

jFrad - a tool to analyze fractal dimension of
objects in binary images

Zhou Ji ©

February 11, 2009

Contents

I	Background and Mathematics	3
II	Method Details	3
1	Methods based on simple pixel borders	4
1.1	Dilation	4
1.2	Euclidean Distance Map	5
1.3	box-counting	5
2	Methods based on traced border	5
2.1	Fast	6
2.2	Fast (Hybrid)	6
2.3	Parallel Lines	6
3	Mass-based methods	6
3.1	Cumulative Intersection	7
3.2	Mass Radius (Long)	8
3.3	Mass Radius (Short)	8
4	Corner-based methods	8
4.1	Corner count	9
4.2	Corner perimeter	9
III	User Instruction	10
5	support image formats	10
6	pre- and post- processing	10
6.1	preprocessing	10
6.2	postprocess	10
IV	History	10

The purpose of jFrad is to evaluate fractal dimension of objects in binary images.

Part I

Background and Mathematics

jFrad is meant to be an improved re-implementation of BIP IV.

Most of techniques to be implemented are described in [8] and [9], but neither of those two sources adequately documented the algorithm details. One of key reference, Matthew Wartel’s web site at Cornell School of Civil and Environmental Engineering about the macro implementation of nine algorithms [16] does not exist any more, although that web site is referenced not only in my thesis [8], but also in a couple of other wbe sites [10, 6], which are still online. Therefore, it is necessary to describe more details here.

‘Fractal dimension’ can be used in a more general sense referring to any of the dimensions commonly used to characterize fractals [18]. Most commonly it is used in its narrower meaning, i.e. capacity dimation [17]

$$d_{\text{capacity}} \equiv - \lim_{\epsilon \rightarrow 0^+} \frac{\ln N}{\ln \epsilon}, \quad (1)$$

where N is the number of element forming the finite cover and ϵ is the size of each element. For example, considering a straightline, the fraction is constant 1 regardless of the size of element; for a square, the fraction is constant $\frac{1}{2}$: it takes $2 * 2 = 4$ squares with the side $a/2$ to cover the same area that is covered by one square with the side a .

Objects in a pixel-based image are in fact never ‘fractal’ in its strict sense because of the finite size of the pixels [1, 2, 7]. In a pixel-base image, pixel is the smallest scale we can reach and the mathemmedical limit does not exist. However, if we believe that the images represent some real objects that are in fact fractals, we can evaluate the objects’ (or their borders’) fractal dimension using the shapes represented by the pixel values. The relationship between such evaluation and the real fractal dimension is not mathematically rigorous considering the definition above (Eq. 1) is based on the limit when $\epsilon \rightarrow 0^+$.

That is enough for the theoretical clarification. Let us thereafter focus on the actual algorithms.

Part II

Method Details

The common assumption is that we are analyze binary (black-and-white) pixel-based images, in which one color, typically white, represent the shape of some

physical ‘objects’, e.g. bacterial colonies in biofilm. Eleven methods are implemented in jFrad to evaluate fractal dimension of those objects. The fractal dimension to be evaluated is with regard to the border of such objects, not the volume or the mass of the object. The latter is basically 2-dimensional except for the the border area.

The eleven methods can be categorized in four groups. Group 1 is based on the purely pixel-represented border. In the other words, we don’t differentiate individual objects. Group 2 is based on the so-called traced-border, meaning we do consider each object separately and the sequential order of the border pixel is used in the procedure of evaluation. Group 3 is the mass-based methods. Although the mass here actually refers to the mass of the border, this group of the methods treats the border (represented either as simply a set of pixels or a traced pixel sequence) as a solid object instead of merely a digital representation of the width-less separation between the real object and the surrounding. The last group is based on the concept of ‘corners’ along the border.

All the methods implemented in jFrad and described here follow the same pattern: for a sequence of an incremental variable r , we calculate certain image measurement A for each value of r ; then we use linear regression to obtain the slope of the $\log - \log$ plot of A vs. r ; fractal dimension is then evaluated by some simple conversion from the slope.

In the literature about fractal dimension, the phrase like ‘the slope of the $\log(A) \sim \log(r)$ plot’ is often used. This term of *slope* is used in the description of all the methods in this document, so we need to clarify the definition and come up with a simple notation as following:

$$slope \sim \frac{y}{x} \tag{2}$$

is defined as, for a sequence of y : y_1, y_2, \dots, y_n and a sequence of x : x_1, x_2, \dots, x_n , the regression coefficient b for the linear least squares fitting[19] of y_i versus x_i : $\hat{y}_i = a + bx_i$.

Other common notations:

- D_f : fractal dimension

1 Methods based on simple pixel borders

All three methods calculate a slope, then $D_f = 1 - slope$. They are also called length-related methods [14].

1.1 Dilation

Algorithm specific:

- r the radius of dilation
- A : the area of dilated border

- slope $\sim \log A / \log r$

- $D_f = 2 - \text{slope}$

$d = 2r$ is called ‘kernel diameter’.

This method uses the dilation of the borders and is developed by Flook. See [14] chapter 6 p. 175. This method is widely used in various applications [12].

for straight line, A/d is constant, so the slope is 0 and $d=1$

also notice that when we assume $\log(A/d)$ vs $\log(d)$ is linear then A/d vs. d is not exponential, not linear. so A vs d is exponential as well

1.2 Euclidean Distance Map

Algorithm specific:

- d the shortest Euclidean distance to the border pixels
- A : the area within that distance
- slope $\sim \log A / \log d$
- $D_f = 2 - \text{slope}$

The direct difference between this method and dilation method is that the distance here is based on the original border pixels. In dilation method, the new area at each step is based on the previous step, not the original starting status.

1.3 box-counting

Algorithm specific:

- a the box size - the number of pixels on each size of the square area
- N : the number of boxes to cover all border pixels
- slope $\sim \log N / \log a$
- $D_f = -\text{slope}$

This method is also known as grid method. It is highlighted in works like [13].

2 Methods based on traced border

This group of methods (fast, fast (bybrid), and parallel lines) are all based on traced border. See [15, 3, 4].

They seems very stable in terms of keeping the result within 1 and 2.

2.1 Fast

Algorithm specific:

- a the interval between pixels
- L : the perimeter connecting the intervalled pixels
- slope $\sim \log L / \log i$
- $D_f = 1 - \text{slope}$

BIP implementation appearantly has some bug.

2.2 Fast (Hybrid)

Algorithm specific:

- a the distance interval between pixels along the traced border
- L : the perimeter connecting the intervalled pixels
- slope $\sim \log L / \log i$
- $D_f = 1 - \text{slope}$

This method is similar ro Fast method except that each pixel connects to the closest pixel at least distance a away. By contrast, in Fast method, the pixel connects to the next pixel by skipping $a - 1$ pixels in between.

2.3 Parallel Lines

Algorithm specific:

- d the interval between parallel horizontal lines
- L : the perimeter connecting the pixels on those lines along the traced border
- slope $\sim \log L / \log d$
- $D_f = 1 - \text{slope}$

This method is also similar to Fast or Fast (Hybrid) method, but each segment along the perimint is more likely to be different.

Conceptually, the parallel lines could be chosen as vertical lines instead horizontal. The choice is arbitrary.

3 Mass-based methods

This group includes three methods. See [14] section 6.1.7.

This group is based on the pixel boundary too, but treat this pixel boundary as mass instead of 'boundary'. The real difference is what kind of operation we apply on it: intead of dilation, or EDM, we ,,,,

3.1 Cumulative Intersection

Algorithm specific:

- r : radius of circle
- A : the area of object within the circle
- slope $\sim \log A / \log r$
- $D_f = slope$

as convented here: area is the same as the pixel count slope is represented as a fraction for convenience

This method in BIP is implemented with bug. The result is my dissertation shows some indication. Tested with THE test image (...11), the fractal dimension is 0.8141.

This method was original used in the 'flower'-shaped objects like neurons[13], or cross-section of something, meaning it has a core and the algorithm starts reasonably from radius 0 or very small radius. IN the case of the 'block-like' object (like in the test image), we only look at the border for fractal dimension because otherwise we should always have a number very close to 2.

Conceptually, the fractal dimension from the objects in the images can only be between 1.0 and 2.0. Any method is expected to get a value within this range. If the 'object' is simple straight line, Cumulative Intersection method will get 1.0. If the 'object' is a completely filled area, the method will return 2.0. See Figures ...

Is it possible for Cumulative Intersection to return something out of [1.0, 2.0] and when? When the circle goes beyond the object, then the cumulative intersection will not increase, the slope and the calcaulted fractal dimension will go down and go under 1 and could approach to 0. Consequently, the method should stop once the circle is beyond the entire object. On the other end, if there is a hole inside the 'object', the increment of cumulative intersection count will be faster than entreyly filled area and result in a calculated fractal dimension that is larger than 2. There are two possible way to avoid this unlogic results: add one pixel for a given circle that there is no pixel at all - on the surface, it also artifically void logirithm of 0; or we can start the method only when it actually intersection the objects. There are two shortcomings for the second soluation: one, it can still be larger than 2.0 if we have only a few steps to go and they are in really large slope. For example, the pixel count is 1, 10, 1000, 1000,000,000. That is threoretically possible considering we are not starting from the origin (usuallt the object's center). Two, in may cases we have multiplebbject in the images, the intermediate no pixel area (ring) exists so it will casue the calculate fractal dimension to be either larger or smaller than the actual onject's fractal dimension in a misleading way. jFrad's approach is thus designed: each object is handled separetae; each object's iteration start from ooriginal and one artifiical pixel is added if there is an interal hole; then the ultimate result is a average of the fractal dimensions of all object weighed on each's pixel count.

3.2 Mass Radius (Long)

Algorithm specific:

- r : radius of circle around each border pixel
- A : the area of object (total pixel count) within the circles
- slope $\sim \log A / \log r$
- $D_f = slope$

Conceptually, this method does not need traced borders. However, if starting with both the pixel array and the traced list of the borders, the algorithm can be implemented more efficiently.

3.3 Mass Radius (Short)

Algorithm specific:

- r : radius of circle around chosen border pixels
- A : the area of object (total pixel count) within the circles
- slope $\sim \log A / \log r$
- $D_f = slope$

This method is the same strategy as the long version except that only a subset of pixels are used as the centers to evaluate mass radius. The long version uses all the pixels. [8] mentioned using random chosen pixels. BIP was not implemented that way. In stead, the subset is chosen as one out of every 5 pixels along the traced borders. jFrad is implemented the same way considering the random version does not have convincing reason to be more accurate or efficient, but complicates the algorithm a little. As future enhancement or alternative, we could (1) add the choice of the random version, and (2) provide a adjustable parameter, probably called ‘pixel step size’, instead of using constant 5. This method needs to use traced borders generally. When the pixel array is available, it could be easier, maybe not more efficient as well, to implement the algorithm.

4 Corner-based methods

The only source of the two methods in this group is a brief description in [8] and the source code of BIP. They are based on the traced borders.

These two methods appear to be more sensitive to the selection of control parameters (the progressive parameters) and are more like to be out of valid range [1, 2].

The concept of ‘corner’ is defined based on height, which is at most of 1, the distance between two adjacent pixels assuming 4-connection is used as in the implementation of all the other methods in jFrad.

4.1 Corner count

Algorithm specific:

- h : the minimum height to define a corner
- N : the number of the corners
- $\text{slope} \sim \log N / \log h$
- $D_f = -\text{slope}$

A corner is defined as the pixel along the border, or vertex, whose height is at least h . The height of vertex P_i is defined as the distance of vertex P_i to the line P_0P_{i+1} , where P_0 is a reference point and P_{i+1} is the following pixel along the traced border. After a corner is found, it serves as the reference point to search the next corner along the traced border.

This method is based on the assumption that on a fractal, the largest size of the skinks or 'corners' we are considering, S , is related to the number of such corners, n , in this way:

$$O\left(\frac{1}{S}\right) \leq n \leq O\left(\frac{1}{S^2}\right) \quad (3)$$

This translates to $-2 \leq \frac{\log P}{\log S} \leq -1$. So the fractal dimension is evaluated as $-\text{slope}$, which will be between 1 and 2. The actually pixel-represented objects may not satisfy the above assumption, though. Consequently, it is still possible to get a result out of the range $[1, 2]$.

This relationship is meaningful when we look at limit of $S \rightarrow 0^+$, definitely not when $S \rightarrow \infty$. So the larger S should be considered less than the smaller S .

Much of the discussion of this method also applies to Corner (Perimeter) method.

4.2 Corner perimeter

Algorithm specific:

- h : the minimum height to define a corner
- L : the perimeter connecting all the corners
- $\text{slope} \sim \log L / \log h$
- $D_f = 1 - \text{slope}$

This method is based on the assumption that on a fractal, the largest size of the skinks or 'corners' we are considering, S , is related to the perimeter connecting such such corners, P , in this way:

$$O(1) \leq P \leq O\left(\frac{1}{S}\right) \quad (4)$$

This translates to $-1 \leq \frac{\log P}{\log S} \leq 0$. So the fractal dimension is evaluated as $1 - \text{slope}$, which will be between 1 and 2.

Part III

User Instruction

How to start More ... batch processing

5 support image formats

jpeg, tiff, bmp, gif, png

6 pre- and post- processing

6.1 preprocessing

binarized, cleaning (removal of particles, filling holes), border finding and tracing
Border tracing is an important step in pre-processing [5, 11]

6.2 postprocess

There is no real postprocessing per se. jFrad does evaluate several other numerical properties of the images as BIP does.

Part IV

History

The author of jFrad participated developing BIP [8, 9], a biology image processor whose main purpose is to evaluate fractal dimension with various methods. The bugs and other issues (e.g. dependency on the three-party library lacking of source code and documentation) made it more reasonable to re-implement the great collection of fractal dimension evaluation methods in BIP than to support it. BIP was implemented in out-of-date version of Microsoft Visual Studio and its MFC technology.

For the above reason, it was not a goal for jFrad to match the results from BIP. Nevertheless, the results from both systems are listed together for a particular test image, fj05090111.tif, just for reference (see table 1). Default parameters are used in both systems.

BIP provided some other image quantitations that are not implemented in jFrad because they're unrelated to fractal dimension. They include: 1. areal porosity, which is trivial and basically covered by the pixel statistics; 2. diffusion

Method	jFrad	BIP
Dilation	1.3822222992792859	1.3589
Euclidean Distance Map	1.3548576074591177	1.1983
Box Counting	1.240308460064921	1.1927
Fast	1.2381078379049395	1.9809
Fast (Hybrid)	1.3506832017067303	1.2833
Parallel Lines	1.3641075349514662	1.3787
Cumulative Intersection	1.8705546183854567	0.8141
Mass Radius (Long)	1.3338819080104474	1.3686
Mass Radius (Short)	1.333280923671464	1.3657
Corner (Count)	1.3528653612994181	1.0272
Corner (Perimeter)	1.6229767731550138	1.2089

Table 1: Results from jFrad and BIP

distance (max and mean); 3. run length (horizontal and vertical, max and mean); and 4. textual measurement (entropy, inverse difference moment, and angular second moment). Textual analyzer is an interesting topic. See http://rsb.info.nih.gov/ij/plugins/download/GLCM_Texture.java.

Please send any comments, questions, and suggestions on jFrad to zhouji@acm.org.

References

- [1] M. ALLEN, G. J. BROWN, and N. J. MILES. Measurement of boundary fractal dimensions : review of current techniques. *Powder technology*, 84(1):1–14, 1995.
- [2] Paul Bourke. F d c - fractal dimension calculator, February 2003.
- [3] Dirk H. de Boer, Mike Stone, and Lucie M. J. Lvesque. Fractal dimensions of individual flocs and floc populations in streams. *Hydrological Processes*, 14(4):653–667, 2000. CP: Copyright 2000 John Wiley & Sons, Ltd. ON: 1099-1085 PN: 0885-6087 AD: Department of Geography, University of Saskatchewan, 9 Campus Drive, Saskatoon, Saskatchewan S7N 5AS, Canada; Faculty of Environmental Studies, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada DOI: 10.1002/(SICI)1099-1085(200003)14:4<653::AID-HYP964>3.0.CO;2-3 US: [http://dx.doi.org/10.1002/\(SICI\)1099-1085\(200003\)14:4<653::AID-HYP964>3.0.CO;2-3](http://dx.doi.org/10.1002/(SICI)1099-1085(200003)14:4<653::AID-HYP964>3.0.CO;2-3).
- [4] P. Dellino and G. Liotino. The fractal and multifractal dimension of volcanic ash particles contour: a test study on the utility and volcanological relevance. *Journal of Volcanology and Geothermal Research*, 113:1–18, 2002.

- [5] T.D. Haig and Y. Attikiouzel. An improved algorithm for border following of binary images. In *Circuit Theory and Design, 1989., European Conference on*, pages 118–122. The University of Wwestern Australia, Australia, 5-8 September 1989. I found the paper on the web. It is very useful for border tracing.
- [6] Michael D Higgins. Quantitative textural measurements in igneous and metamorphic petrology, 2008.
- [7] Herbert Jelinek, David Cornforth, and Leighton Weymouth. Fractop v0.3b - a software to analyze fractal dimension. <http://seal.itee.adfa.edu.au/s3165516/Fractop/>.
- [8] Zhou Ji. Quantitative analysis of biofilm images using fractal dimensions. M.s. in computer science, The University of Memphis, August 2000. Major Advisor: Dr. Giri Narasimhan.
- [9] Zhou Ji, Qichang Li, A. Heydorn, S. Molin, K. Mathee, and Giri Narasimhan. Quantitative analysis of pseudomonas aeruginosa biofilm images using fractal dimensions. In *Biofilms 2000 (international conference)*, Montana, 2000.
- [10] Chris Lucas and David Kirshbaum. Calresco.
- [11] Hassan A. Kingravi M. Emre Celebi and Jeongkyu Lee. Fast and accurate border detection in dermoscopy images using statistical region merging. In *Proceeding of SPIE Medical Imaging*, San Diego, CA, USA, Feb. 17 - 22 2007.
- [12] C. S. McLachlan, H. F. Jelinek, S. K. Kummerfeld, N. Rummery, P. D. McLachlan, P. Jusuf, C. Driussi, and J. Yin. A method to determine the fractal dimension of the cross-sectional jaggedness of the infarct scar edge. *Redox Report*, 5(2-3):119–121, 2000. no pdf.
- [13] Nebojsa T. Milosevic, D. Ristanovic, and J.B. Stankovic. Fractal analysis of the laminar organization of spinal cord neurons. *Journal of Neuroscience Methods*, 146(2):198 – 204, 2005.
- [14] Mustafa K. Khokha Philip M. Iannaccone. *Fractal Geometry in Biological Systems: An Analytical Approach*. CRC Press, 1996.
- [15] H. Schwarz and H. E. Exner. The implementation of the concept of fractal dimension on a semi-automatic image analyser. *Powder Technology*, 27(2):207 – 213, 1980. has pdf; fractal dimension.
- [16] Matthew Wartel. Fractal analysis macro, 2000. now gone.
- [17] Eric W. Weisstein. Capacity dimension. From MathWorld—A Wolfram Web Resource. <http://mathworld.wolfram.com/CapacityDimension.html>.

- [18] Eric W. Weisstein. Fractal dimension. From MathWorld—A Wolfram Web Resource. <http://mathworld.wolfram.com/FractalDimension.html>.
- [19] Eric W. Weisstein. Least squares fitting. From MathWorld—A Wolfram Web Resource. <http://mathworld.wolfram.com/LeastSquaresFitting.html>.