ELSEVIER

Journal of Neuroscience Methods 93 (1999) 121–131

JOURNAL OF NEUROSCIENCE METHODS

www.elsevier.com/locate/jneumeth

Computer-vision-based extraction of neural dendrograms

Roberto Marcondes Cesar Jr. [a,b,*,1], Luciano da Fontoura Costa [a,2]

[a] *Cybernetic Vision Research Group, GII-IFSC-University of São Paulo, Caixa Postal 369, São Carlos, SP 13560-970, Brazil*
[b] *Department of Computer Science, IME-University of São Paulo, Rua do Matão, 1010, São Paulo, SP 05508-900, Brazil*

Received 31 December 1997; received in revised form 3 July 1999; accepted 18 August 1999

Abstract

This work introduces a new approach to the characterization of neural cells by means of semi-automated generation of dendrograms; data structures which describe the inherently hierarchical nature of neuronal arborizations. Dendrograms describe the branched structure of neurons in terms of the length, average thickness and bending energy of each of the dendritic segments and allow in a straightforward manner, the inclusion of additional measures. The bending energy quantifies the complexity of the shape and can be used to characterize the spatial coverage of the arborizations (the bending energy is an alternative for other complexity measures such as the fractal dimension). The new approach is based on the partitioning of the cell's outer contour as a function of the high curvature points followed by a syntactical analysis of the segmented contours. The semi-automated method is robust and is an improvement on the time consuming manual generation of the dendrograms. Several experimental results are included in this paper which illustrate and corroborate the effectiveness of the approach. The technique presented in this paper is limited to planar neurons but could be extended to a 3D approach.

Keywords: Neural shape; Computer vision; Dendrograms; Curvature; Formal grammars; Image processing

1. Introduction

Different problems in the analysis of neural cells have been addressed in the literature, ranging from computer-aided extraction of neural data from micrographs (also known as neuron reconstruction) (Wann et al., 1973; Capowski and Sedivec, 1981) to the measurement of quantitative features such as area, length, number of branch points, orientation histograms and convex hull (e.g. Ventimiglia et al., 1995). A particularly important related research field is the computer-aided analysis of neural morphology and automated neuromorphometry, where numerical measurements are extracted allowing quantitative analysis of neural shape (Murray, 1995; Panico and Sterling, 1995; Smith et al., 1996; Costa et al., 1998). Additional examples of works dealing with quantitative analysis of neuronal morphology include (Sholl, 1953; Ramon-Moliner, 1962; Smith et al., 1989; Masseroli et al., 1993; Matesz et al., 1995). It is important to note that neural morphology is one of the most relevant parameters in the analysis of neurons, since this kind of information can be used for simulation or studies dealing on interplay between form (morphology) and function (Saito, 1983; Fukuda et al., 1984; Poznanski, 1992; Mainen and Sejnowski, 1996; Costa et al., 1998). This work introduces a new approach to neural shape representation and analysis through semi-automated dendrogram generation.

The newly proposed framework is based on the following steps: (a) an effective algorithm for determination of branch points and terminations has been developed with basis on image analysis techniques; (b) the obtained branch points and terminations are graphically presented to the user, who can check the results and edit the representation if needed; and (c) the dendrogram is derived by using computational grammars. The remainder of the present paper elaborates on these three principal processes and the respectively obtained results. The computational background for the imple-

* Corresponding author. Tel.: +55-16-273-9858; fax: +55-16-271-3616.

E-mail addresses: cesar@ime.usp.br (R.M. Cesar, Jr.), luciano@ifqsc.sc.usp.br (L. da Fontoura Costa)

[1] http://www.ime.usp.br/~cesar

[2] http://www.ifqsc.sc.usp.br/visao

0165-0270/99/$ - see front matter
PII: S0165-0270(99)00120-X

mentation of the techniques discussed in this paper may be found in Castleman (1996) and Cesar and Costa (1997a) for the material about digital contours and curvature; in Brigham (1974) for the material about Fourier transforms and algorithms; and in Fu (1982) for the material about syntactical pattern recognition.

This work starts by establishing a conceptual overview of the proposed method, as well as the basic terminology that is adopted throughout the paper (Section 2.1). The method for neural contour partitioning is introduced in Section 2.2.1. Finally, the parsing analysis and the subsequent dendrogram generation are discussed in Sections 2.2.2 and 2.2.3, respectively. Section 2.2.4 presents how, once the dendrogram has been extracted, it is possible to obtain the *skeleton* of the neural cell in a simple and straightforward manner. The skeleton is a structure composed by a set of line segments (which may be one pixel wide) that represents the neural cell, including many different useful features for shape analysis (Ballard and Brown, 1982). Experimental results using real neurons are presented, thus corroborating the effectiveness of this approach. The article concludes with some comments on the ongoing and future research, presented in Section 4. It is worth emphasizing that the method proposed in this paper may be understood conceptually from Sections 1 and 2.1. Nevertheless, since this does not supply enough information allowing the method to be effectively implemented, the necessary details are given in Section 2.2.

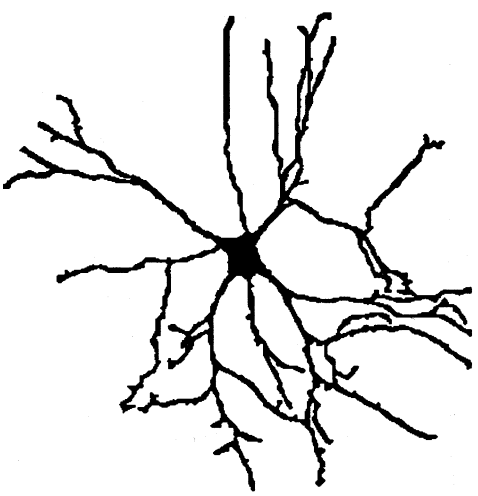

Fig. 1. Cell used in the experiments of this paper, which has been adapted from Smith et al. (1989), where further details may be found (reprinted from Smith Jr TG, Marks WB, Lange GD, Sheriff WH, Neale EA. A fractal analysis of cell images, J. Neurosci. Methods, 1989;27:173–80, with permission from Elsevier Science).

2. Material and methods

2.1. Preliminary considerations

2.1.1. Conceptual overview

Among the different approaches to the characterization of neural cells, the so called *dendrogram* (Sholl, 1953; Poznanski, 1992; de Schutter and Bower, 1994; Turner et al., 1995; Velte and Miller, 1995; Cesar and Costa, 1997b) deserves special attention because of the high degree of comprehensiveness and meaningfulness made explicit by this data structure (tree). When extended to include not only the underlying branching structure of biological patterns, but also the length and other physical properties, such as diameter, thickness and curvature, along each branch, the binary tree created to describe such branched structures receives the name of *dendrogram*. Fig. 3 presents a neural dendrite (respective to the neural cell presented in Fig. 1, which has been reproduced from Smith et al. (1989)), whose respective dendrogram is shown in Fig. 7. In this example, the length of each dendritic segment (i.e. arc length between each pair of bifurcations and terminations) is proportional to the length of the horizontal stripes represented in the dendrogram.

In spite of the importance of such data structures as subsidy for the characterization, classification, modeling and simulation of natural structures, the possibility of computer-based extraction of dendrograms from pictorial representations has received scant attention in the literature. Though developed in a 2D space, the technique described in the current article can be used to process some 3D cells, especially those presenting relatively few processes and/or presenting a predominantly planar structure, in terms of their 3D projections. Typical examples of predominantly planar neurons are provided by the dendritic arborizations of some neurons in the retina of vertebrates. Dendrograms of such neural cells are important not only as invaluable resources for simulations of the respective electrochemical neural activity (transmission cable modeling) (e.g. Poznanski, 1992), but also as a means for the effective characterization and classification of neural cells and structures. Considering that neural dendrograms have been traditionally traced by human operators, involving a long, tedious and often subjective task, the automation of such a process constitutes an important resource for neuroscientific research.

The method introduced in this paper is based on contour analysis of neural cells. In order to gain insight into the different steps involved in the method, specific upper-left dendrite in Fig. 1, which is shown in detail in Fig. 2(a) will be used as a reference example throughout the paper. This dendrite is shown in more detail in Fig. 3. The proposed approach can be easily understood from the schematic sequence shown in Fig. 2. The basic

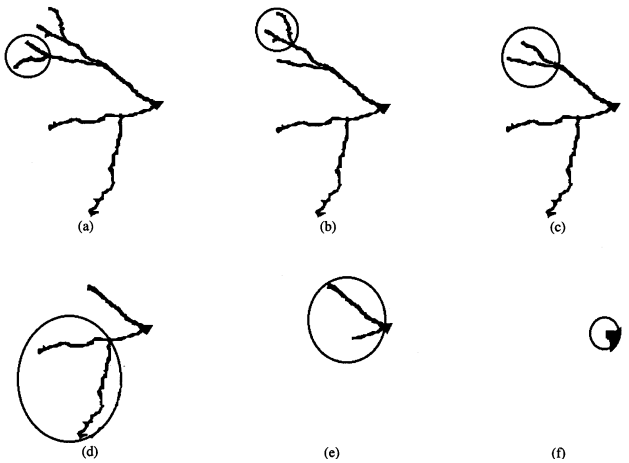

Fig. 2. Basic underlying idea of the method: the algorithm analyses the outer and smaller most dendritic branches first, as if a process of cutting off these branches were being carried out.

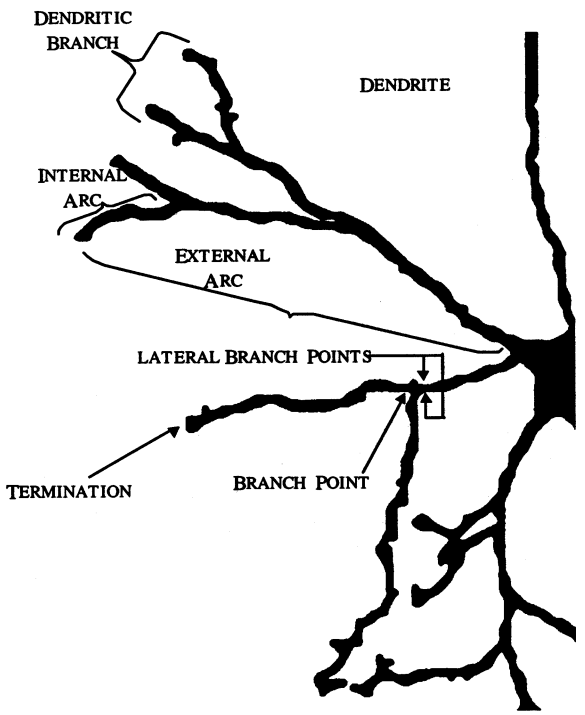

Fig. 3. Terminology adopted throughout the paper.

idea underlying the method is that the algorithm tries to identify the outermost (and smaller) dendritic branches (see the adopted terminology in Fig. 3). When such a dendritic branch is identified, the respective measures are taken (e.g. segments length and average width) and stored in the dendrogram data-structure. Once this operation has been carried out, the analyzed dendritic branch can be skipped, as if it had been 'cut off' (see Fig. 2a and b). The algorithm then proceeds in a similar way in order to find the next outermost (and smaller) dendritic branch (see Fig. 1b–f). The relation between the idea of pruning distal branches with the parametric contour representation of the dendrite lays in the parsing procedure, as it is explained below, which identifies the portions of the contour that are eliminated as the syntactic productions are applied.

The present approach has considered the analysis of the dendritic branches directly in terms of the outer boundaries of the cell, as illustrated in Fig. 4. Two special kind of critical or *dominant points* along such contours are considered in the present article: the terminations, identified by the prefix *e*, and the branch points, identified by *b*, which are characterized as belonging to convex and concave portions of the contour (the concavity can be inferred from the sign of the curvature at the contour points). Assuming that such points have been correctly identified, it is possible to approach the problem of dendrogram extraction in terms of computer grammars, more specifically through the parsing of a basic sequence defining the underlying structural atoms in dendrograms, namely the sequence termination/branch-point/termination. However, this parsing can not be performed directly, since there are ambiguities in such grammatical representations of dendritic trees. The solution to such problems is described in the following sections.

As has been commented, the method attempts to identify the outer and smaller most dendritic branches first (see Fig. 2), proceeding recursively as these structures are identified. The detection of these structures is based on the identification of terminations and branch points along the contour. The latter step is carried out from the analysis of the curvature along the dendritic contour, while the former is done through a syntactical analysis of the string obtained from the sequence of terminations and branch points. The output of the syntactical procedure is a data structure, known as a *parsing tree* (Fu, 1982), which describes the dendritic branching structure, and from which the dendrogram can straightforwardly be obtained. These three basic steps of the method, namely contour segmentation in terms of terminations and branch points, syntactical analysis and dendrogram generation from the parsing tree, constitutes the kernel of the proposed approach, being explained in detail in Sections 2.2.1, 2.2.2 and 2.2.3, respectively.

2.1.2. Terminology and conventions

A complete set of relatively formal specific terms related to parts of neural cells is introduced here in order to facilitate the description of the algorithms in the following sections. Fig. 3 presents an image of a dendrite illustrating the basic terminology adopted throughout the paper. Fig. 3 shows the aforementioned neuron emphasizing one of its *dendrites*, which corresponds to the complete dendritic structure originating at the soma. The dendrite is composed of a set of dendritic segments delimited by a *branch point* on one

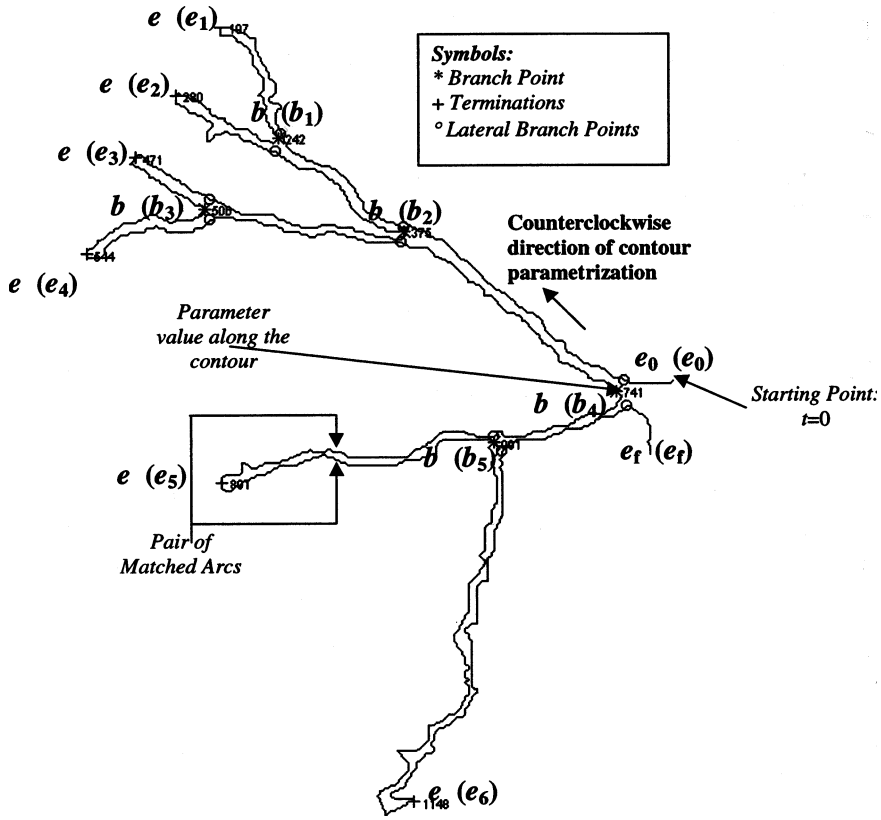

Fig. 4. A dendrite with its segmentation points (branch points and terminations) indicated. Contour segments of a dendrite and its segmentation points are automatically detected by the segmentation method.

side and a *termination* or another branch point on the other side.

The branching structures that emanate from other branches are called *dendritic branches*, as depicted in Fig. 3. The analysis of each dendrite is carried out from its outer contour, which begins and ends at the soma. The success of the method does not depend on the direction which the contour is tracked (i.e. clockwise or counterclockwise). It is important to emphasize that this direction determines the sign of the curvature of the terminations and branch points.

A bifurcation of the dendrite is defined by a branch point and two associated *lateral branch points*, as indicated in Fig. 3. The central branch point is connected to a termination by the *internal arc*. In a similar manner, the termination is connected to the next branch point by the *external arc*. It is important to emphasize that the definition of internal and external arcs is done *with respect to each branch point*. This means that, the internal arc with respect to a given branch point may be the external arc with respect to a previous or a consecutive branch point. For instance, in Fig. 3, the indicated internal arc is defined in this manner with respect to the branch point where it originates. The algorithm keeps track of such situation by analyzing the dendrite with respect to the current branch point. In this work, any contour portion will be henceforth referred as an *arc*.

2.2. The dendrogram generation method

2.2.1. Contour segmentation

The methods presented in this section are based on the multiscale curvature approach (derived from the *curvegram*) introduced in Cesar and Costa (1997a). The framework assumes the neural cell is represented in terms of its outline contour, which is defined by a parametric curve $C(n) = (x(n), y(n))$, where the parameter $n = 0,\ldots, N-1$ and N is the number of points along the contour. The number of points along the contour depends whether the contour has been extracted directly from the digital image or interpolated so that n can be taken as the arc length parametrization of C (in this case, the distances between consecutive points are equal). A detailed discussion about the importance of the contour tracking algorithm with respect to the estimation of curvature may be found in Bennett and MacDonald (1975). A complex representation of the contour follow from the definition of the complex signal $u(n)$, i.e. $u(n) = x(n) + iy(n)$ and $i = \sqrt{-1}$. The Fourier transform pair of $u(n)$ is given in Eqs. (1) and (2) (Brigham, 1974):

$$U(s) = F\{u(n)\} = \sum_{n=0}^{N-1} u(n)\, e^{-i2\pi(sn/N)}, \quad s = 0,\ldots, N-1, \tag{1}$$

$$u(n) = F^{-1}\{U(s)\} = \frac{1}{N}\sum_{s=0}^{N-1} U(s)\, e^{i2\pi(sn/N)}. \tag{2}$$

In order to estimate the discrete derivatives of a digital signal $u(n)$, the auxiliary function $\eta(s)$ should be defined as expressed by Eq. (3):

$$\eta(s) = \begin{cases} s, & \text{if } s = 0, 1,..., N - \text{floor}(N/2) - 1 \\ N - s, & \text{if } s = N - \text{floor}(N/2), \\ & N - \text{floor}(N/2) + 1, ..., N-1, \end{cases} \tag{3}$$

where floor($N/2$) is the traditional truncation function that returns the largest integer that is smaller than $N/2$. Function $\eta(s)$ is a useful tool for implementing the alignment with the standard indexing representation normally produced by the FFT (because FFT algorithms usually lead to a frequency representation formed by the second part of a period followed by the first part of the next period). Therefore, the first and the second derivatives of $u(n)$ can be defined in terms of $U(s)$ as expressed by Eqs. (4) and (5):

$$\dot{u}(n) = F^{-1}\{\dot{U}(s)\} = F^{-1}\{i2\pi\eta(s)U(s)\}, \tag{4}$$

$$\ddot{u}(n) = F^{-1}\{\ddot{U}(s)\} = F^{-1}\{-(2\pi\eta(s))^2 U(s)\}, \tag{5}$$

where $n = 0, 1, \ldots, N-1$ and $s = 0, 1,\ldots, N-1$. The differentiation process defined by Eq. (4) and Eq. (5) corresponds to high-pass filters, which can increase the high-frequency noise commonly found in digital contours. In order to cope with such a problem, the differentiated spectra should be smoothed by using a low-pass filter to control the noise. By taking the Gaussian $G_a(s)$ as low-pass filter, where $G_a(s) = \exp(-(a\eta(s))^2)$, the smoothed version $u(n, a)$ of $u(n)$ can be estimated as:

$$u(n, a) = F^{-1}\{U(s)G_a(s)\}. \tag{6}$$

The above equation is equivalent to a convolution operation due to the convolution theorem (Brigham, 1974). Gaussian smoothing implies an undesired effect known as contour shrinking (Cesar and Costa, 1997a), which can be avoided through perimeter preservation. The perimeter of the original neural contour can be estimated as:

$$L = \sum_{n=0}^{N-1} |u(n) - u(n-1)|,$$

where $|u(n)|$ denotes the complex modulus (absolute value of the complex number) and the convention that $u(-1) = u(N-1)$ is adopted (this is needed because the signal u represents a closed periodic contour stored in a vector). The perimeter of the filtered contour (at scale a) can be estimated analogously, and we denote it as $L(a)$. The amount of shrinkage can be calculated as:

$$P(a) = \frac{L}{L(a)}.$$

The differentiated signals $\dot{u}(n, a)$ and $\ddot{u}(n, a)$, filtered at scale a and corrected in order to prevent the undesirable shrinking effect, can be given by Eqs. (7) and (8):

$$\dot{u}(n, a) = P(a)F^{-1}\{\dot{U}(s)G_a(s)\}, \tag{7}$$

$$\ddot{u}(n, a) = P(a)F^{-1}\{\ddot{U}(s)G_a(s)\}. \tag{8}$$

The multiscale curvature description of $C(n)$, known as *curvegram* (Cesar and Costa, 1997a), is given by Eq. (9):

$$k(n, a) = \frac{-\text{Im}\{\dot{u}(n, a)\ddot{u}^*(n, a)\}}{|\dot{u}(n, a)|^3}, \tag{9}$$

where z^* denotes the complex conjugate. The curvegram $k(n, a)$ describes the curvature of $C(n)$ (recall that $C(n)$ is the original contour that has been represented by the complex-valued signal $u(n)$) analyzed at scale a (one can also think that k describes the curvature of a family of smoothed curves indexed by the scale parameter a). Fig. 5(a) presents the signals $x(n)$ and $y(n)$ obtained from the neural cell seen in Fig. 2. Fig. 5(b) and (c) show the corresponding curvatures for a small and for a large scale, respectively. As can be seen, fine details are filtered out from the curvature as the analyzing scale increases.

An important shape feature that can be measured from the curvature is the so called *bending energy* (Cesar and Costa, 1997a), which express the amount of energy that would be spent in order to bend a given shape into its lowest energy state, namely a circle (Young et al., 1974; Cesar and Costa, 1997a). Let $k(n, a_0)$ be the estimated curvature at scale a_0 (obtained from the curvegram). Then the mean bending energy is defined as:

$$B(a_0) = \frac{1}{N}\sum_{n=0}^{N-1} k(n, a)^2. \tag{10}$$

The bending energy has been extensively discussed in the context of neural shape analysis elsewhere (Cesar and Costa, 1997a). It is important to observe that the bending energy can be also calculated with respect to a portion of the contour, which is actually done in this work, where the bending energy of the separated dendritic segments is calculated and represented in the respective dendrogram.

The identification of the neural contour terminations and branch points is carried out by choosing an appropriate scale a_0, followed by two threshold operations, one for the detection of the branch points of the neurons (defined by negative minima of curvature, which characterize concavities, assuming that the contour is traversed counterclockwise), and other for the detection of the terminations (defined by positive maxima of curvature characterizing convexities, assuming

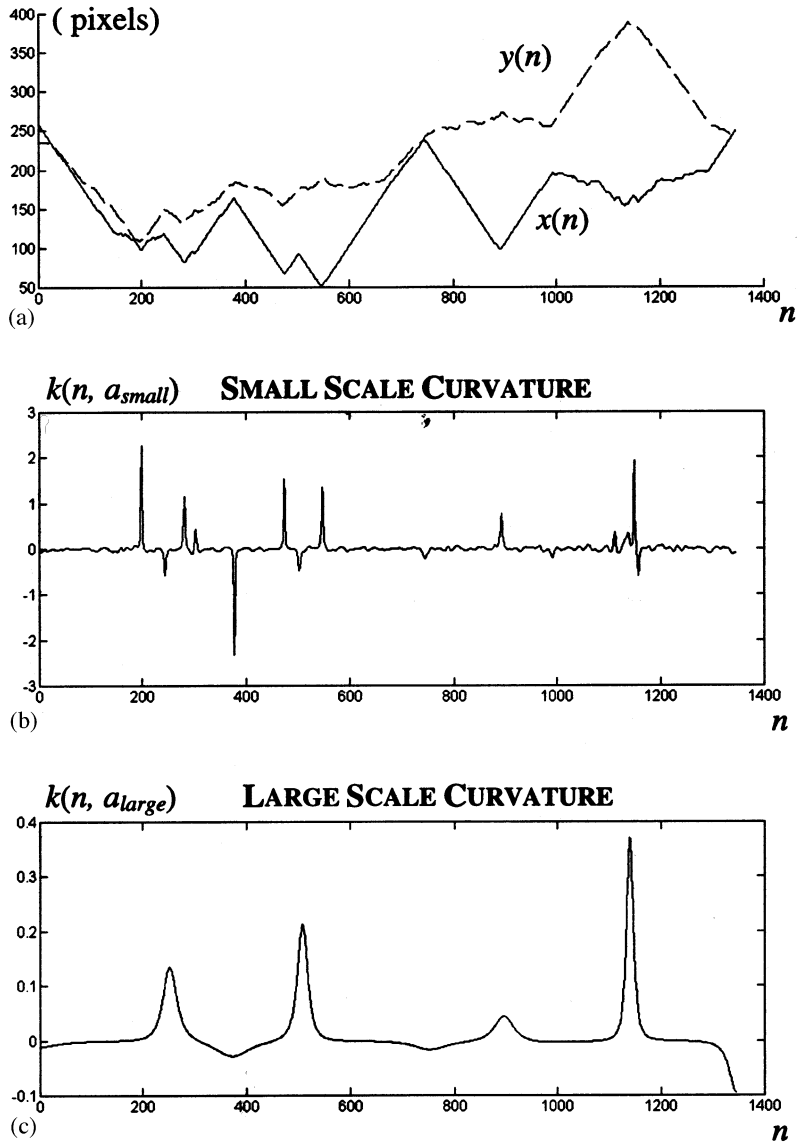

Fig. 5. (a) Signals $x(n)$ (–) and $y(n)$ (– –) of the contour of the neuron shown in Fig. 2; (b) curvature of the contour for a small scale, where the fine details can be detected; (c) curvature of the contour for a large scale, where only the overall structure of the original contour remains. It is assumed that $a_{\text{large}} > a_{\text{small}}$.

that the contour is traversed counterclockwise). Furthermore, it is assumed that there is a minimum neighborhood around each dominant point (be it a termination or a branch point) inside which there must be only one dominant point. The algorithm for dominant point detection can be summarized as follows:

1. Algorithm 1: dominant point detection

2. Calculate $k(n, a_0)$, as defined by Eq. (9), where a_0 is a constant;

3. Find all branch points defined as $B = \{u(n)$ such that $k(n, a_0) < T_B$, *and* $k(n, a_0)$ *is a local negative minimum*$\}$;

4. Find all terminations defined as $E = \{u(n)$ such that $k(n, a_0) > T_E$, *and* $k(n, a_0)$ *is a local positive maximum*$\}$;

5. Filter E and B, so that the minimum neighborhood around each termination (or branch point) contain only one termination (or branch point);

6. End.

In our implementation, $k(n, a_0)$ is a local positive maximum if $k(n, a_0) > k(n+1, a_0)$ and $k(n, a_0) > k(n-1, a_0)$. The local negative minimum may be defined analogously. **Algorithm 1** requires the specification of four parameters, the analyzing scale a_0, the thresholds T_E and T_B and the size of the minimum neighborhood for dominant points. These parameters are set by three semi-automated procedures that may require operator interventions. A fourth semi-automated procedure is also carried out, which asks the operator for final adjustments in the contour partitioning, both for the inclusion of missing and exclusion of

false dominant points detected by **Algorithm 1**. Details about the implementation of these procedures using graphical user interfaces may be obtained by request from the authors. In all experiments, the minimum neighborhood for the digitized version of the contour has been set as three points on each side of the dominant point. Once each dendrite is properly segmented, the program execution continues by analyzing each dendrite separately. In order to accomplish this task, each dendrite is reparameterized, with the parameter along the dendrite starting from the first branch point on the soma, tracking the dendrite boundary and terminating on the second branch point on the soma. Hence, in the example of Fig. 4, the new parameterization starts for $t = 0$ in the e_0 on the soma, continues reaching the first termination located at $t = 197$, and so on.

2.2.2. Parsing

The concepts developed in this section, which are central to the dendrogram generation, are based on the structural or syntactic approach for pattern recognition (Fu, 1982). The objective of this step is to derive a parsing tree reflecting the structure of the neural dendrite. The parsing step starts by labeling the dominant points obtained by the contour partitioning algorithm described in last sections, so that the initial and final points of the dendritic branch are labeled with the symbols 'e_0' and 'e_f' (see Fig. 4), respectively. Each termination point is labeled by the symbol 'e' and each branch point by the symbol 'b' (refer to Fig. 4). Therefore, we define a grammar $G = (V_N, V_T, P, S)$, where $V_N = \{E\}$, $V_T = \{e_0, e_f, e, b\}$, $S = \{E0\}$, and P corresponds to the following production rules:

1. $E0 \rightarrow e_0 E e_f$
2. $E \rightarrow EbE$
3. $E \rightarrow e$

As an illustration, the corresponding symbols have been indicated in the segmented dendrite in Fig. 4. As can be seen, the string representing that dendrite is: $e_0 ebebebebebee_f$. The bottom-up parsing procedure is (seemingly) very simple, due to the simplicity of the grammar. The implementation of the parsing procedure makes use of a data structure called a *ternary tree*, which may have up to three derivations per node. This tree is constructed incrementally during the parsing analysis.

It is important to note that, during the execution of the bottom-up syntactical analysis, the algorithm should keep a control list that indicates the upper nodes of the unfinished parsing tree, i.e. those nodes that have been already created but not yet assigned to the hierarchy. This list, referred to henceforth as the *control string*, plays a central role in the parsing analysis, as well as in the determination of the lateral branch points, as it will be explained. A preliminary version of the algorithm is:

1. Algorithm 2: Parsing (Version 1)

2. Replace all the occurrences of 'e' by 'E', through the application of rule 3;
3. Replace the occurrences of 'EbE' by 'E', through the application of rule 2;
4. Replace the occurrence of '$e_0 E e_f$' by '$E0$', through the application of rule 1;
5. End.

It is useful to compare this algorithm with the procedure explained conceptually in Section 2.1.1 and summarized in Fig. 2. The applications of step 3 of the above algorithm corresponds to the 'cutting' of an outer and smaller most dendritic branch.

Nevertheless, there is an important problem associated with step 3 of **algorithm 2** since different derivations may lead to the same string, which means that the order of application of rule 2 defines different parsing trees. As a means of circumventing this ambiguity, a heuristic is applied in step 2 of **algorithm 2**. In order to understand how this works, imagine that each application of rule 2 of the grammar during the parsing procedure is equivalent to cut out a branch corresponding to the respective 'EbE' that is being substituted by 'E'. Then, the key idea underlying the heuristics is to construct the parsing tree by cutting the smaller, which are also the last, outermost branches first. Therefore, a simple analysis of the dendritic tree of Fig. 4 indicates that the correct order of application of rule 2 is that presented in Fig. 6. The parsing algorithm must decide whether or not to make the substitution $EbE \Rightarrow E$ once an EbE sub-string is found during the parsing.

This decision, which is based on a comparison between the arc length of segments of the current branch and the arc length of segments of its neighbor branches, is taken as follows. In the example of the dendrite of Fig. 4, the application of the step two of **algorithm 2** leads to the following substitutions in the control string:

$$e_0 ebebebebebee_f \rightarrow e_0 EbEbEbEbEbEe_f \tag{11}$$

Having the symbols in Fig. 4 been properly enumerated, the following notation can be used:

$$e_0 e_1 b_1 e_2 b_2 e_3 b_3 e_4 b_4 e_5 b_5 e_6 e_f \rightarrow e_0 E_1 b_1 E_2 b_2 E_3 b_3 E_4 b_4 E_5 b_5 E_6 e_f \tag{12}$$

Refer to Fig. 6. It is worth emphasizing that the enumerated symbols are used in the paper only for explanation purposes and that the algorithm actually manipulates only the symbols defined in the formal grammar introduced in the beginning of this section. First, once an EbE sub-string is found, which will be called *the current EbE*, the algorithm tests whether there is another EbE sub-string to the right of the current EbE, where the second 'E' of the first sub-string is the first 'E'of the second. Following the notation of the above example indicated by Eq. (12) the algorithm

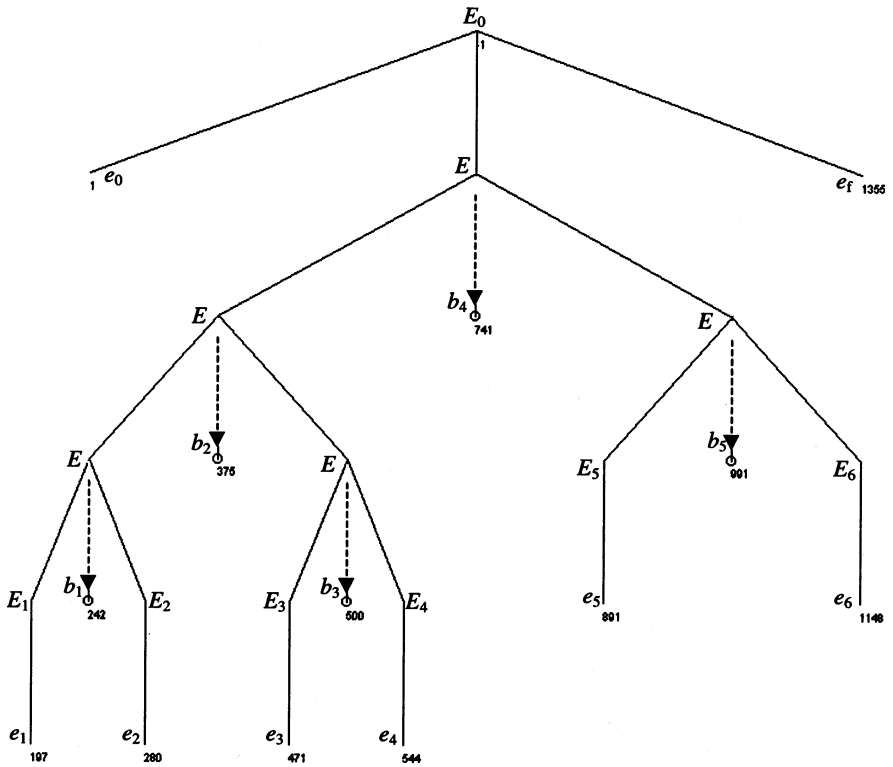

Fig. 6. Parsing tree obtained by the parsing algorithm for the dendrite shown in Fig. 4.

first finds the sub-sequence $E_1b_1E_2$ and verifies whether the union of this sub-string with the next two symbols actually forms the sub-string $E_1b_1E_2b_2E_3$. In this case, the arc length between the first 'b'[3] and the middle 'E' (i.e. the *internal arc*, see Fig. 3) is compared to the arc length between this same middle 'E' and the second 'b' (i.e. the *external arc*, see Fig. 3). In the case of the current example, the length of the arc between b_1 and E_2 (the current right internal arc) is compared to the length of the arc E_2 and b_2 (the current right external arc; refer to Fig. 6).

In a similar manner, the algorithm checks whether there is another *EbE* sub-string to the left of the current *EbE*, where the union of these two sub-strings forms the sub-string '*EbEbE*'. In this case the length of the left internal arc is compared to the length of the left external arc:

if (length(left internal arc) ≤ length(left external arc))
and (length(right internal arc) ≤ length(right external arc))
then
the sub-string of the current *EbE* is replaced by *E*.
else
the algorithm skips the first sub-string and tries to substitute the second *EbE* sub-string, repeating the same test.

These two comparisons are made assuming the existence of a sub-string '*Eb*' to the left and a sub-string '*bE*' to the right of the current '*EbE*'. In the case when one (or both) of them are not present, the respective(s) test is just ignored, which occurs when the branch does not have neighboring branches to the right or to the left (or both). For instance, in the example of Fig. 4 the sequence '$E_1b_1E_2$' (corresponding to a '*EbE*' sub-string) has the sub-string 'b_2E_3' (corresponding to a '*bE*' sub-string to the right), allowing the test to be carried out with respect to the right side; on the other hand, this same sequence '$E_1b_1E_2$' does not present a previous branch (a branch to the left; hence, there is no sub-string '*Eb*' to the left) and the test with respect to the left side is simply ignored.

This heuristic induces the algorithm to attempt making substitutions corresponding to the smaller branches first, as desired. Therefore, **algorithm 2** should be updated to version 2, which will be called **algorithm 3**:

1. Algorithm 3: Parsing (Version 2)
2. Change all the occurrences of '*e*' by '*E*', through the application of rule 3;
3. Find the first occurrence of '*EbE*' of the control string and call it 'current "*EbE*"'
4. While there are remaining '*EbE*' sub-strings do:

[3] The contour branch point corresponding to the '*b*', or any other symbol in a string, will be henceforth referred to as 'first "*b*" point' and so on, for simplicity's sake.

5. If a '*bE*' sub-string to the right of the current '*EbE*' is found then
   6. If (length(right internal arc) ≤ length(right external arc)) then
      7. Substitution_flag = true;
   8. Else
      9. Substitution_flag = false;
   10. End_if /*of step 6*/
11. Else
   12. Substitution_flag = true;
13. End_if /*of step 5*/
14. If a '*Eb*' sub-string to the left of the current '*EbE*' is found then
   15. If (length(left internal arc) ≤ length(left external arc)) then
      16. Substitution_flag = true;
   17. Else
      18. Substitution_flag = false;
   19. End_if /*of step 15*/
20. Else 21. Substitution_flag = true;
22. End_if /*of step 14*/
23. If (Substitution_flag = true) then
   24. Change the current '*EbE*' by '*E*' of the control string, following rule 2
25. End_if /*of step 23*/
   26. Find the next occurrence of '*EbE*' of the control string and call it 'current "*EbE*"'
27. End_while /*of step 4*/
28. Change the occurrence of 'e_0Ee_f' by '*E*0', through the application of rule 1;
29. End.

2.2.2.1. Detection of lateral branch points. An additional important feature of the parsing procedure is that it allows a region-based segmentation of the neural cell in terms of its dendritic segments. Fig. 4 presents a schematic illustration of the concepts involved in this segmentation process. First, note that each dendritic segment is defined in terms of its respective *branch point*, which is located by the (contour) segmentation algorithm introduced previously, and the point located on the opposite side of the dendritic segment, called *lateral branch point* (see Fig. 4). The lateral branch points of the experiments of this work are denoted by '°', as illustrated in Fig. 4. Each branch point is associated to two lateral branch points (the left branch point and the right branch point), since two dendritic segments appear from a branch point, and there is always a termination or a dendritic branch between the branch point and its respective lateral branch point. The control string provides a straightforward heuristics that can be applied to define and locate the lateral branch point. Recall that the dominant points found by the segmentation algorithm define a contour partitioning in terms of segments between each pair '*eb*' or '*be*', and that the initial control string of the dendritic tree of Fig. 4 is $e_0ebebebebebee_f$. Therefore, the left lateral branch point of the first branch point of this dendritic tree is defined as the nearest point belonging to the segment comprised between the next two symbols to the left of this branch point, i.e. between e_0 and e_1 in our example. Both branch points and associated lateral branch points are indicated in Fig. 4. All other lateral branch points can be located during the parsing procedure in a similar manner, by searching for the nearest point (to the current branch point) belonging to the segment defined between the next two symbols to the left (or to the right) of the control string.

Finally, it is important to observe that each dendritic segment is defined by a *pair of matched arcs*, each of them defined in terms of a branch point and a lateral branch point, which establish the beginning of the dendritic segment, and a termination or a dendritic sub-tree, which identifies its respective ending. An example of such a pair of matched arcs is indicated in Fig. 4.

2.2.3. The dendrogram

The dendrogram is obtained from the parsing tree in a straightforward manner. Recall that the dendrogram describes the dendritic segments between the successive branch points and terminations. It is also important that the parsing procedure explained previously generates a parsing tree that respects the smaller-to-larger branches in a dendritic structure (see Fig. 6). Furthermore, the branch points and the terminations can be retrieved in an ordered manner from the parsing tree, once each node of a symbol '*E*' is the father of either a node '*b*' (i.e. a branch point) or of a node '*e*'. Therefore, the dendrogram can be obtained by traversing the parsing tree, starting from the initial parent node, and measuring the desired feature (e.g. the arc length distance) between the current node and its derivations (i.e. the length of each dendritic segment). It is equally important to note that each branch is composed of a pair of matched arcs (see Fig. 4). Hence, the length of each branch is defined as the mean between the arc lengths of one pair of matched segments. In other words, let $u_l(t)$ and $u_r(t)$ be the pair of matched segments, properly reparameterized, where u_l and u_r denote the left and the right segment, respectively. Then, the length of the horizontal line segment of the dendrogram corresponding to that dendritic segment is proportional to the mean of the arc lengths of $u_l(t)$ and $u_r(t)$.

In order to understand this process, refer to Figs. 6 and 7, which present, respectively, the parsing tree automatically generated by the parsing algorithm and the corresponding dendrogram of the dendrite in Fig. 4. The first dendritic segment corresponds to that beginning in the first and the last points of the dendrite (points e_0 and e_f of Fig. 4) and finishing at the lateral

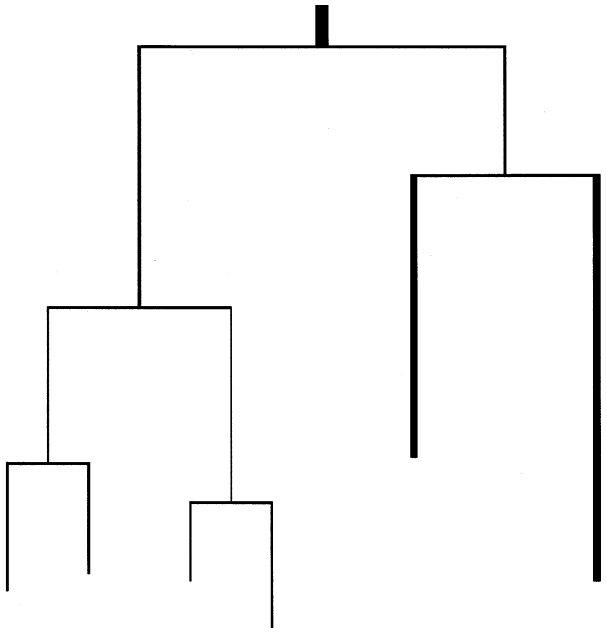

Fig. 7. Dendrogram of the dendrite indicated in Fig. 3. In this dendrogram, the length of each dendritic segment is coded by the length of the respective horizontal line of the dendrogram; the average thickness by the thickness of the dendrogram horizontal lines; and the bending energy (a measure of segment complexity) by the gray-level, higher values of energy being coded by darker gray-levels.

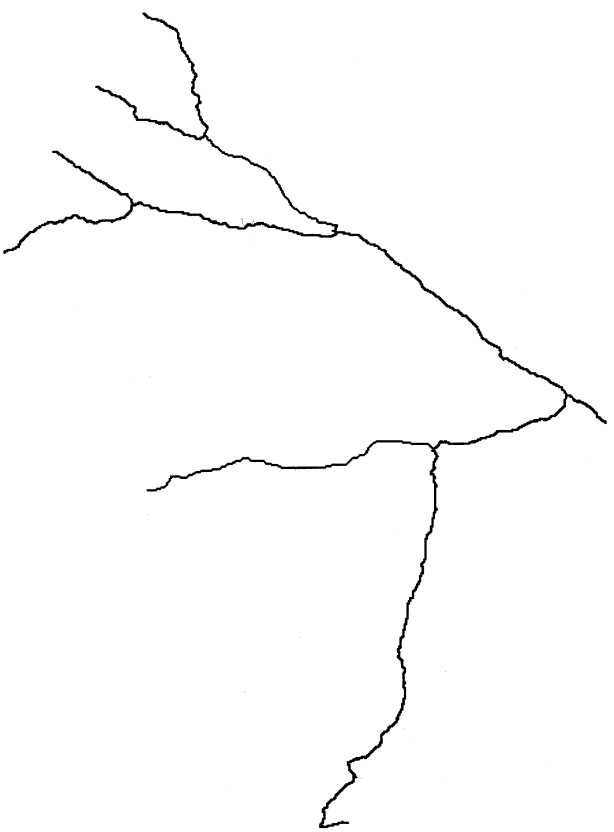

Fig. 8. Skeleton of the dendrite of Fig. 4 generated automatically from its respective parsing tree.

branch points associated to the branch point b_4 (see Fig. 4). Next, a horizontal straight line is drawn by the algorithm with length proportional to the length of this first segment, as shown in Fig. 7. Two new branches appear at this first branch point, and the dendrogram is ramified. The branch to the right of the parsing tree finishes at the branch point b_2, defining a new pair of matched segments, and the algorithm draws a new horizontal straight line with length proportional to the length of this segment. The algorithm continues recursively until all branches have been analyzed. The average thickness may be easily calculated as a mean of the distances between each pair of matched points, i.e. as the average of the distances between $u_l(t)$ and $u_r(t)$ for each point along the pair of matched contour segments.

2.2.4. Obtaining the skeleton

As has been explained, each dendritic segment is composed of a pair of matched contour segments. Therefore, the skeleton of each dendritic segment may be defined by the curve whose points are equidistant to a pair of corresponding points on the pair of matched contour segments. In other words, let $u_l(t)$ and $u_r(t)$ be a pair of matched contour segments, properly re-parameterized, of a given dendritic segment (see Section 2.2.3). Therefore the skeleton $\xi(t)$ of that dendritic segment is defined as:

$$\xi(t) = \frac{u_l(t) + u_r(t)}{2}.$$

The skeleton of the dendrite indicated in Fig. 4 can be seen in Fig. 8.

3. Experimental results

As has been commented, the dendrogram of the dendrite of Fig. 3 is presented in Fig. 7. This dendrogram actually represents three geometrical features for each dendritic segment: the dendritic length, represented in the dendrogram by the length of each horizontal line segment; the average thickness of the dendritic segment, represented by the thickness of each horizontal line segment of the dendrogram; and the bending energy of each dendritic segment, represented by the gray-level of each horizontal line segment of the dendrogram, darker gray-levels representing higher values of the bending energy. Furthermore, in order to make the bending energy to represent the overall shape complexity of each dendritic segment, the curvature around each segmentation point (terminations and branch points) was not considered in the bending energy summation (Eq. (10)), since those points present very high curvature values, which could mask the overall shape complexity. A neighborhood of three points around each segmentation point has been used in this process.

4. Discussion and conclusions

This work has presented a new approach for shape analysis of neural cells based on the semi-automated generation of dendrograms. The proposed framework, which relies on the multiscale curvature-based segmentation of the neural contour, followed by syntactic analysis of the partitioned shape, has shown to be robust and to greatly improve the dendrogram generation process. The proposed method allows the measuring of different geometrical features from the dendritic segments that compose dendritic arborization. The geometrical features that can be measured include length, average thickness, bending energy and average curvature, to name but a few. As an additional virtue of the proposed framework, the skeleton of the dendrite can be obtained in a straightforward manner as a by-product of the parsing procedure employed in the shape analysis.

Further refinements on the method will include the introduction of more intelligent mechanisms to help the operator in procedure 4, as well as automatically checking for valid strings generated by the segmentation algorithm. The extension to 3D structures and applications to other biological (2D) branching-like shapes are also being considered.

Acknowledgements

Luciano da F. Costa is grateful to FAPESP (94/3536-6, 94/4691-5 and 97/14678-4) and CNPq (301422/92-3) for financial help. Roberto M. Cesar Jr. is grateful to FAPESP (97/04186-7 and 98/07722-0) and CNPq (300722/98-2) for the financial support.

References

Ballard DH, Brown CM. Computer Vision. Englewood Cliffs, NJ: Prentice-Hall, 1982.

Bennett JR, MacDonald JS. On the measurement of curvature in a quantized environment. IEEE Trans Comput 1975;C-24(8):803–20.

Brigham EO. The Fast Fourier Transform. Englewood-Cliffs, NJ: Prentice-Hall, 1974.

Capowski JJ, Sedivec MJ. Accurate computer reconstruction and graphics display of complex neurons utilizing state-of-the-art interactive techniques. Comput Biomed Res 1981;14:518–32.

Castleman KR. Digital Image Processing. Englewood Cliffs, NJ: Prentice-Hall, 1996.

Cesar RM Jr, Costa L da F. Application and assessment of multiscale bending energy for morphometric characterization of neural cells. Rev Sci Instr 1997a;68(5):2177–86.

Cesar Jr RM, Costa L da F. Semi-automated dendrogram generation for neural shape analysis. In: Figueirado LH, Neto ML, editors. Proc. Brazilian Symposium of Computer Graphics and Image Processing (SIBGRAPI-97, Campos do Jordão, SP, Out 1997; http://www.visgraf.impa.br/sibgrapi97/anais/ART10/), IEEE Computer Society Press, Los Alamitos, California, 1997, pp. 147–54.

Costa L da F, Cesar Jr RM, Coelho RC, Tanaka JS. Perspective on the analysis and synthesis of morphologically realistic neural networks. In: Poznanski R, editor. Modeling in the Neurosciences: From Ionic Channels to Neural Networks. Amsterdam: Gordon and Breach Science Publishers, 1998 (in press).

Fu KS. Syntactic Pattern Recognition and Applications. Englewood Cliffs, NJ: Prentice-Hall, 1982.

Fukuda Y, Watanabe C-F, Ito H. Physiologically identified Y-, X-, and W-cells in cat retina. J Neurophysiol 1984;52(6):999–1013.

Mainen ZF, Sejnowski TJ. Influence of dendritic structure on firing pattern in model neocortical neurons. Nature 1996;382:363–6.

Masseroli M, Bollea A, Forloni G. Quantitative morphology and shape classification of neurons by computerized image analysis. Comput Methods Prog Biomed 1993;41:89–99.

Matesz C, Birinyi A, Kothalawala DS, Székely G. Investigation of the dendritic geometry of brain stem motoneurons with different functions using multivariant statistical techniques in frog. Neuroscience 1995;65(4):1129–44.

Murray JD. Use and abuse of fractal theory in neuroscience. J Comp Neurol 1995;361:369–71.

Panico J, Sterling P. Retinal neurons and vessels are not fractal but space-filling. J Comp Neurol 1995;361:479–90.

Poznanski RR. Modeling the electronic structure of starburst amacrine cells in the rabbit retina: functional interpretation of dendritic morphology. Bull Math Biol 1992;54:905–28.

Ramon-Moliner E. An attempt at classifying nerve cells on the basis of their dendritic patterns. J Comp Neurol 1962;119:211–67.

Saito H-A. Morphology of physiologically identified X-, Y-, and W-type retinal ganglion cells of the cat. J Comp Neurol 1983;221:279–88.

de Schutter E, Bower JM. An active membrane model of the cerebellar purkinje cell. I. Simulation of current clamps in slice. J Neurophysiol 1994;71:375–400.

Sholl DA. Dendritic organization in the neurons of the visual and motor cortices of the cat. J Anat 1953;87:387–407.

Smith TG Jr., Marks WB, Lange GD, Sheriff WH, Neale EA. A fractal analysis of cell images. J Neurosci Methods 1989;27:173–80.

Smith TG Jr., Lange GD, Marks WB. Fractal methods and results in cellular morphology-dimensions, lacunarity and multifractals. J Neurosci Methods 1996;69:123–36.

Turner DA, Li X-G, Pyapali GK, Ylinen A, Buzsaki G. Morphometric and electrical properties of reconstructed hippocampal ca3 neurons recorded in vivo. J Comp Neurol 1995;356:556–80.

Velte TJ, Miller RF. Dendritic integration in ganglion cells of the mudpuppy retina. Visual Neurosci 1995;12:165–75.

Ventimiglia R, Jones BE, Møller A. A quantitative method for morphometric analysis in neuronal cell culture: unbiased estimation of neuron area and number of branch points. J Neurosci Methods 1995;57:63–6.

Wann DF, Woolsey TA, Dierker ML, Cowan WM. An on-line digital-computer system for the semiautomatic analysis of Golgi-impregnated neurons. IEEE Trans Biomed Eng 1973;BME-20(4):233–47.

Young IT, Walker JE, Bowie JE. An analysis technique for biological shape. I. Inform Control 1974;25:357–70.