In theory, the amount A of staining material present at the position of the measuring spot in the object plane is a function f of the intensity $I_{x,y}$ of the light measured at the spot, and the intensity (I_B) of the

light measured when no object would be present, i.e., in the background:

$$A = f(I_{x,y}, I_B)$$

The total amount T of staining material — which is referred to further as 'the content' — is then found by taking the sum of all values of A that belong to the object. From this general approach it is clear that in the determination of T it is essential to know which values in the data array belong to the object and which ones do not. The proper location of the background region and the accurate determination of I_B is also of great importance. This can be illustrated by the case of absorbance cytophotometry, most commonly used to determine the content. In this case, A is proportional to the optical density, provided a number of linearity restrictions:

$$A = C \cdot (\log(I_B) - \log(I_{x,y}))$$

In which C represents the proportionality constant. This implies that any (stochastic) error in I_B results in a systematic error in A . Therefore, attention has to be paid to the proper determination of I_B , since the error in I_B has a n -fold influence on the value of T , if

n is the number of object points. For other methods of determining T , like microfluorometry, other functions have to be taken for f [4].

In a previous communication, a data acquisition system for stage scanning (HIDACSYS) is described [5]. This system contains three different programs which are representative of current approaches for the computation of the content.

The first program, FASTSCAN, does not perform any object delineation. The background is computed per line and defined as the average of the first and last few — say three — measurements of the scan line, i.e., from two strips at the sides of the scanned area, beside the object. The content of each scan line is corrected for its background value thus computed, and added to obtain the total content. Obviously, this program takes up little of a computer's facilities, but it is restricted to cases in which there is just one single object present in the scanned area, surrounded by a background region of sufficient width, geometrical regularity and homogeneity. Any part of another object, or even noise, present in the scanned area will falsely contribute to the computed content. This contribution may be negative as well, if another object is even partly situated in one of the background strips. In order to be aware of these conditions, visual display of the scanned image is always necessary in this approach.

The second program, ARRAYSCAN, does perform object delineation, but does not discriminate among objects in the scanned array. The program computes a delineation level for the distinction between object and background points from the background peak in the histogram (for details, see below). Thresholding of the image with this level leads to object boundaries independent of the over-all light intensity. However, contributions of other objects may still be substantial unless the data array is edited manually by means of a light-pen or cursor. The background value that has to be subtracted from the data may also be set manually in the data histogram, preferably at the histogram peak. An alternative for the background value is the average of all points below a threshold computed from location and width of the histogram peak. In this approach, ARRAYSCAN is quite similar to, e.g., the program SCANCANS [6].

The aim of this communication is to describe a program that automatically computes an accurate and

reproducible value of the total amount of staining material (or content) of a microscopical object. This program, DIODA, does not strictly need display facilities to see whether other objects are present in the scanned area, or whether the background region is small, irregular or inhomogeneous. In most of these cases, the program does not require interaction. This entails that the program should perform two functions automatically that are performed otherwise in the programs mentioned above:

- delineation of the object area;
- determination of the region that provides the background value, to be used in the computation of the total content.

Several methods are known for the determination of the object boundary as assessed visually. For a recent overview see [7] or [8]. The oldest and most commonly used method is thresholding of the image with a distinct grey-level, usually derived from the data histogram. This method is simple but does not take into account local information. The second group of methods consists of thresholding in the gradient transformed image. These methods are better proof against global variations of the background intensity. However, a high resolution is usually needed to reduce the influence of high frequency noise. Other methods cluster the data according to their local similarity. Though the method may seem attractive, the setting of the cluster algorithm is rather problem dependent. Finally other techniques for image segmentation use local information derived from a former trial to obtain a better segmentation afterwards (relaxation).

In DIODA the approach is to restrict the required computing time by using a simple method, namely grey-level thresholding and adapting the result according to a simple dimensionless quality measure, e.g., the contour ratio, computed from preliminary trials. The contour ratio may be defined as the largest (circular) area that can be enclosed by the perimeter, divided by the actual object area, i.e., the perimeter squared, divided by 4π times the object area. The microscopical objects considered visually show smooth-edged contours, so minimizing the contour ratio may be assumed equivalent to optimizing the contour quality.

It should be emphasized that the region from which the background value is derived is not per se

the region just outside the boundary as assessed visually or by other means. Therefore the histogram peak is commonly used to define the background value. A more precise measure for this value is the average of all values below a threshold, avoiding ambiguity in defining the histogram peak. Local background regions just outside extended objects have a better performance [9,10] but may be influenced by the presence of other objects. By analogy, the object may be taken as a mountain surrounded by a more or less hilly country with other mountains, with altitude corresponding to grey-value. The mountains may even be connected by ridges at an altitude higher than the surrounding hills. On a contour map, the mountain can only be isolated by taking a contour at an altitude higher than that of the ridge. This contour should then be representative of the size and shape of the mountain on the map. If we want to find a surrounding region with low altitude (the background), we first have to go down as far as possible from the delineating contour.

The process is analogous to the propagation of a wave-front, locally arrested by every raise in altitude encountered. It ends in the surrounding valleys and basins, most probably situated in the hilly country of the background (unless the mountain would be surrounded by a closed wall at a higher altitude, but below the first contour). The result of the monotonically decreasing (or non-increasing) propagation process is called the fall-set [11], and the background region is assumed to consist of the picture elements directly adjacent to this fall-set, under certain restrictions (for details, see below).

2. Data

A first general requirement for input data is sufficient precision. In any case, the image data are the result of an analogue-to-digital (A/D) conversion. If n is the number of bits produced by the A/D-converter, the maximum individual digitization error is $2^{(1-n)}$, reflecting the uncertainty in the last digit at mid-range. This implies that at least 8 bits are required to keep the individual digitization error below 1%. In most computers 8 bits represent exactly 1 byte, making it attractive to use this data representation for input. However, the advantage of smaller

storage requirements does not offset the loss in precision sometimes suffered from necessary conversions. Besides, DIODA uses the sign bit as a marker, reducing the number of available bits in a byte to 7, by which the maximum individual digitization error increases to $\sim 1.5\%$. It has turned out that under current conditions the expected mean error in the content happens to lie in the same range, i.e., 1% for 8-bit, 1.5% for 7-bit and 3% for 6-bit data [10]. Consequently, the full 16-bit word is used. A second requirement, imposed by DIODA, is that the background values are lower than those in the object. A third requirement is that the data values should be additive quantities, because in the end we have to subtract an average background value from each object value before adding the latter to the total content.

These requirements are not always met by the input data representing an image that may have been scanned in different modes:

- absorbance, fluorescence or reflectance microscopy;
 - original object or photographic negative;
 - light intensity or optical density values measured.
- In principle, this would result in $3 \times 2 \times 2 = 12$ different modes, but this number may be reduced to 4 for the purpose at hand by putting reflectance equivalent to fluorescence, fluorescence negatives equivalent to absorbance originals, and fluorescence originals to absorbance negatives. The conversions between individual values of light intensity I and optical density D (without background correction) are:

$$D = C \cdot \log(I_{\max}/I) \quad (1)$$

$$I = I_{\max} \cdot \exp(-D/C) \quad (2)$$

In which the constant $I_{\max}$ represents the maximum value of I , and C is an arbitrary scaling constant, which has the value $1000/\log(10)$ in the program. The conversion between an original image (suffix o) and a photographic negative (suffix n) are:

$$D_o = (D_{n,\max} - D_n)/\gamma \quad (3)$$

$$D_n = D_{n,\max} - \gamma \cdot D_o \quad (4)$$

In which the constant γ represents the slope of the straight part of the Hurter-Driffeld curve, and $D_{n,\max}$ represents the maximum density of the

photographic material used. The latter constant ensures that the results of (3) and (4) are always positive, whereas the former accounts for the gradation or γ of the negative. These conversions can be used to meet the second and third requirements, that restrict the input to absorbance original density data or to fluorescence negative intensity data.

The use of optical density values is only allowed if the conversion (1) is performed before digitizing, because otherwise the distribution of density values may be severely distorted. Applying this conversion after digitizing, (1) gives for the derivative of D with respect to I the value $-C/I$. In order to prevent loss of precision by converting the result back to integer, the absolute value of this derivative should be greater than unity for every value of I . Consequently C should be greater than $I_{\max}$, to ensure that the smallest gap between two adjacent values of D is at least unity. However, the largest gap between two adjacent values of D will then become $C \cdot \log(2)$, corresponding with the difference between $I = 1$ and $I = 2$. If, e.g., the input is digitized in 8 bits, this largest gap represents 12.5% of the full range of D -values, and half of the full range of D -values is used by 6.25% of the full range of I -values. The distortion of the resulting density value distribution appears for low I -values, i.e., in the dark parts of the scanned image, through noise amplification and many empty classes in the histogram. On the other hand, the differences between the high I -values partly vanish, which is not desirable if operations of differential type are to be applied. The electronics of the data acquisition system in use do not provide a logarithmic signal, so the only feasible input mode for us consists of intensity values. This implies that, e.g., with absorbance images of original slides, a conversion is necessary to meet the second requirement, i.e., to have background values lower than the object ones. Using for this purpose one or more of the conversions introduced so far would severely distort the distribution of data values, as illustrated above for the example of conversion (1). So the whole array of original intensity values I_o is converted into 'pseudo-negative' intensity values I'_n instead. This conversion is performed upon entering DIODA through:

$$I'_n = I_{o,\min} + I_{o,\max} - I_o \quad (5)$$

In which $I_{o,\min}$ and $I_{o,\max}$ are the lowest and highest

values of I_o , respectively. This conversion serves no other end than to make the background values less than the object ones, while conserving the noise characteristics of the image. Note that this conversion is computationally the same as (3) or (4) with $\gamma = 1$. When the corresponding optical density value is needed, an individual value I'_n is first converted back to I_o before conversion (1) is applied.

Summarizing, the necessary conversions to the additive quantity, i.e., density D_o for absorbance images, are:

- from pseudo-negative intensity I'_n through (5) and (1);
- from intensity I_o through (1);
- from negative intensity I_n through (1) and (3);
- from negative density D_n through (3).

Likewise, the necessary conversions to intensity I_o for fluorescence images are:

- from pseudo-intensity I'_o through (5), (1), (4) and (2);
- from negative intensity I_n through (1), (4) and (2);
- from negative density D_n through (4) and (2);
- from density D_o through (2).

Actually, these conversions are all performed by the function LOGTAB, which may also be applied for most other conversions by giving the input data type an alias. LOGTAB is initialized for every image with the minimum and maximum data values. In order to save computing time, the conversion (1), involving the logarithmic function, is performed by a table look-up. The table is created once at the first entrance of LOGTAB. Conversion (2), involving the exponent function, is not tabulated, simply because it is used less frequently.

So far, we have only been concerned with the data type, but another important aspect is their structure, i.e., the way in which the values associated with the

Fig. 1. Direction codes used to designate vector elements (vexels).

picture elements (pixels) are ordered. In DIODA, we have chosen a one-dimensional structure, as a result of weighing economy of pixel co-ordinate transfer against conceptual simplicity. Most frequently, pixel co-ordinates are used in relation to neighbours, of which each pixel has eight unless it lies on the array boundary. The neighbours are reached via vector elements (sometimes called *vexels*) with direction codes 1–8, designated in fig. 1. Another name for these codes is Freeman-codes [12].

Using a one-dimensional pixel array, we have tried to make a virtue of need by accomodating the process of reaching a neighbour in a separate subroutine called NEXT. This routine converts the original pixel co-ordinate into that of a neighbour using the specified direction code information. If an outward direction is specified originating at a pixel on the boundary of the data array, the output direction code is changed to the next higher one — i.e., rotated counterclockwise — that indicates an existing pixel. Consequently, a neighbour is always found, though sometimes in another direction than specified initially.

3. Program

The program DIODA has a modular structure, each module performing a specific task. The function of DIODA itself is just to provide an interface for calling and to provide inter-module facilities for reactions on unwanted events. The functions provided by the modules are:

- delineation level computation from the histogram (HISPAR);
- location of a starting point for the contour (STAPIX);
- location of the object boundary (CONTUR);
- location of the area within that boundary (MARKER);
- propagation of the fall-set (EXTEND);
- determination of the background value (BACKGR).

The procedures LOGTAB and NEXT, already mentioned in the previous section, are used throughout these modules.

In DIODA, the object is delineated by thresholding at a distinct level. The value of the delineation level involved in this process not only determines the

size and shape of the object found, but even whether or not an object is found. The level is computed by the subroutine HISPAR, which determines the minimum and maximum values of the pixel array, and uses them to initialize the data conversion function LOGTAB. If necessary, it inverts the values of the data array in order to provide object values that are higher than the values of the background region: with absorbance data from intensity to pseudo-negative intensity values or from negative density to density values. With fluorescence data the same inversions are performed interchanging negative and original. A histogram is then made of the pixel values. In order to prevent low or zero class occupations, the histogram classes are merged such that the resulting average class occupation approximates the resulting number of classes. Actually, this merging procedure is simulated by boxcar-averaging in order to prevent difficulties arising from the arbitrary location of the widely spaced class boundaries.

In the averaged histogram, the highest peak is assumed to result from the background pixels. This might not be true for closely packed objects in tissue sections. In such cases a user-specified level, set manually or computed elsewhere, may be supplied for object delineation. But then probably more sophisticated methods for scene segmentation are demanded, because the use of a simple threshold is not sufficient.

Using the routine HISPAR however, the classes with a frequency nearest to a specified fraction (actually 0.4) of the background peak frequency are detected, starting at the peak itself in both directions. The distance between the lower class centre and the upper class centre is then added to the latter, resulting in the initial value of the delineation level (cf. fig. 2). The distance involved, i.e., the width of the background peak at the specified fraction of the peak frequency, is roughly proportional to the background noise. This parameter is referred to as the 'background spread' in the subroutine BACKGR. In order to prevent a too high level in noisy pictures, the delineation level is forced by HISPAR not to exceed the value half-way between that of the centre of the upper class at the background peak slope and the highest pixel value present (cf. fig. 2).

Now that the delineation level between object and background has been found, the starting point of the contour is searched. A pixel is considered as a possi-

Fig. 2. Derivation of the delineation level L from the pixel value histogram (schematically drawn). The background peak P is intersected at 0.4 times the height H_p at the points Q and R . The difference between the pixel values at Q and R (S_B) is added to the value at R to find the delineation level L . The highest permissible value of L ($L_{\max}$) is halfway between the value of R and the maximum pixel value.

ble starting point for the object contour if and only if:

- it has a value greater than the delineation level;
- it has at least one neighbour with a value greater than that level (isolated pixels are not considered to be objects);
- it has at least one neighbour with a value less than that level, which is also not isolated itself.

The search for a starting pixel that fulfills these conditions is performed by the subroutine STAPIX. The search may be from any pixel within the array in any orthogonal direction specified by the user. By default however, the search is in all four orthogonal directions starting from the centre of the scanned array with direction code 2. This implies that not only the data array should contain the object completely, but also that any object besides the one encountered first is not taken into consideration.

In STAPIX the starting pixel is only searched for in the even coded directions, i.e., excluding diagonal directions, thereby preventing deadlocks at the completion of the contour later on (cf. fig. 1 and 3). In one of these directions, first a pixel with a value not less than the delineation level is searched. Once found, the neighbours in the even-coded directions are checked on the presence of at least one with a value also not less than the delineation level. Otherwise the search is simply continued.

Now that a candidate object pixel has been found, the search is continued in the same directions as before, but now for the object boundary, in this case a pixel with a value less than the delineation level. Once found, this pixel is also checked on not being isolated. Otherwise the search for a pixel with value less than the level is continued in the same direction as before. If during the search the array boundary is

Fig. 3. Example of failure to close the contour at pixel (S) that has been searched for along the broken line. The numbers next to the long arrows indicate the order in which the contour direction codes are found. The short arrows indicate the direction in which the neighbour examination did not yield a new contour pixel. Consequently S may only be searched for in orthogonal directions. (Shading indicates object for BBQ-contour, and background for OBQ-contour.)

reached, the search starts again from the beginning at the central pixel in the next even-coded direction. If the search in all four directions is not completed successfully an event parameter signalizes this condition.

Beginning at the starting pixel produced by STAPIX, a contour can be made consisting of the pixels with a value not less than the delineation level with at least one neighbour with a value less than this level. The first candidate pixel for the contour is the neighbour in the direction with the code that was used last in STAPIX plus one. If this pixel has a value less than the delineation level, the direction code is incremented by one modulo eight, and the neighbour in the new direction is examined. The process is repeated until the examination reveals a pixel value not less than the delineation level. The existence of such a pixel is warranted by the definition of the starting pixel, which cannot be isolated. Now the pixel examined takes over the part of the starting pixel, and the first neighbour that will be examined is in the direction with code l having the following relation to the last code (k):

$$l = (\text{mod}((k - 1 - \text{mod}(k, 2)), 8)).$$

This relation will limit multiple examination of the same pixel to a minimum and still guarantee the completion of the contour. For every contour pixel found in this way, the direction code k is stored in a buffer. In case of contour buffer overflow, and touching of the array boundary (signalled by a direction code

change in the subroutine NEXT) the event flag is set appropriately. When the pixel index equals that of the starting pixel, the contour is closed. This implies that the starting pixel should always be attainable by the examination procedure, which condition is met only if an even direction code has been used in STAPIX (cf. fig. 3).

The contour construction procedure described so far is called (8-connected) Object Boundary Quantization (OBQ) [13]. The alternative procedure, called Background Boundary Quantization (BBQ), starts at the neighbour with a value less than the level of the OBQ starting pixel. The first candidate pixel for the BBQ contour is then in the direction that was last used in STAPIX plus 5. The candidate is accepted if it has a value less than the level; apart from that the procedure to be followed is the same as described above for the OBQ contour. This results in a clockwise turning construction of the BBQ contour, whereas the OBQ contour is constructed in the opposite direction. The advantage of the BBQ contour is that it contains already a first rim of pixels with value less than the delineation level around the object. With an OBQ contour, this rim would be added to the object at a later stage, during the propagation of the fall-set. For this reason, in the actual program the BBQ contour construction procedure is adopted.

Both contour types may show penninsular protrusions that are connected to the object region (OBQ) or to the background region (BBQ) by an 'isthmus'

Table 1

Actions in MARKER for all combinations of incoming (row-number) and outward (column number) vector element direction codes (parentheses indicate impossible combinations)

OBQ-contour									BBQ-contour							
1	2	3	4	5	6	7	8		1	2	3	4	5	6	7	8
2	2	3	3	3	(0)	2	2	1	1	0	0	0	0	(1)	1	1
0	0	1	1	1	1	(0)	(0)	2	1	0	0	0	0	0	(1)	(1)
0	0	1	1	1	1	3	(0)	3	3	2	2	2	2	0	0	(3)
(0)	(0)	1	1	1	1	3	3	4	(3)	(2)	2	2	2	0	0	0
3	(0)	1	1	1	1	3	3	5	0	(2)	2	2	2	0	0	0
2	2	(0)	(0)	0	0	2	2	6	0	0	(2)	(2)	2	0	0	0
2	2	3	(0)	0	0	2	2	7	1	0	0	(3)	3	1	1	1
2	2	3	3	(0)	(0)	2	2	8	1	0	0	0	(3)	(1)	1	1

0, no operation; 1, add (mark) pixels below and including the contour pixel; 2, subtract (un-mark) pixels below and excluding the contour pixel; 3, add (mark) the contour pixel only. Below is in the direction with code 6.

of only one pixel wide. The protrusions contribute a lot to the perimeter and little to the area of the delineated object. Shape parameters computed from the contour may also be influenced seriously by the protrusions. Therefore, any protrusion involving an isthmic pixel is cut off during the contour construction. To this end the contour pixels are marked preliminarily through unary negation. If a pixel has been assigned before, the contour is traced back and undone until the same pixel is reached again. Now the search is resumed again in the direction that was used last in the examination plus 5. Before leaving the subroutine, all preliminary marking of the contour is undone.

In the next stage, performed by the subroutine **MARKER**, the perimeter and area of the delineated object are computed and the object pixels are marked. The computation of the area bears analogy with that of integrating around a closed curve: going around the contour in a clockwise direction, the number of pixels below and including the contour pixel is added to the area while going from left to right, and the number of pixels below and excluding the contour pixel is subtracted from the area while going from right to left. Because we are dealing with discrete pixels and vector elements (vexels), we have to analyse how to handle all 64 combinations of incoming and outward vexels. Fortunately, 12 combinations are impossible due to the definition of the contour, and we have to choose between only 4 different actions for each combination:

- 0: no action, i.e., neither add or subtract;
- 1: add the pixels below and including the contour pixel;
- 2: subtract the pixels below and excluding the contour pixel;
- 3: add the contour pixel only.

In table 1, the appropriate actions for each combination of incoming and outward vexels are presented, assuming direction code 6 as 'below'.

Due to the definition of the contour algorithm, the object area is systematically estimated too high within a BBQ contour. Therefore the computed area is then corrected for the contour width by subtracting the number of contour vexels from the number of object pixels. The resulting area is checked against a user-specified lower limit (actually 20).

During the process of area computation, the con-

tour is also examined on its direction of rotation: if the rotation is counterclockwise the contour is within an object, otherwise it is a contour of the desired type. The direction of rotation is computed by adding the differential direction codes, corresponding with the change in vixel direction at each contour pixel. To this end, the difference between the direction codes of the outward and incoming vexels at a contour pixel is brought into the range of possible values (from -3 to 4) by adding or subtracting 8, being one complete revolution. The resulting sum of the differential direction codes may either have the value 8 or -8. If it is 8, the contour was obviously within the object, and a new starting pixel has to be found on the 'outside' of the object. The search for that pixel should start from within the concavity enclosed by the contour, and in the same direction as in which the former starting pixel was found.

The contour length is also computed during the same process by assigning an average vixel length $\bar{l}$ to each of the direction codes. As an estimate of this average length, the following values have been adopted for the vexels with even (suffix e) and odd (suffix o) direction code:

$$\bar{l}_e = \frac{\frac{\pi}{4} - \frac{1}{2} \log 2}{\sqrt{2} - 1} (\approx 1.06) \quad \bar{l}_o = \frac{\log 2}{2 - \sqrt{2}} (\approx 1.18)$$

These values have been derived in a statistical analysis of the length of continuous contour segments cut off by the digitization grid, taking the direction into account [13]. In order to get a contour length that is most closely related to the computed area, a correction is necessary. This correction should account for the fact that the continuous contour does not run through the pixels but in between the OBQ and BBQ pixels. The correction consists of the average length of the two vexels, which should be added to (OBQ) or subtracted from (BBQ) the sum of the average vixel lengths.

The values adopted for the average vixel lengths sometimes may lead to a paradox: an object perimeter may result which is shorter than the circumference of a circle with the same area enclosed. This situation arises, e.g., with a 'regular' octagonal BBQ contour with side 2, enclosing an area of 21, and having a perimeter of 15.7 corresponding with a circular area of 19.6, resulting in a contour ratio of

0.93. The cause of the paradox is in the assumption of negligible curvature of contour segments, or sufficiently high sampling rate, used in the derivation of the average voxel lengths. If the direction of rotation and the object area each have met their specific requirements, the object pixels are marked. The marking process is analogous to that of computing the area: instead of adding or subtracting pixel values, they are negated in situ. Negation is feasible for marking because the input values are definitely positive. After the marking of the object, resulting negative pixel values are transformed by the function LOGTAB and added resulting in the total content of staining material, at this stage not yet corrected for the background value. At the same time, the corresponding pixels are displayed on a $x - y$ -addressable device, if desired.

The next step is to extend the object into the background region, i.e., to propagate the fall-set. This is accomplished in the subroutine EXTEND by examining every neighbour of a contour pixel that is not yet assigned to the object, to see if the value is less than that of the contour pixel. If this condition is met indeed, the neighbour pixel is assigned to the (extended) object and displayed if desired, in the same way as in the subroutine MARKER. Besides, the index is stored in a buffer, so that the process may be iterated after application to all contour pixels, by substituting the freshly assigned pixels for the contour pixels. The iteration is finished if no new pixel can be assigned to the extended object, or after a distinct number of iterations (actually 10). The freshly

assigned pixel indices are stored in the same buffer as the previously assigned ones. To this end, two buffer indices running in the same direction are used for reading and writing, the latter starting with an arrearage to the former indices. At the beginning of each new iteration the read/write direction is reversed. The only error condition that may arise in EXTEND is overflow of the buffer, like with the storage of direction codes in CONTUR.

In another method proposed by Rutovitz [11] the array is scanned in two opposite directions, meanwhile propagating decreasing pathways from the object in (4) forward directions. The method renders virtually the same results, although it is more elegant and less complicated than ours. But it seems less efficient on sequential computers, especially if the size of the extended object is small compared with the pixel array size, because it addresses the search for fall-set pixels sequentially, instead of randomly as we do. The method of Rutovitz seems more efficient however, if an array-processor is used, especially if the object is small initially.

In the subroutine BACKGR, the pixels that are representative of the background are selected. We expect to find these pixels at or near the fall-set contour, but still we have to choose between the contour pixels themselves (the inner contour), their direct neighbours just outside the fall-set (the outer contour), or a weighted mean of both. The propagation of the fall-set is halted at the inner contour by higher valued pixels. Therefore the inner contour pixels have a higher probability than their neighbours of being a

Fig. 4. Histogram (drawn schematically) of the fall-set pixel values. The broken lines indicate the distributions of the inner (I) and outer (O) contour pixel values.

local minimum, and consequently of not being representative of the local background as well (cf. fig. 4). These outer pixels are rejected if they differ from their inner contour neighbour more than the background spread (an output parameter of the subroutine HISPAN). Then they are assumed to belong to a neighbouring object.

The choice between the inner and the outer contour pixel values, or their weighted mean, was based upon an experiment involving cells and chromosomes, taken in absorbance as well as in fluorescence mode, scanned from negatives as well as from original slides, resulting in density as well as in intensity values. The only selection criterion for the test image data was that the total content of staining material could also be computed using the FASTSCAN method using two rectangular homogeneous background zones of at least three pixels wide. In this way the total content values resulting from DIODA could be compared with those from FASTSCAN. It should be borne in mind that FASTSCAN certainly does not provide an absolute standard, because the background region is defined completely ad-hoc as two strips beside the object, and because it is not always certain that the object is the only one present in the scanned array.

In the experiment the influence of noise was studied by adding gaussian distributed noise in exponentially increasing amounts to the original data. This led to the following conclusions, calling the total content based on the inner or outer contour background value T_i or T_o , respectively:

- the systematic error in T_i is always positive and nearly always greater than the absolute value of the same error in T_o ;
- the random variations in T_i are generally greater than in T_o ;
- the systematic errors are largely reduced by the application of median smoothing [14], to a level of $< \pm 1\%$ of T_o ;
- the random variations in T are slightly reduced by the application of median smoothing, to a level of $\sim 1.5\%$ of the total content;
- there is no fixed ratio between the two systematic errors in T_i and T_o precluding the use of a weighted mean of the two total contents.

Summarizing these results, it is clear that the outer contour provides the most reliable background value, with one exception however: if it consists of too few

pixels. A low number of pixels may be caused by propagation of the fall-set up to the pixel array boundary, or up to many pixels rejected because they are assumed to belong to a neighbouring object. So, if the number of outer contour pixels remains below a specified limit (actually 10), the background value is computed from the median smoothed inner contour pixels. The experiment has also been extended to the question whether to propagate the fall-set through pixels not greater than the previous one (strictly non-increasing), or through pixels less than the previous one (strictly decreasing fall-set). It should be noted that if the precision of the pixel values is sufficiently high, no difference between the alternatives will be found. If this is not the case however, the strictly non-increasing fall-set has the advantage that it has turned out to produce results that are much less dependent on microscope focusing [11]. On the other hand, it entails a high risk of the propagation being halted by the array boundary rather than by higher-valued pixels. In that case, the number of outer contour pixels remains low, resulting in an inaccurate background value. Weighing the alternatives against each other, and considering that our input data have a precision of ten bits, we adopted the strictly decreasing fall-set.

As already mentioned in section 3, DIODA is the enveloping subroutine for the other subroutines described so far. It starts with calling HISPAN for the computation of an initial value of the delineation level. This level is then optimized towards a minimal value of the contour ratio. The minimum is found by preliminarily executing the subroutines STAPIX, CONTUR and MARKER (the last one without actually marking the object pixels) in a user specified number of trials (actually 11). The lowest trial value of the delineation level is just above the background (at R in fig. 2), and with each next trial this value is incremented by half the background spread as computed in HISPAN. The trial level at which any unwanted event occurs is rejected.

Experiments show that the minimum of the contour ratio is usually found after 3–5 increments, and that the increments are small enough to detect the minimum. In our implementation the number of trials is rather high to ensure separation of touching objects (cf. fig. 5). On the other hand a high number of trials presents some risk of the object falling apart

by the segmentation. For banded chromosomes, e.g., this may happen after more than 5 trials, also depending on the specified lower limit for the object area.

With the optimum delineation level obtained in this way, the sequence of subroutines is executed once again, now including EXTEND and BACKGR. By way of experiment this has been done for several other values of the delineation level as well. This experiment shows that a change in the level as computed in HISPAR by an amount equal to the background spread induced a change in the area of the fall-set of <1.5%, and in the content of <0.3%, the changes in object area and perimeter being rather unsystematic. This outcome shows that the object content is virtually not influenced by changes in the delineation level within reasonable limits. If the number of trials is kept low by the user, there is a certain risk that all trials end in any of the unwanted events described so far, requiring an adequate reaction in DIODA. The events may be summarized as follows, stating their respective event parameter values and messages:

- 1 in STAPIX no startpixel .GE. level found;
- 2 in STAPIX an isolated startpixel .GE. level found;
- 3 in STAPIX no pixel pair .GE./LT. level found;
- 4 in CONTUR/EXTEND buffer size too small;
- 5 in CONTUR pixel array boundary touched;
- 6 in MARKER counterclockwise (=inner) contour detected;
- 7 in MARKER area below specified minimum;
- 8 number of background pixels below specified minimum.

The last event does not require any reaction, because it is just a signal that the background value was not computed from the outer fall-set contour, but from the inner one instead. An adequate reaction to the other conditions is a retry of DIODA with a different delineation level. This level should be decreased after events 1, 2, 6 and 7, and increased after events 3, 4 and 5. Before decreasing the level after event 6, a retry is performed starting at the neighbour of the current starting pixel in the current search direction. Note that decreasing the level in this case only makes sense if the previous level produced an inner contour within the object, and not one within the background region. In the latter case the level should be increased instead. If the delineation level needs to be changed, this is initially done by an

amount of half the background spread, like in the optimizing trials. If a change in level is necessary again, this amount is kept at the same value if the sign of the change remains unchanged. Otherwise the amount of change in level is multiplied by $-\frac{1}{2}$, in order to produce a convergent series of level values when subsequent events impose conflicting demands. After a specified number of iterations (actually 10) the data is assumed to be incorrect input for DIODA, because it does not contain an object meeting the following requirements:

- having an area greater than the specified minimum (event 2 or 7);
- not touching the array boundary (event 5 or 6);
- having a contour as well as a largest extension rim fitting into the available buffer (event 4).

Finally, there is the extreme situation where all pixels have the same value, resulting in events 1 and 3 alternatively.

As mentioned already in the introduction, the program DIODA is intended to make redundant a display of the scanned image, and interaction with the latter. If DIODA is not completed successfully however, or if the resulting total content or other parameters raise suspicion, visual examination of the scanned image becomes necessary. In these cases, a good rendition of the pixel values is very important. But the display and hard-copy facilities that provide grey-value rendition are quite expensive. So, like many others, e.g. [6], we resort to binary (bilevel) dot display devices, such as a storage-tube graphic terminal or a printer-plotter, that can produce grey-values by representing a pixel through a square submatrix of ($n \times n$) dots. The number of illuminated (or white) dots is made proportional to the measured light intensity at the pixel: all $n \times n$ dots illuminated correspond with maximum intensity.

There are many methods known to fill these submatrices [15,16]. Most of them require considerable computing capacity to avoid perceptibility of the submatrical edges or regularity in the pattern of illuminated dots. Pertaining to sharpness of the image, this is especially necessary with images consisting of relatively few pixels, like 0.5 μm scans of single biological objects. These images need a strong magnification (e.g., $n = 10$) in order to be viewed from a distance that makes the individual dots hardly perceptible.

One of the simplest methods for bilevel display

mentioned in literature is the sigma—delta method [17]. Essentially, this method conceives each image pixel as a matrix of $n \times n$ elements with each the value of the pixel. Scanning the image thus expanded from top left to bottom right, an element is displayed if the sum (sigma) of the elements past exceeds the maximal grey-value value. The rest of the sum beyond the maximum (delta) is the first contribution to the sum over the next elements, and serves as a randomizer for the displayed dot position. In this way the total illumination of the resulting picture differs only by the final value of delta — i.e., by less than the maximal grey-value — from the total intensity of the image. This procedure is adapted to two dimensions in order to avoid display of anisotropic dot patterns. But still the history of delta is similar over n subsequent lines, apart from the delta values at the end of the line. The total illumination is preserved also if the direction of the scanning with delta is reversed — from left to right to the opposite, or vice versa — at the beginning of each new scan line. Then the history of the randomizer is different, resulting in a still improved picture quality. Results of the three procedures are given in fig. 5.

The procedure (MASID) described requires a minimum of computer capacity and renders a picture quality which may only be improved at the cost of considerable computing capacity by the use of a larger environment than two elements for the determination of delta, or by the use of adaptive matched thresholds [15,16]. The latter methods do not introduce so much artificial texture as equally spaced dots, but they introduce an artificial dot pattern in a grid instead.

4. Specifications

The program DIODA is entirely written in FORTRAN IV. After compilation with the F4P compiler (Digital Equipment Co.) the total size of the described object modules is ~6 k (16 bits) words. The largest object modules are those of LOGTAB and MASID, which together account for nearly 50% of the total size, mainly because they each have an internal buffer of 1 k words. So the smallest feasible total size of the DIODA object is ~3.5 k words, if the buffer is removed from LOGTAB and MASID is discarded. To this size should still be added the size of the scan data array and the object module size of the main program that provides I/O facilities.

DIODA only requires an (alphanumeric) $x - y$ -addressable display device if the resulting fall-set pixels are to be displayed. If MASID is used, some bilevel display device is required, such as a storage-tube graphic terminal or a printer-plotter. The program has been tested on various members of the DEC PDP-11 family, resulting in differences in the speed of execution only.

The performance of DIODA has been measured on a PDP-11/60 by implementing it in a parameter extraction program (PEP), running under the RSX-11D operating system. The test input and results are summarized in table 2, which shows that the required computing time is related to the size of the processed data. Discarding the level optimization procedure turned out to shorten the required computing time by ~30%. Note that the indicated times do not include the execution of the main program (with disk I/O) and that no display facilities have been used.

Table 2
Average required computing time for DIODA per object without I/O for various test data (absorption).

	Data	Image type	Approx. size in core	Average elapsed time	Number of failures
430	cells (cervix)	pos.	0.1 k 1.5 k	1.7 s	1
377	cells (cervix)	neg.	2 k 8 k	4.2 s	—
246	cells (mamma)	pos.	2 k 8 k	4.7 s	1
290	cells (mamma)	neg.	2 k 8 k	4.0 s	—
43	chromosomes	neg.	3 k 4 k	4.5 s	3

Although sophisticated methods for evaluation of the results are available [18], their application is considered beyond the scope of this communication. The quality of the resulting segmentation was sufficient for practical purposes, except for the failures indicated in table 2. With the cell images both failures were due to event 8: the number of outer fall-set contour pixels was below 10. With the chromosome images, the failures did not result in any event code, but in an area that was far too small, since one band of the chromosome was mistaken for the object. This means that the lower limit set for the area of the cells (20 pixels) has been too small, preventing the signaling of this fragmentation through event code 7. After adjustment of the area lower limit to 500 (or of the number of trials to 5, which is preferred) the chromosome images were processed without any failure.

It should be noted that outright wrongly scanned images were excluded from the test input summarized in table 2. Besides, it should be borne in mind that the quality of the segmentation also depends on the type and quality of the staining and microscopy. In our situation the program has proven to produce reliable results.

5. Availability

Readers interested in obtaining program copies are invited to contact the first author directly.

References

- [1] W.A.L. Duijndam, A.W.M. Smeulders, P. van Duijn and A.C. Verweij, Optical Errors in Scanning Stage Absorbance Cytophotometry, I. Procedures for Correcting Apparent Integrated Absorbance Values for Distribution, Glare and Diffraction Errors, *J. Histochem. Cytochem.* (1979) submitted.
- [2] M. van der Ploeg, P. van Duijn and J.S. Ploem, High-Resolution Scanning Densitometry of Photographic Negatives of Human Metaphase Chromosomes, I. Instrumentation, *Histochemistry* 42 (1974) 9–29.
- [3] M.L. Mendelsohn, B.H. Mayall, J.M.S. Prewitt, R.C. Bostrom and W.G. Holcomb, Digital Transformation and Computer Analysis of Microscopic Images, *Adv. Opt. Electron Microsc.* 2 (1968) 77–150.
- [4] M. van der Ploeg, A.M. Vossepoel, F.Th. Bosman and P. van Duijn, High-Resolution Scanning Densitometry of Photographic Negatives of Human Metaphase Chromosomes, III. Determination of Fluorescence Emission Intensities, *Histochemistry* 51 (1977) 269–291.
- [5] M. van der Ploeg, K. van den Broek, A.W.M. Smeulders, A.M. Vossepoel and P. van Duijn, Hidacsys: Computer Programs for Interactive Scanning Cytophotometry, *Histochemistry* 54 (1977) 273–288.
- [6] E. Bengtsson, J. Holmquist, B. Olsen and B. Stenkvist, Scancans – An Interactive Scanning Cell Analysis System, *Comput. Prog. Biomed.* 6 (1976) 39–49.
- [7] L.S. Davis, A Survey of Edge Detection Techniques, *Comput. Graph. Image Proc.* 4 (1975) 248–270.
- [8] K.S. Fu and J.K. Mui, A Survey on Image Segmentation. in: *Proc. Int. Conf. Pattern Recognition of Cell Images*, Chicago, IL, May 21–23, 1979.
- [9] M.L. Mendelsohn, B.H. Mayall, Chromosome Identification by Image Analysis and Quantitative Cytochemistry, in: *Human Chromosome Methodology*, J.J. Yunised, 2nd edn, pp. 311–346 (Academic Press, New York, 1974).
- [10] F.C.A. Groen and M. van der Ploeg, DNA Cytophotometry of Human Chromosomes, *J. Histochem. Cytochem.* 27 (1979) 436–440.
- [11] D. Rutovitz, Expanding Picture Components to Natural Density Boundaries by Propagation Methods. The Notions of Fall-Set and Fall-Distance, in: *Proc. 4th Int. Joint Conf. on Pattern Recognition*, Kyoto, Nov. 7–10, 1978.
- [12] H. Freeman, A Review of Relevant Problems in the Processing of Linedrawing Data, in: *Automatic Interpretation and Classification of Images*, A. Grasselli ed., pp. 155–174 (Academic Press, New York, 1969).
- [13] F.C.A. Groen and P.W. Verbeek, Freeman-Code Probabilities of Object Boundary Quantized Contours, *Comput. Graph. Image Proc.* 7 (1978) 391–402.
- [14] L.W. Abele and F.M. Wahl, A Digital Procedure for Boundary Detection and Elimination of Background in Cytologic Images, in: *Medinfo 77*, D.B. Shires and H. Wolf, eds, pp. 1035–1038 (North-Holland, Amsterdam, 1977).
- [15] J.A.G. Hale, Dot Spacing Modulation for the Production of Pseudo Grey Pictures, *Proc. SID* 17 (1976) 63–74.
- [16] R.W. Floyd and L. Steinberg, An Adaptive Algorithm for Spatial Grey Scale, *Proc. SID* 17 (1976) 75–77.
- [17] H. Inose and Y. Yasuda, A Unity Bit Coding Method By Negative Feedback, *Proc. IEEE* 51 (1963) 1524–1535.
- [18] W.A. Yasnoff, J.K. Mui and J.W. Bacus, Error Measures for Scene Segmentation, *Pattern Recogn.* 9 (1977) 217–231.