Chapter 7

Fractal Dimension Estimation Methods for Biomedical Images

Antonio Napolitano, Sara Ungania and Vittorio Cannata

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/48760

1. Introduction

The use of medical images has its main aim in the detection of potential abnormalities. This goal is accurately achieved with the synergy between the ability in recognizing unique image patterns and finding the relationship between them and possible diagnoses. One of the methods used to aid this process is the extrapolation of important features from the images called texture; texture is an important source of visual information and is a key component in image analysis.

The current evolution of both texture analysis algorithms and computer technology made boosted development of new algorithms to quantify the textural properties of an image and for medical imaging in recent years. Promising results have shown the ability of texture analysis methods to extract diagnostically meaningful information from medical images that were obtained with various imaging modalities such as positron emission tomography (PET) and magnetic resonance imaging (MRI). Among the texture analysis techniques, fractal geometry has become a tool in medical image analysis. In fact, the concept of fractal dimension can be used in a large number of applications, such as shape analysis[1] and image segmentation[2]. Interestingly, even though the fact that self-similarity can hardly be verified in biological objects imaged with a finite resolution, certain similarities at different spatial scales are quite evident. Precisely, the fractal dimension offers the ability to describe and to characterize the complexity of the images or more precisely of their texture composition.

2. Fractals

2.1. Fractal geometry

A fractal is a geometrical object characterized by two fundamental properties: *Self-similarity* and *Hausdorff Besicovich dimension*. A self-similar object is exactly or approximately similar to a part of itself and that can be continuously subdivided in parts each of which is (at least approximately) a reduced-scale copy of the whole. Furthermore, a fractal generally

shows irregular shapes that cannot be simply described by Euclidian dimension, but, fractal dimension (fd) has to be introduced to extend the concept of dimension to these objects. However, unlike topological dimensions the fd can take non-integer values, meaning that the way a fractal set fills its space is qualitatively and quantitatively different from how an ordinary geometrical set does.

Nature presents a large variety of fractal forms, including trees, rocks, mountains, clouds, biological structures, water courses, coast lines, galaxies[3]. Moreover, it is possible to construct mathematical objects which satisfy the condition of self-similarity and that present fd (Figure 1).

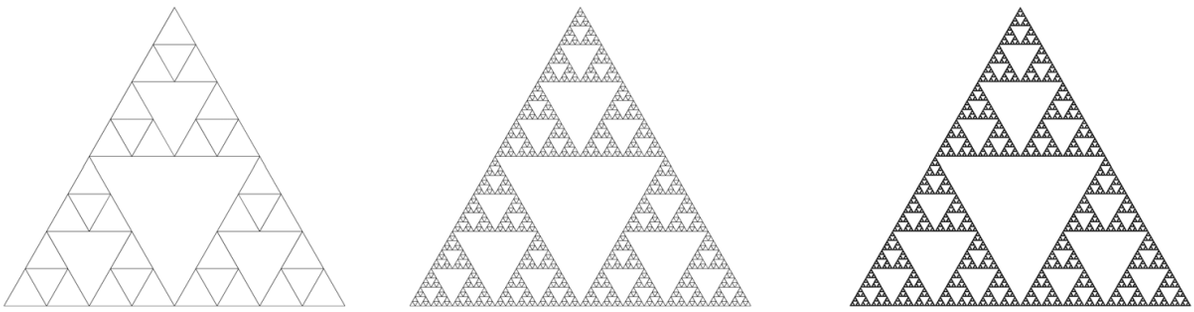


Figure 1. Sierpinski triangle: starting with a simple initial configuration of units or with a geometrical object then the simple seed configuration is repeatedly added to itself in such way that the seed configuration is regarded as a unit and in the new structure these units are arranged with respect to each other according to the same symmetry as the original units in the seed configuration. And so on.

The objects in Figure 1 are self-similar since a part of the object is similar to the whole and the fractal dimension can be calculated by the equation:

$$D = \frac{\log N}{\log S} \tag{1}$$

where N is the number of the auto-similar parts in which an object can be subdivided and S is the scaling, that is, the factor needed to observe N auto-similar parts. According to the Eq.1, the following values are obtained for the Koch fractal and the Sierpinski triangle:

$$D_{Koch} = \frac{\log 4}{\log 3} \approx 1.26 \qquad D_{Sierpinski} = \frac{\log 3}{\log 3} \approx 1.58 \tag{2}$$

In mathematics, no universal definition of fd exists and the several definitions of fd may lead to different results for the same object. Among the wide variety of fd definitions that have been introduced, the Hausdorff dimension D_H is surely the most important and the most widely used[4]. Such definition can be theoretically applied to every fractal set but has the disadvantage it cannot always be easily determined by computational methods.

2.2. Hausdorff dimension D_H

Hausdorff dimension D_H was introduced in 1918 by mathematical Felix Hausdorff [3]. Since many of the technical developments used to compute the Hausdorff dimension for highly irregular sets were obtained by Abram Samoilovitch Besicovitch, D_H is sometimes called Hausdorff-Besicovitch dimension.

Hausdorff formulation[3] is based on the construction of a particular measure, H_δ^D, representing the uniform density of the fractal object.

Intuitively we can sum up the construction as follows: let be A a fractal and $C(r, A) = \{B_1....B_k\}$ a complete coverage of A consisting of spheres of diameter smaller than a given r that approximate A, so $\delta_i = \delta_i(B_i) < r$.

We define the Hausdorff measure as the function H_δ^D that identifies the smallest of all the covering spheres for A with $\delta < r$:

$$H_\delta^D(A) = \omega_D \lim_{r \to 0} \{inf \sum_i \delta_i^D\} \tag{3}$$

with ω_D volume of the unit sphere in R^D for integer D.

We obtain an approximate measurement of A, the so-called *course-grained volume*[4].

In the one-dimensional case ($D = 1$), H_δ^D supplies the length of set A measured with a ruler of length r. The shorter the ruler, the longer the length measured, a paradox known as the *coastline paradox*[3].

Hence, when $r \to 0$ the effective length of A is well approximated. Limit for small r calculated for other values of D, however, lead to a degenerate H_δ^D:

$$\mathcal{H}_\delta^D \to 0 \quad \text{and} \quad \mathcal{H}_\delta^D \to \infty \tag{4}$$

Therefore, D_H can be defined as the transition point for the function H_δ^D monotonically decreasing with D:

$$D_H(A) = inf_{D>0} \{H_\delta^D(A) = 0\} \tag{5}$$

with H_δ^D the D-dimensional Hausdorff measure given by Eq. 3.

The *course-grained volume* defined by Eq. 3 normally presents a scaling like:

$$H_\delta^D \sim r^{(D-D_H)} \tag{6}$$

that provides a method to estimate the dimension D_H.

In the uni-dimensional case $D = 1$ we can easily obtain:

$$L_\delta^D \sim r^{(1-D_H)} \qquad \text{with } L_\delta^D = \textit{measured length} \tag{7}$$

from which we derive D_H.

3. Methods

Although the definition of Hausdorff dimension is particularly useful to operatively define the fd, that presents difficulties when implementing it. In fact, determining the lower bound value of all coverings, as defined in Eq. 5, can be quite complex. For example, let's consider the uni-dimensional case, in which we want to compute the fd of a coastline (Koch Curve). According to Eq. 3 in the case of $D = 1$ the coastline length is measured by a ruler of length r. Accuracy of the measure increase with decreasing r. For $r \to 0$ the coastline will have infinite length. Similar arguments can be applied to $D = 2$; for $r \to 0$ the measure of $H_\delta^D \to 0$.

This discussion implies that our coastline (ex. Koch Curve) will have a fd value more than one-dimensional and less than two- dimensional. For this reason, the fd is considered as the transition point (the lower bound value in Eq. 5) between $H_\delta^D \to 0$ and $H_\delta^D \to \infty$.

Several computational approaches have been developed to avoid the need of defining the lower bound at issue. Therefore many strategies accomplished the fd computation by retrieving it from the scaling of the object's bulk with its size. In fact, object's bulk and its size have a linear relationship in a logarithmic scale so that the slope of the best fitting line may provide an accurate estimation of this relationship. By using this log-log graph, called *Richardson's plot*, the requirement of knowing the infimum over all coverings is relaxed.

Several approaches have been developed to estimate fractal dimension of images. In particular, this section will introduce two fractal analysis strategies: the *Box Counting Method* and the *Hand and Dividers Method*.

These methods overcome the problem by choosing as covering a simple rectangle fixed grid in order to obtain an upper bound on D_H.

Five algorithms for a practical fd calculation based on these methods will also be presented.

3.1. Box counting method

The most popular method using the best fitting procedure is the so-called *Box Counting Method*[5][6]. Given a fractal structure A embedded in a d-dimensional volume the box-counting method basically consists of partitioning the structure space with a d-dimensional fixed-grid of square boxes of equal size r.

The number $N(r)$ of nonempty boxes of size r needed to cover the fractal structure depends on r:

$$N(r) \sim r^{-D} \tag{8}$$

The box counting algorithm hence counts the number $N(r)$ for different values of r and plot the log of the number $N(r)$ versus the log of the actual box size r. The value of the box-counting dimension D is estimated from the Richardson's plot best fitting curve slope.

$$-D = \lim_{r \to 0} \frac{\log N(r)}{\log r} \tag{9}$$

Figure 2 shows the Box counting method for the Koch Curve.

Several algorithms[7][8][9] based on box counting method have been developed and widely used for fd estimation, as it can be applied to sets with or without self-similarity. However, in computing fd with this method, one either counts or does not count a box according to whether there are no points or some points in the box. No provision is made for weighting the box according to the number of points belonging to the fractal and inside the current box.

3.2. Hand and dividers method

Useful features and information can be deducted from the contours of structures belonging to an image and there is a number of techniques that can be used when estimating the boundary fractal dimension.

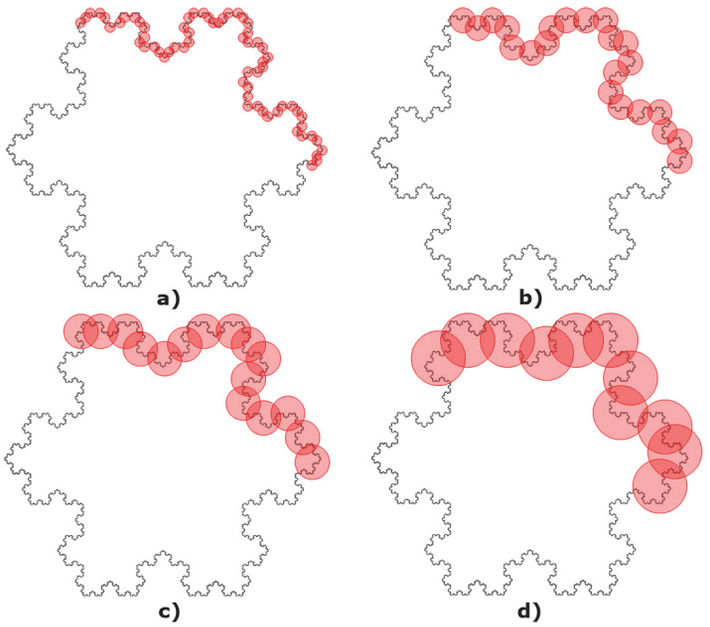


Figure 2. The Box-counting method applied to the Koch Curve with box size r = 0.4 (a); r = 1 (b); r = 1.4 (c); r= 2 (d)

The most popular methods are all based on the *Hand and Divers Method* which was originally introduced by Richardson[10] and successively developed by Mandelbrot[11].

The Richardson method employs the so-called *walking technique* consisting of "walking" around the boundary of the structure with a given step length.

The actual structure boundary is so approximated by a polygon whose length is equal to:

$$l(\epsilon) = \epsilon\, n(\epsilon) \tag{10}$$

In a nutshell, it corresponds to the length of the single step multiplied by the number of steps needed to complete the walk.

The process is then reiterated for different step lengths:

$$P_i = l(\epsilon_i) = \epsilon_i\, n(\epsilon_i) \tag{11}$$

With P_i the perimeter calculated with steps of length ϵ_i.

The object's boundary fd D is finally estimate from:

$$D = 1 - m \tag{12}$$

where m is the slope of the Richardson's plot.

The perimeter length of the boundary depends on the step length used so that a large step provides a rough estimation of the perimeter whereas a smaller step can take into account finer details of the contour.

Consequently, if the step length ϵ decreases the perimeter P increases.

In practice, the perimeter length is obtained by constructing a generally irregular polygon which approximate the border. Let δB be the set of coordinates of object boundary and let be

ϵ a fixed step length. Given a starting point, an arbitrary contour point (x_s, y_s), the next point on the boundary (x_{s_2}, y_{s_2}) in a fixed direction (e.g. clockwise) is the point that has a distance

$$d_i = \sqrt{(x_s - x_{s_2})^2 + (y_s - y_{s_2})^2} \tag{13}$$

as close as possible to ϵ.

The reached point then becomes the new starting point and is used to locate the next point on the boundary that satisfies the previous condition. This process is repeated until the initial starting point is reached.

The sum of all distances dj corresponds to the irregular polygon perimeter (Figure3).

A number of different perimeters for each polygon at each fixed step length are used to build the Richardson's plot and the slope of its best linear fit is exploited to estimate the fd.

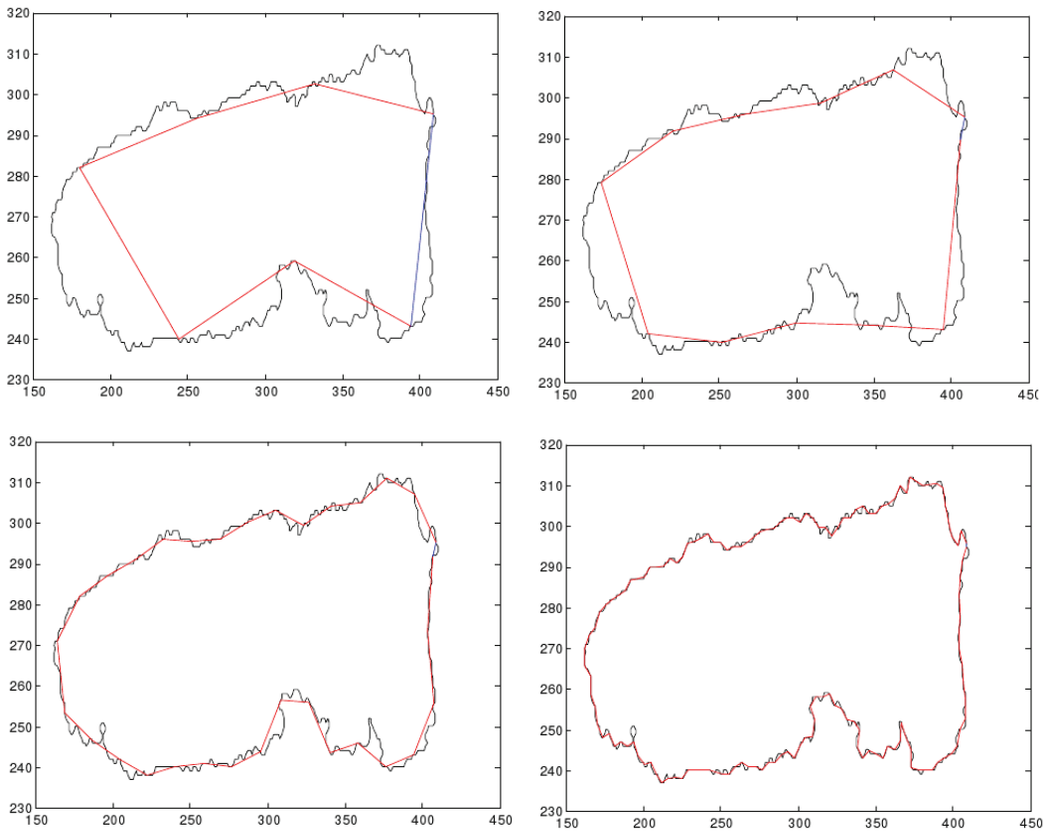


Figure 3. Walking technique applied to a coastline with different step lengths.

4. Algorithms

All Hand and dividers techniques rely on the same identical principle that attempt to approximate the border perimeter with a different polygons. However, since the point coordinates belonging to border set are discrete, all the implemented methods differ in the choice of which point in the set has a distance that better approximate the step length.

The following two methods are the implementations of two different choices about how to overcome this particular issue.

4.1. HYBRID algorithm

The HYBRID algorithm is a computer implementation of Hand and Dividers method developed by Clark[12]. Let δB be the boundary of the object whose fd we wish to compute. The main part of the method focuses on the perimeter estimation and the corresponding Richardson's plot is then attained by reiterating this hard core part at different step size. Figure 4 shows the flow chart of the method.

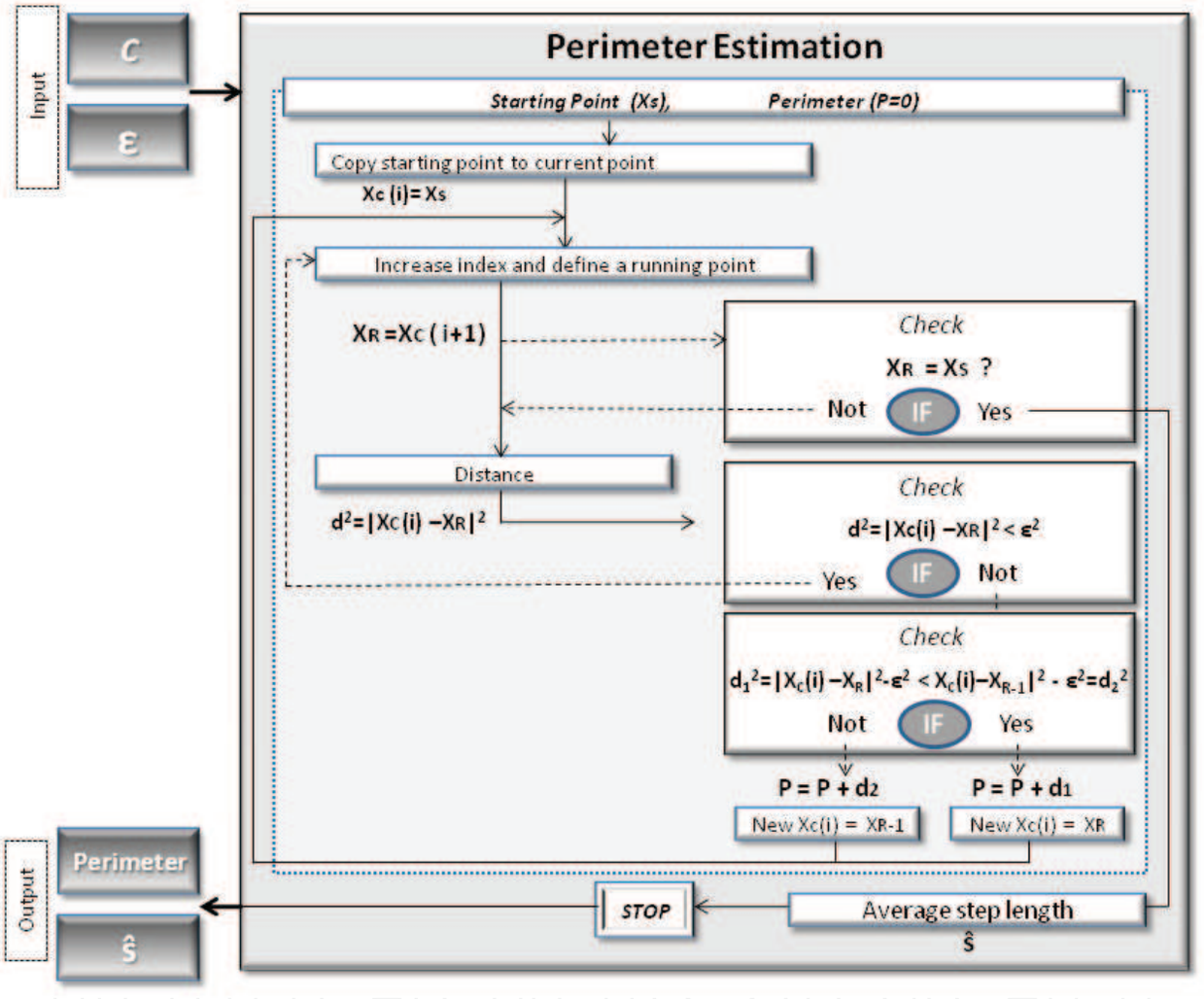


Figure 4. Perimeter estimation by HYBRID method flowchart: an arbitrary *starting point* S (x_S, y_S) on the boundary line is chosen and copied in a new variable, which is called *current point* C $(x_C(i), y_C(i))$. The index i runs through the total number of coordinate points and is iteratively increased defining the *running point* R with coordinates (x_R, y_R). The distance d between S and R is calculated and a check on d when smaller than a fixed step length ϵ is done. The process is repeated until a boundary point whose distance from (x_S, y_S) is larger than the step ϵ is reached. The next pivot point on the boundary line is determined by choosing between the two points the closest to the step length. The distance is then stored and this point becomes the new starting point in order to calculate the next pivot point and so on, until the initial starting point S is reached.

Given an arbitrary *starting point* S and its coordinates (x_S, y_S) on the boundary, the algorithm searches for the next pivot point. In particular the starting point is copied into a *current point*,

$C\ (x_C, y_C)$, which identifies all points having a mutual distance of about ϵ. The actual point running through the entire border is indicated as *running point* $R\ (x_R, y_R)$.

Therefore the program searches for a specific running point having a distance from C as near as possible to the step ϵ. In particular, in the HYBRID method the real step may be chosen to be longer or shorter than the fixed step depending on the minimum deviation from it. Similarly once the running point hits a contour point having a distance from the actual current one bigger than the size step, the choice is made between that point and the preceding one.

Afterward, the computed distance between these two points R and C is stored and the running point becomes the new current point.

The procedure continues until the initial starting point is reached. Obviously it is likely that after a complete walking the starting point S may be reached before having hit the following current point C. In other words, there may not be a multiple of step size ϵ so that the final incomplete step length r is added to the others stored distances, whose sum represent the boundary's perimeter. Since the fixed step length is adapted every time during the perimeter computation, its averaged value is then computed and used in the Richardson's plot.

4.2. EXACT algorithm

The EXACT algorithm was proposed for the first time by Clark in 1986[12]. As it will be shown, this method requires a longer computational time by providing a simpler solution to the choice of the best current points.

Similarly to the previous method the entire perimeter estimation is displayed in the flow chart of Figure 5.

The procedure is very similar to the one used for the previous method. As before (see Figure 5), the end of the step may not coincide with the digitized coordinates of the boundary.

The way the EXACT method attempts to overcome this problem relies on the assumption of piecewise linearity, meaning that all the points on the contour can be joined by a series of straight line[13, 14] (see Figure 6 (a)).

The location of the next current point C on the boundary from the one previously determined is schematically illustrated in Figure 6 (b).

The procedure starts from an arbitrary starting point (x_S, y_S) and the algorithm searches for the next pivot point. In particular the starting point is copied into a *current point*, $C\ (x_C, y_C)$, which identifies all points having a mutual distance of about ϵ. The actual point running through the entire border is indicated as *running point* $R\ (x_R, y_R)$.

The distance from the current point to each point on the contour line is then calculated until the step length ϵ falls between two consecutive boundary points, (x_R, y_R) and (x_{R+1}, y_{R+1}) for which:

$$\sqrt{(x_R - x_C)^2 + (y_R - y_C)^2} < \epsilon < \sqrt{(x_{R+1} - x_C)^2 + (y_{R+1} - y_C)^2} \tag{14}$$

The exact position of the point N with coordinates (x, y) is deduced by a process of geometric interpolation between the two consecutive running points (x_R, y_R) and (x_{R+1}, y_{R+1}). This

point then becomes the new current point and is used to calculate the next boundary point and so on.

The process is stopped when we come back to the initial starting point (x_S, y_S) in order to obtain a polygon as is shown in Figure 8.

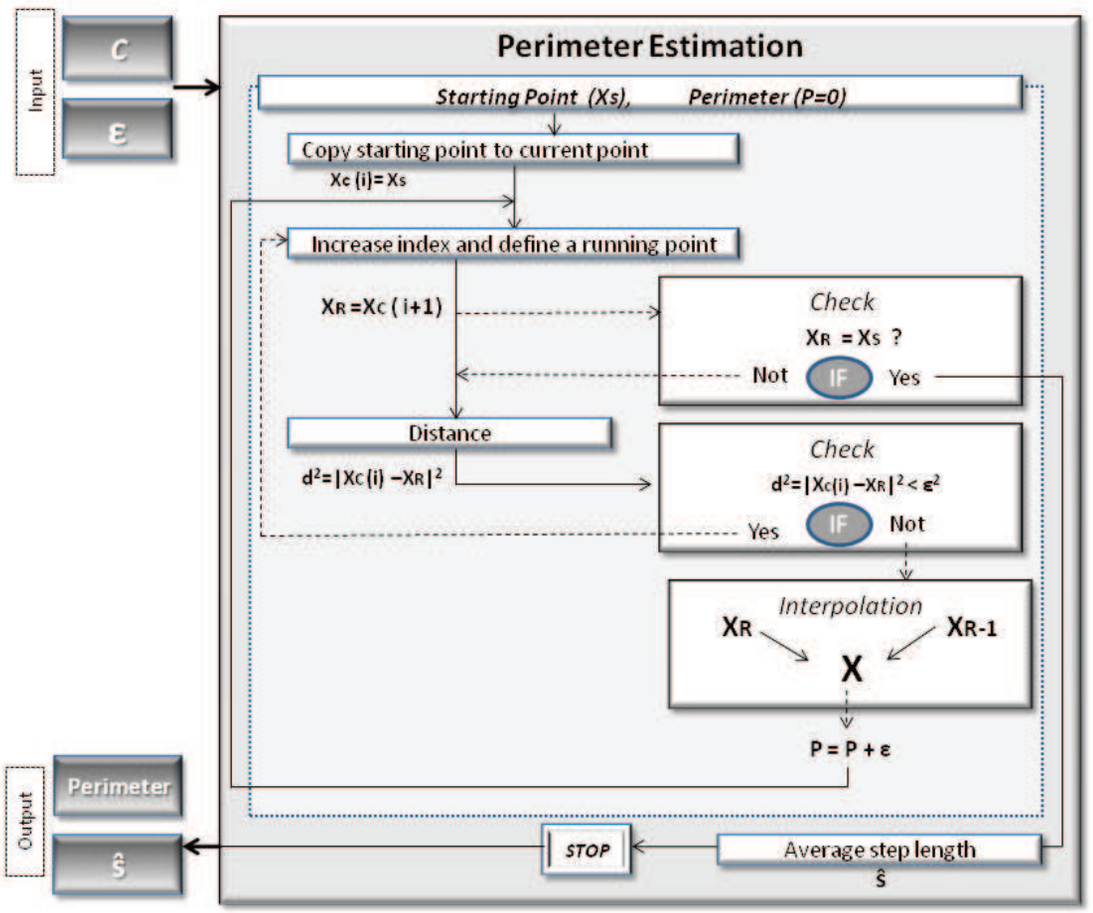


Figure 5. Perimeter estimation by EXACT method flowchart: an arbitrary *starting point* $S\ (x_s, y_s)$ on the boundary line is chosen and copied in a new variable, which is called *current point* $C\ (x_c(i), y_c(i))$. The index i runs through the total number of coordinate points and is iteratively increased defining the *running point* R with coordinates (x_R, y_R). The distance d between S and R is calculated and a check on d when smaller than a fixed step length ϵ is done. The process is repeated until a boundary point whose distance from (x_s, y_s) is larger than the step ϵ is reached. The exact position of the next pivot point (x, y) on the boundary line is determined by interpolating the two consecutivepoints (x_R, y_R) and (x_{R+1}, y_{R+1}).
The point (x, y) becomes the new starting point in order to calculate the next pivot point and so on, until the initial starting point S is reached.

The perimeter length of the polygon is found by adding the final incomplete step length to the sum of the other step lengths needed to entirely cover the boundary.

The procedure is then repeated for different step lengths[15].

The results, i.e., perimeter lengths versus step lengths, are plotted on a log-log Richardson's Plot. From the slope of the fitting line on the Richardson's plot we obtain the fd of the examined boundary[1, 4, 12, 16–21]

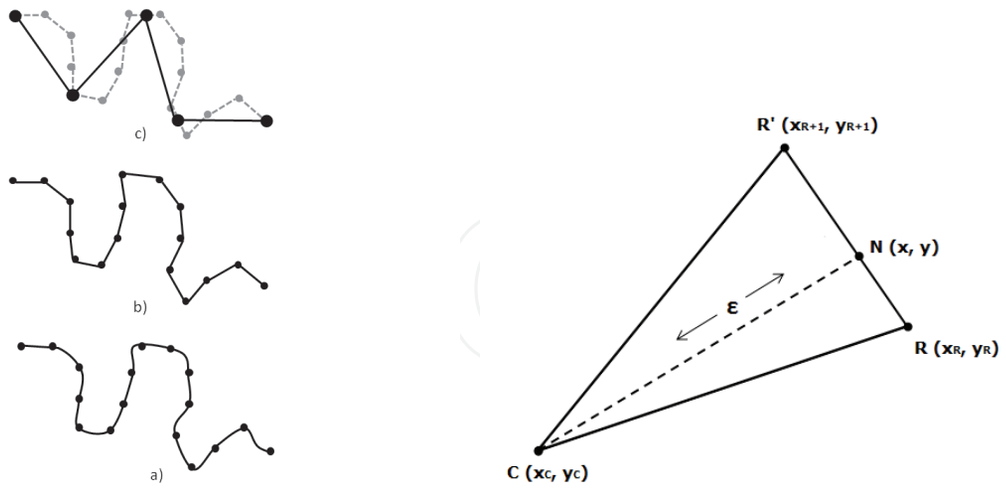


Figure 6. a) The piece-wise linear assumption (a) (b) and the EXACT algorithm (c); b)Geometric EXACT interpolation scheme, with S starting point given by (x_C, y_C) coordinates, R and R' two consecutive boundary running points respectly given by (x_R, y_R) and (x_{R+1}, y_{R+1}) coordinates, N new current point obtained by the interpolation between R and R' and ϵ a fixed step length.

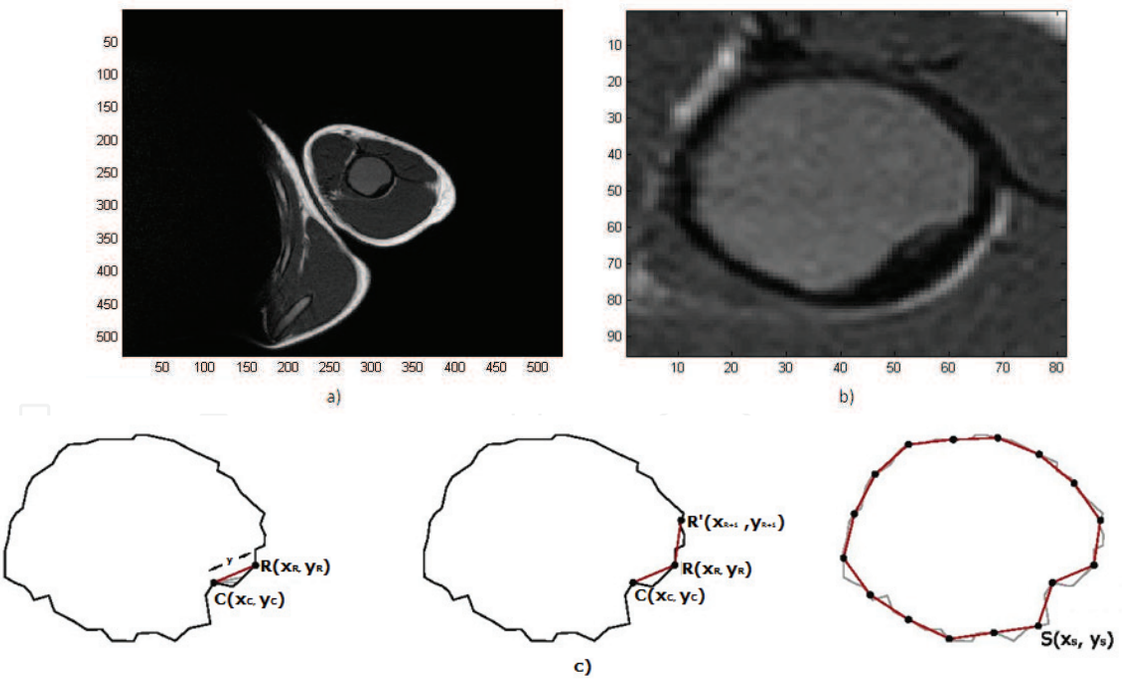


Figure 7. MRI image of an aneurismatic bone cyst (a), (b). Walking technique applied to an aneurysmatic bone cyst boundary (c).

4.3. Box-counting algorithm

The Box-counting algorithm implementation of box-counting method relies on the basic idea of covering a given digital binary image with a set of measuring boxes of sizes S and then to count the number of boxes which actually contain the image.

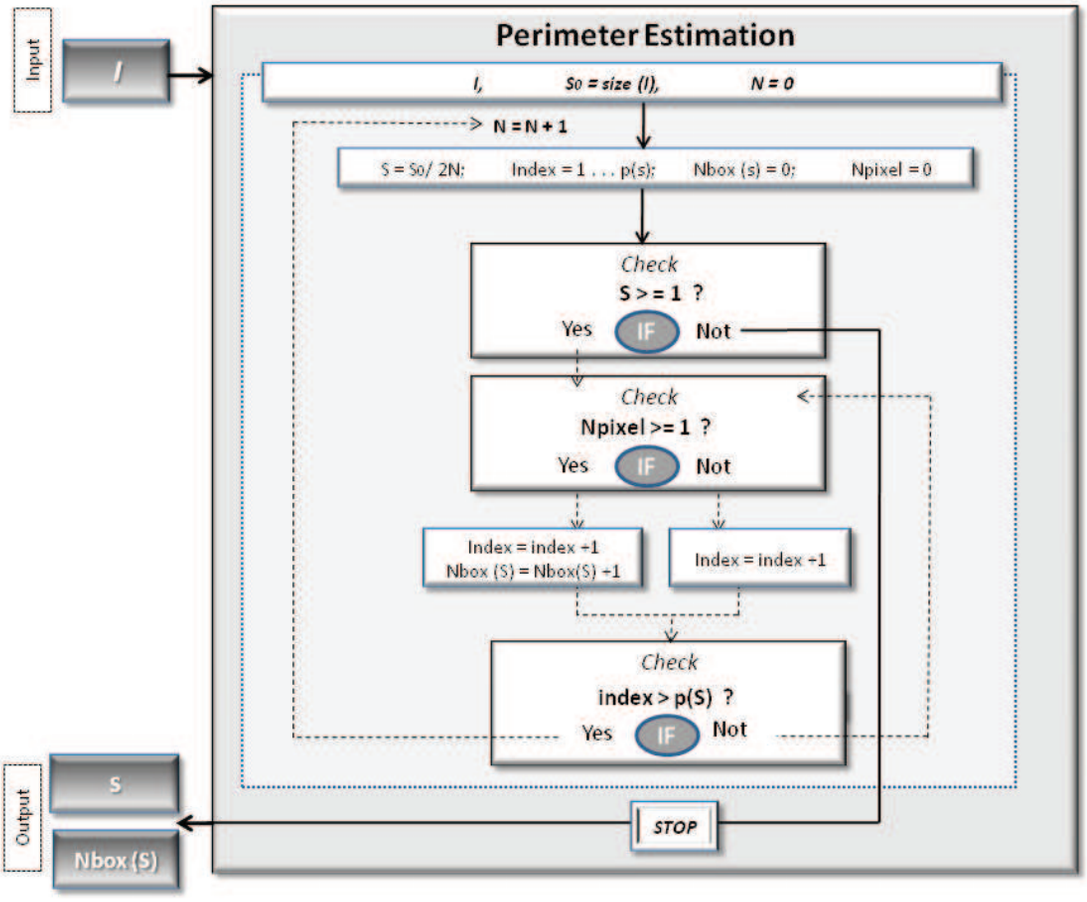


Figure 8. The Box-counting algorithm flowchart: given an image I, its size, S_0, is set as the maximum size from which the computer program starts to calculate the others decreasing box-sizes according to $S = S_0/2^N$. The S value has minimum value which is equal to the pixel size. The number p indicates the total number of box size S. The next step is a check on whether at least one pixel is in the box: if the box is non-empty, the check is stopped when one pixel is found. The procedure continues until the maximum index $p(s)$ is reached. Then the number $Nboxes(S)$ for a given size S is stored and the process restarts with a different box size. When the minimum box size is reached the program stops and gives the output variables of $Nboxes$ and the size value. Using the Eq. 8 the fractal dimension D can be estimate, from the least square linear fit.

Figure 8 shows the flow chart for box-counting fd estimation and for different box sizes. Moreover, since the procedure of size scaling ($S = S_0/2^N$ with N number of iterations) may be not always applicable to any image matrix size, image padding with background pixels is performed.

Therefore the final image I has a dimension that is a power of 2. This can be easily implemented by using *padarray* matlab function.

4.4. Differential Box-counting algorithm (DBC)

The box counting method is an extremely powerful tool for fd computation; in fact, it is easy to implement as well as flexible and robust.

However, a major limitation lies on the fact that the counting process of nonempty boxes implies its use only for binary images rather than gray scale ones. An extension of the

standard approach to gray scale images is called the *Differential Box Counting (DBC)* and has been proposed in 1994 by N. Sarkar and Chaudhuri[8].

In the DBC method, a gray level image I is considered as a 3-D spatial surface with (x, y) denoting the pixels spatial coordinates and the third axis z the pixels gray level.

As for the standard box counting, the $M \times M$ image matrix is partitioned into non-overlapping $s \times s$-sized boxes, where s is an integer falling in the interval $[M/2, 1]$.

Then, the scale of each block is $r = s$. On each block there is a column of boxes of size $s \times s \times s'$, with s' denoting the height of a single box. Named G the total number of gray levels in I, hence s' is defined by the relationship $G/s' = M/s$[7].

Let numbers $1, 2, 3...$ be assigned to the boxes so to group the gray levels. Let the minimum and the maximum gray level of the image in the $(i, j)th$ grid fall in box number k and l, respectively.

The number of boxes covering this block is calculated as:

$$n_r(i, j) = l - k + 1 \tag{15}$$

In Figure 9 for example $s = s' = 3$, hence $n_r = 3 - 1 + 1$. Extending to the contribution from

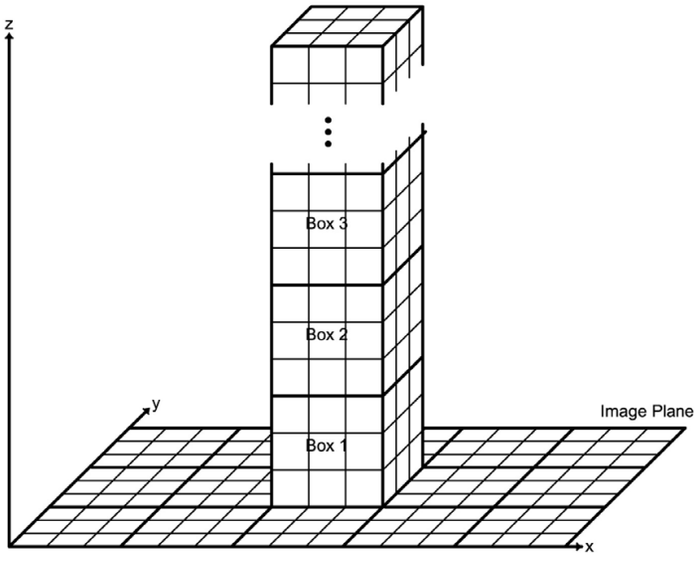


Figure 9. Example of DBC method application for determining the number of boxes of size $s \times s \times s$, when $s = 3$.

all blocks:

$$N_r = \sum_{i,j} n_r(i, j) \tag{16}$$

The Eq. 16 is computed for different box size s (so for different r) and the values of N_r are plotted versus the values of r in a log-log plot.

A matlab implementation of DBC can make use of functions such as *blockproc* or $colfilt$ in order to make the box partitioning and apply the Eq. 15.

The DBC procedure has some weak points in the method used to select an appropriate box height[7], since the values of s is limited to the image size and s' is limited by the number of blocks of size $s \times s$ in which the image is divided.

Secondly, the box number calculation may lead to overestimate the number of boxes needed to cover the surface. Let A and B be the pixels associated with the minimum and the maximum gray level of the block respectively, as is illustrated in Figure 10.

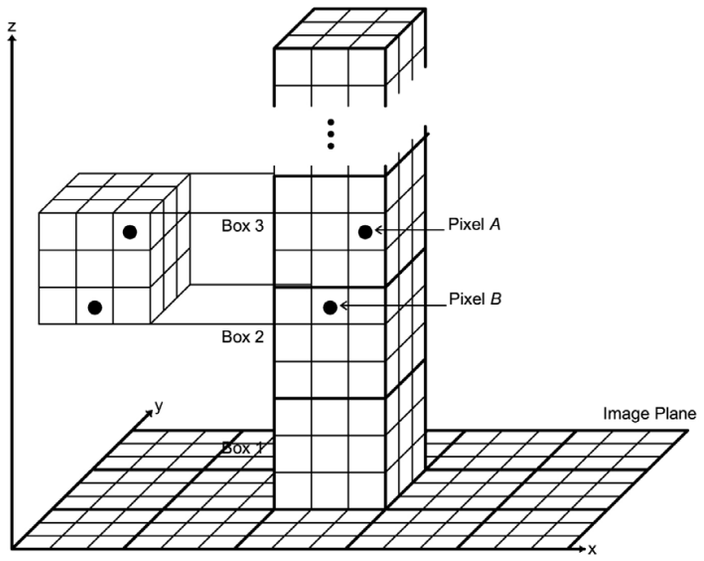


Figure 10. Example of DBC method application with boxes of $s \times s \times s$, when $s = 3$. The two pixels A and B, denoting the maximum and the minimum gray levels of the block, are assigned in two differents boxes, having distance in eight direction smaller than the box size $s = 3$.

According to DBC procedure, the two pixels are assigned in boxes 2 and boxes 3. The distance between A and B is smaller than 3, which is the size of the box.

Hence, when calculating Eq. 15, the block can be covered by a single box but its pixels with minimum and maximum gray levels fall into two different boxes.

To solve the aforementioned problems some modifications was proposed by J. Li, Q. Du and C. Sun[7]. Given a digital image I of size $M \times M$, a new scale r is defined instead of r, i.e. $r' = r/c$ where $c1$ is a positive real number.

In particular, let μ and σ be the mean and the standard deviation of I respectively. Hence, if the greater part of image pixels fall into the interval of gray level within $[\mu - a\sigma, \mu + a\sigma]$, where a is a positive integer, the height of the boxes is given by:

$$r' = \frac{r}{1 + 2a\sigma} \tag{17}$$

If dz is the height of the boxes in the direction of z, the number of the column of boxes on a single image block correspond to the integer part of $(dz/r' + 1)$ instead of $(dz/r + 1)$ as in the original DBC method. Thus, since $r' < r$, the residual part of dz/r' is smaller than that of dz/r.

As a result, the errors introduced using r' are smaller than in the original DBC method. A box with smaller height is chosen when a higher intensity variation is present on the image surface. So the improved method uses, in general, finer scales to count[7].

Moreover, the use of dz instead of z to count the number of boxes leads to the following modification of Eq. 15:

$$n_r = \begin{cases} ceil(\frac{l-k}{r'}), \\ 1, \qquad\qquad l = k \end{cases} \tag{18}$$

with ceil(.) denoting the function rounds the elements of the quantity into (.) to the nearest integers greater or equal to it.

Eq. 18 relies a new way to count the number of boxes that cover the $(i, j)th$ block surface in which the boxes are assigned to the minimum gray level to the block rather than gray level 0[7].

As an example, suppose that the $(i, j)th$ block is covered by a column boxes with the size $3x3x3$. If the pixels A and B represent the maximum and the minimum gray levels of the block, the two pixels will be assigned as in Figure 10.

According to Eq. 18 the number of counted boxes is $n_r = 1$, which is exactly the number of boxes covering the block.

As in standard box counting method, after having determined the number nr(i,j) for each block, the total number of boxes N_r covering the full image surface is computed for different scales r. Plotting the linear fit of log N_r versus the log r (Richardson's plot) the fd is finally estimated.

5. Applications and discussion

Each described method has been implemented in Matlab 2010a and applied to either well-known fractals or biomedical images.

The results on the hand and dividers methods are shown in the table 1. The computed values are also compared to the theoretical fd values. The computational time for a 2.50 GHz 5i CPU is also shown.

The value ranges for the step size are not displayed but they were automatically chosen based upon the computation of the structure's maximum caliber diameter which is defined as the major axis of an ellipse in which the structure can be embedded. The range was then running from the 40% of the maximum caliber diameter to the minimum step defined as the maximum distance between any two contiguous border points.

In practice, both EXACT and HYBRID methods computed the different step sizes by scaling each time the maximum step by a^k with k the number of the iteration. The chosen value of $a = 1.2$ is a compromise between a sufficient number of fitting points and the need to avoid too small variations of the step size so to duplicate perimeter estimation. The latter usually occurs in HYBRID method for it hits the same current points if the step does not vary enough in two consecutive iterations.

The parameter's estimation uncertainty is also shown in the table 1; that is calculated from the fitting accuracy based upon standard linear regression.

The number of data points used in the Richardson's plot was about 60 and two examples of that computation using EXACT and HYBRID are shown in Figure 12.

On the table 2 the computation results for the box counting method are also shown. The type of the displayed values are similar to the previous ones with the exception of Box counting uncertainty. In fact, the way an image can be partitioned into several boxes may affect the final computation of the number of nonempty boxes.

To investigate the variability of the fd for different box partitioning layouts, random box subdivisions have been applied. Therefore, the results on the table 2 show the standard deviation of the different computed fds and the mean values for each fractal at issue. In general, that variability is more pronounced in images having rougher resolution.

Fractal	fd_{theo}	fd_{exp}	*Time* (sec)	*BC error*	*Image size*
Apollonian Gasket	1.3057	1.408	1.5	0.001	2000×2000
Sierpinski	1.5849	1.587	0.3	0.005	1000×1000
Dragon	2.0000	1.747	7.2	0.006	3670×3978
Hexaflake	1.7719	1.640	1.6	0.011	1050×1050

Table 1. Tabular of results for box counting method application.

Fractal	fd_{theo}	fd_{exp}	*Time* (sec)	*BC error*	*Vector size*
Twin Dragon Hybrid	1.5236	1.466	8.6	0.006	117005
Twin Dragon Exact	1.5236	1.465	11.5	0.006	117005
Dragon Hybrid	1.5236	1.474	11.1	0.005	115665
Dragon Exact	1.5236	1.462	12.8	0.004	115665
Koch Hybrid	1.2619	1.276	31.2	0.004	786433
Koch Exact	1.2619	1.260	154.9	0.003	786433
Gosper Hybrid	1.1292	1.133	3.8	0.001	23280
Gosper Exact	1.1292	1.128	4.7	0.001	23280

Table 2. Tabular of results for walking-based methods application.

In general, the EXACT and the HYBRID methods appeared to be more precise than the box counting method but on the other hand they have a less wide range of applicability. However, this is also the reason of the fortune of the box counting methods compared to the others. Also, HYBRID technique is computationally less expensive than EXACT especially when the number of border points is quite large. The use of a variable step length which can be shorter or longer than the fixed step size leads to a larger variability and so to a Richardoson's plot having a less accurate fitting. That has effects on the uncertainties of the parameter to estimate. Because of that, a more careful choice of the step size range is needed in the case of HYBRID method.

Importantly, it is quite clear that the choice of the starting point may also affect the perimeter value as the following currents points will depend upon this. A test on 80 random starting points for the Gosper Island fractal revealed that the fd computation performed with the HYBRID method appeared to be more stable than the one with EXACT.

As for walking method, in box counting the process of scaling from the maximum box size is limited by the pixel size so in principle a gross resolution might be the reason of a bad estimate of fd. It is noteworthy that the tests performed do not show any correlation between resolution and fd accuracy; that may be also caused by the fact that some fractals such as dragon does not reproduce the real fractal at small scales.

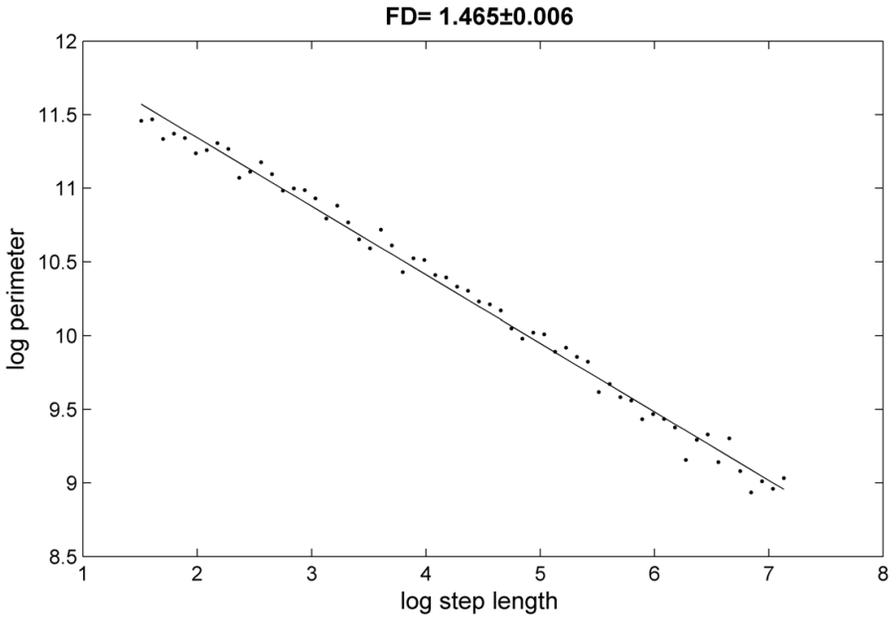


Figure 11. EXACT method apllied to the twin dragon fractal: Richardson's Plot.

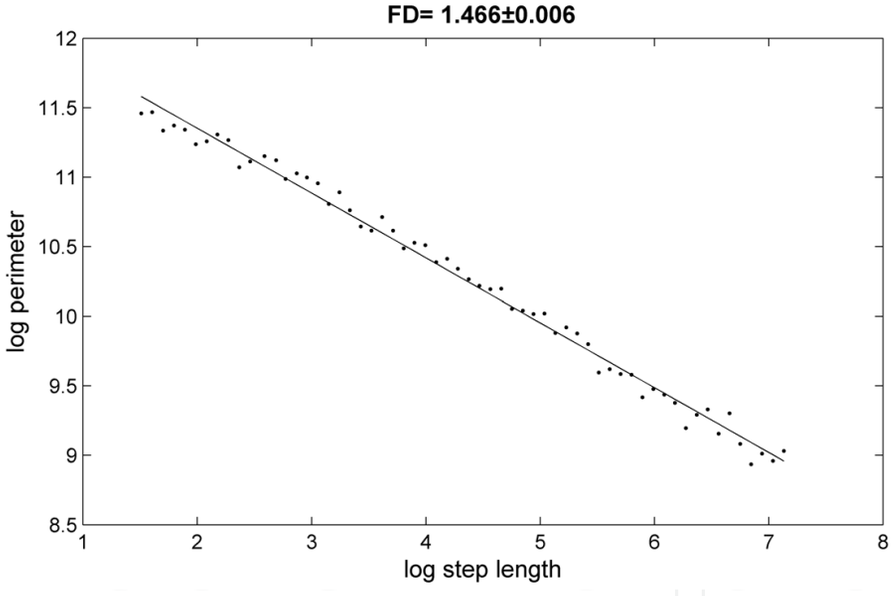
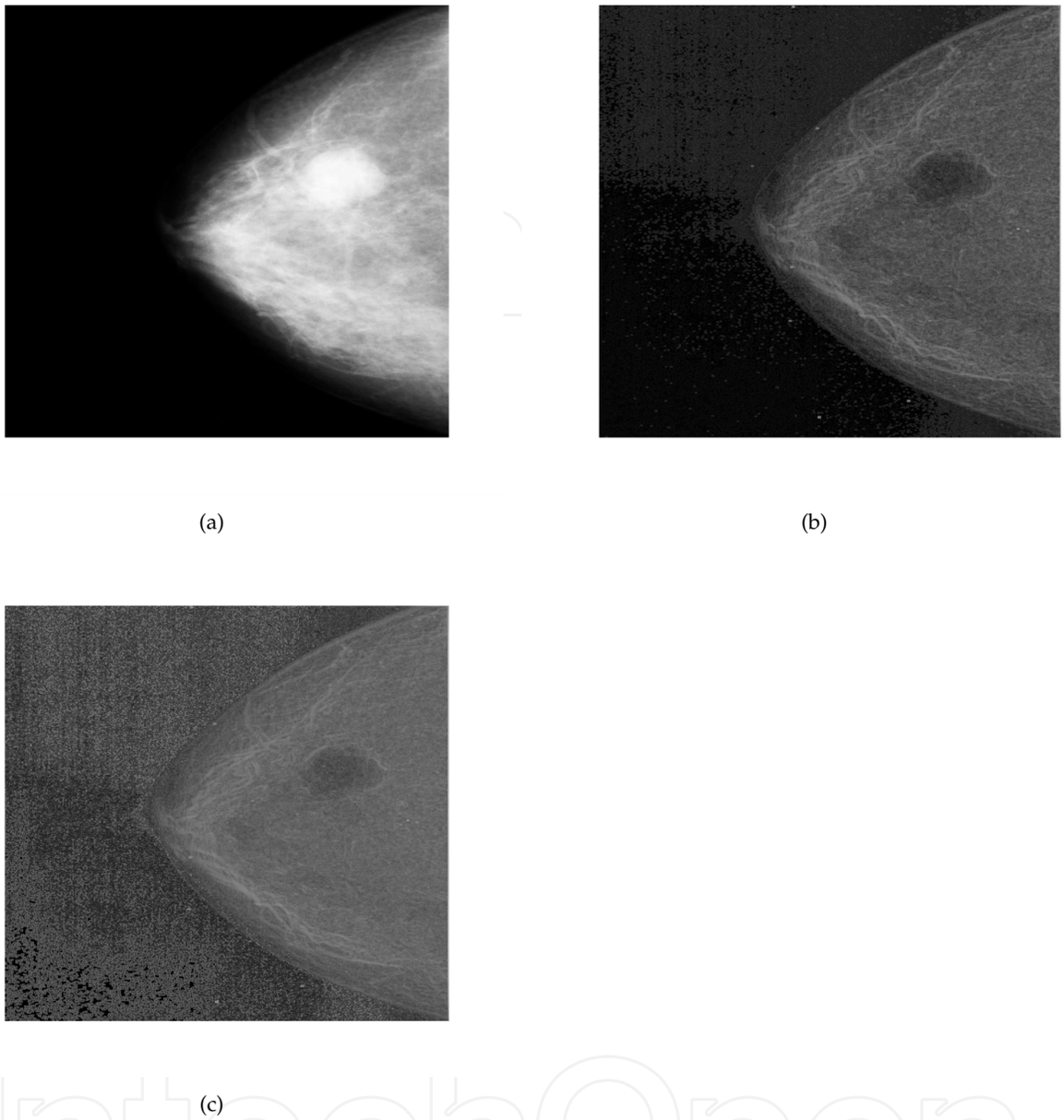


Figure 12. HYBRID method apllied to the twin dragon fractal: Richardson's Plot.

An application of the DBC method on a x-ray image is also shown in Figure 13 where breast cancer mammography image has been processed. The method uses a sliding technique as implemented in *blockproc* or $colfilt$ matlab functions so to produce an image rather than a single fd value as previously described.

The second DBC method shows higher contrast in the area of the cancer and consequently lower fd values. Due to the enormous amount of linear fitting performed for an image size of 3450×3100 the computational time reached 15 minutes.



(a) (b)

(c)

Figure 13. High resolution mammography image (a); fd recostruction image by standard Differential Box Counting (DBC) (b); fd reconstruction image by modified DBC (c).

6. Conclusions

In this chapter some of the most widely used and robust methods for fractal dimension estimation as well as their performances have been described. For few of them a detailed description of the algorithm has been also reported to make much easier for a beginner to start and implement his own Matlab code. Computational time is not excessively long to necessitate compiled functions such as C-mex files but that can be an advantage when using very high resolution images. The use of the described algorithms is obviously not restricted to the sole field of the image processing but it can be applied with some changes to any data analysis.

Author details

Antonio Napolitano, Sara Ungania and Vittorio Cannata
Department of Occupational Health and Safety, Medical Physics, Bambino Gesù Children's Hospital, Rome, Italy

7. References

[1] J. Orford and W. Whalley, *The use of the fractal dimension to quantify the morphology of irregular-shaped particles*, [Sedimentology, 30, 655-668], (1983).
[2] J. Keller et al., *Texture description and segmentation through fractal geometry*, [Computer Vision, Graphics and Image Processing, 45, 150-166], (1989).
[3] F. Hausdorff, *Dimension und ausseres Mass*, [Math. Annalen 79, 157] (1919).
[4] J. Theiler, *Estimating fractal dimension*, [7, 6/June 1990/J. Opt. Soc. Am. A] (1989).
[5] B. Mandelbrot, *The Fractal Geometry of Nature*, W.H.Freeman and Company, New York, (1983).
[6] S. Deepa,T. Tessamma *Fractal Features based on Differential Box Counting Method for the Categoritazion of Digital Mammograms*, [International Journal of Computer Information System and Industrial Management Applications, 2, 011-019], (2010).
[7] J. Li, Q. Du, *An improved box-counting method for image fractal dimension estimation*, [Pattern Recognition, 42, 2460-2469], (2009).
[8] N. Sarker, B. B. Chaudhuri, *An efficient differential box-counting approach to compute fractal dimension of image*, [IEEE Transaction on Systems, Man, and Cybernetics, 24, 115-120], (1994).
[9] A. P. Pentland, *Fractal-based description of natural scenes*, [IEEE Transaction on Pattern Analysis and Machine Intelligence, 6, 661-674], (1984).
[10] L. F. Richardson, *Fractal growth phenomena*, [Ann. Arbor, Mich. : The Society 6, 139] (1961).
[11] B. Mandelbrot, *How long is the coast of Britain? Statistical self-similarity and fractional dimension*, [Science 155, 636-638] (1967).
[12] N. Clark, *Three techiques for implementing digital fractal analysis of particle shape*, [Powder Technology 46, 45] (1986).
[13] M. Allen, G. J. Brown, N. J. Miles, *Measurement of boundary fractal dimensions: review of current techinques*, University of Nottingham , UK (1994).
[14] M. Allen, *Ph.D. Thesis*, University of Nottingham , UK (1994).
[15] P. Podsiadlo, G. W. Stachowiak, *Evaluation of boundary fractal methods for the characterization of wear particles*, [Wear 217(1) , 24-34] (1998).
[16] J. D. Farmer, E. Ott. and J. A. Yorke, *The dimension of chaotic attractors*, [Physica 7D, 153] (1983).
[17] B. Kaye, *A Random Walk Through Fractal Dimension*, [VCH Verlagsgesellschaft] (1986)
[18] L. Niemeyer, L. Pietronero and H. J. Wiesmann, *Fractals dimension of dielectric breakdown*, [Phys. Rev. Lett. 52, 1033] (1984).
[19] H. Schwarz and E. Exner, *The implementation of the concept of fractal dimension on semi-automatic image analyzer*, [Powder Technology, 27, 207-213], (1980).
[20] J. Russ, *Fractals surfaces*, [Plenum Press] (1994).
[21] H. von Koch, *Sur une courbe continue sans tangente, obtenue par une construction geometrique elementaire*, Archiv for Matemat., [Astron. och Fys. 1, 681-702] (1904).