Efficient N-Dimensional surface estimation using Crofton formula and run-length encoding

Gaëtan Lehmann[1,2] and David Legland[3,4]

February 21, 2012

[1]INRA, UMR 1198 Biologie du Développement et Reproduction, Jouy-en-Josas, F-78350, France
[2]ENVA, UMR 1198 Biologie du Développement et Reproduction, Jouy-en-Josas, F-78350, France
[3]INRA, UMR 0782 Génie et Microbiologie des Procédés Alimentaire, Thiverval-Grignon, F-78850, France
[4]AgroParisTech, UMR 0782 Génie et Microbiologie des Procédés Alimentaire, Thiverval-Grignon, F-78850, France

Abstract

Unlike the measure of the area in 2D or of the volume in 3D, the perimeter and the surface are not easily measurable in a discretized image. In this article we describe a method based on the Crofton formula to measure those two parameters in a discritized image. The accuracy of the method is discussed and tested on several known objects. An algorithm based on the run-length encoding of binary objects is presented and compared to other approaches. An implementation is provided and integrated in the LabelObject/LabelMap framework contributed earlier by the authors.

Latest version available at the Insight Journal [http://hdl.handle.net/10380/3342]

Contents

1 Introduction

Surface area and perimeter are widely used parameters describing the size of objects observed in images, and are commonly used for computing various shape factors. Unlike the area that can be easily measured by counting pixels belonging to the object, measuring the perimeter is not as straightforward, and naive methods can lead to huge systematic errors [3, 5]. Usually, the countour of the object is extracted, then its length is measured. The same approach is usually applied to 3D images: the boundary of the 3D object is first reconstructed, e.g. by using marching cubes [6], then its surface area is computed by summing area of individual triangles.

The Crofton method is an alternative method that allows estimating the perimeter of 2D objects, the surface area of 3D objects, and more generally the $(d-1)$-dimensional measure of d-dimensional object boundary. It is based on counting intercept number of the object boundary with a set of isotropic test lines, and provide an unbiased estimate of the actual perimeter or surface.

The article first recalls the mathematical principles of d-perimeter measures in digital images using the Crofton method. An efficient implementation based on run-length encoding is then presented. The method is evaluated on various $2D$ and $3D$ synthetic shapes and compared with other methods. The effect on the roundness shape factor is also investigated.

2 Principles

2.1 Surface area and perimeter estimation

Perimeter measure of 2D objects, surface area measure of 3D objects and more generally $(d-1)$-surface measure of d-dimensional objects can be expressed in a unified formalism by using the Crofton formula. This formula consists in integrating the intercept number of the object with lines of various orientation and positions. Its expression in the general form is:

$$S^{(d-1)}(X) \quad = \quad \frac{d \cdot v_d}{v_{d-1}} \int_{\mathcal{L}^d} \chi(X \cap L)\, dL \tag{1}$$

where X is the structure of interest, v_d is the volume of the d-dimensional ball, $\mathcal{L}^d$ is the set of all lines in the d-dimensional space. The above integral is normalised such that the mass of lines hitting the unit ball equals v_{d-1}, the measure of the unit ball projection on a $(d-1)$-dimensional plane.

The Euler-Poincaré characteristic χ is equal to number of connected components of the intersection of X with a line L, or equivalently half the number of intersections of the boundary of X with the line.

For planar and 3D cases, Equation 1 can be rewritten:

$$P(X) = \pi \int_{\mathcal{L}^2} \chi(X \cap L)\, dL \quad (2)$$

$$S(X) = 4 \int_{\mathcal{L}^3} \chi(X \cap L)\, dL \quad (3)$$

2.2 Crofton formula in discrete images

The Crofton formula can be easily applied to discrete binary images [4, 5]. The integral over lines can be decomposed into an integral over a finite set of directions and an integral over all the lines parallel to a given direction.

For the planar case, the perimeter can be estimated by considering horizontal and vertical lines, i.e. two directions. An alternative is to use the diagonals, resulting in four directions. Perimeter estimate is then written as:

$$P(X) \simeq \pi \sum_k \frac{c_k}{\lambda_k} \chi(X \cap L_k) \quad (4)$$

where c_k is the discretization weight associated to direction k, λ_k is the density of discrete lines in direction k, and L_k is the set of all discrete lines in direction k.

The line density λ_k may vary according to the directions, due to image resolution or to the use of diagonals. It is computed as the ratio of the distance between two neighbor pixels with the area associated to a pixel. When only horizontal and vertical directions are used, associated weights c_k equal $1/2$. When diagonals are used, weights are obtained by projecting direction vectors on the unit circle, and by computing the relative fraction of circle associated to each direction (Fig. 1).

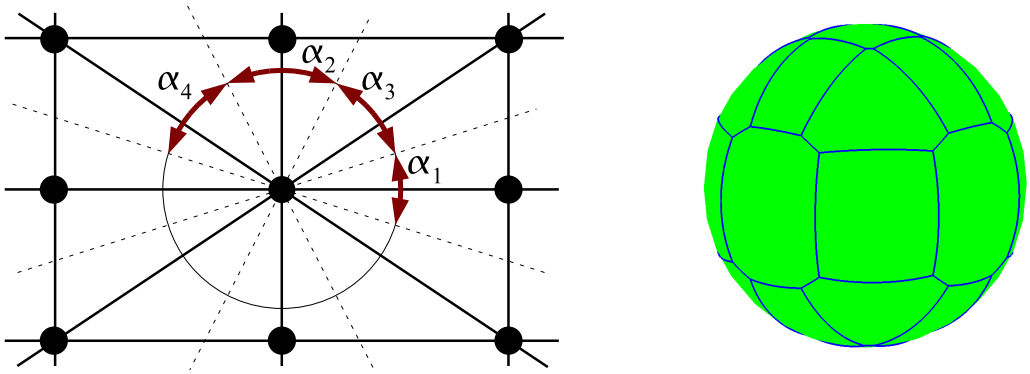

Figure 1: Computation of direction weights for rectangular grids. Left: planar case. Black dots correspond to pixel centers. Angular regions are computed between direction bisectors. Right: 3D case. The unit sphere is partitioned into spherical polygons corresponding to each of the 26 oriented directions.

In a similar way, surface area of $3D$ objects may be estimated using following approximation:

$$S(X) \simeq 4 \sum_k \frac{c_k}{\lambda_k} \chi(X \cap L_k) \quad (5)$$

where L_k is the set of $3D$ discrete lines parallel to direction k, c_k the discretization weight associated to $3D$ direction k, and λ_k the density of discrete lines in direction k.

Line densities are computed as for the planar case, by dividing distance of neighbor voxels in direction k by the volume associated to a single voxel. When surface area is estimated by taking into account the three main direction in images, associated weights c_k equal $1/3$. If more directions are considered, the strategy of Ohser and Mücklich [7] was applied: each direction was projected on the unit sphere, and was used as a germ for computing VoronoÃ¯ diagram on the unit sphere. The relative surface area of each spherical domain was used as weight for the corresponding direction (Fig. 1).

2.3 Intercepts count in digital images

The number of connected components in the intersections of the structure with a discrete line is obtained by counting the number of intercepts with its boundary. This intercepts count can be efficiently computed by using the run-length encoding of the binary objects.

The run-length encoding groups all the pixels on the same line in a single element. If the binary object is compact enough, as it is often the case for connected components, the run-length encoding heavily reduces the number of elements required to represent the object, compared to the simple pixel representation. Thanks to the run-length encoding, it is possible to compute the intercepts the binary object line by line, instead of computing it pixel by pixel.

The data structure used as the input of the algorigthm is a set of all the indexes of the object with their first element remove – in 3D, this is all the possible (y, z) pairs – to which are associated a list of lines. The lines are coded as a x position and a length.

The algorithm used to count the intercepts is described in Figure 2.

2.4 Roundness

Roundness is a commom parameter used for comparing shapes independently of their size. It equals the square root of the "shape factor", or "isoperimetric deficit". It should be noted that although widely used, these parameters present many drawbacks: high sensitivity to perimeter measure, value possibly out of the expected bounds, similar values for different shapes[8]...

We defined here the roundness as the ratio of equivalent d-perimeter over the measured d-perimeter. The

Algorithm 2.1: COUNTINTERCEPT(O)

```
comment: intercept count
ic ← ∅
comment: the offset of the neighbors on the first axis
xno ← offset(0)
xno[0] ← 1
for each ls ∈ O
do { comment: 2 intercepts per line on the first axis
     ic[xno] ← ic[xno] + 2 · length(ls)
     for each nls ∈ neighbors(ls)
     do { comment: straight and diagonal offsets of the neighbors
          no ← offset(nls)
          dno ← offset(nls)
          dno[0] ← 1
          if isEmpty(nls)
          then { comment: all the lines are on a contour
                 for each l ∈ ls
                 do { ic[no] ← ic[no] + length(l)
                      ic[dno] ← ic[dno] + 2 · length(l) }
          else { comment: iterate over the lines in ls and the empty lines in nls
                 il ← iterator(ls)
                 inl ← iterator(nls)
                 nMin ← −∞
                 nMax ← pos(inl) − 1
                 while isNotAtEnd(il)
                 do { lMin ← pos(il)
                      lMax ← pos(il) + length(il) − 1
                      comment: measure the intersection of these two lines
                      ic[no] ← ic[no] + max(0, min(lMax, nMax) − max(lMin, nMin) + 1)
                      ic[dno] ← ic[dno] + max(0, min(lMax, nMax + 1) − max(lMin, nMin + 1) + 1)
                      ic[dno] ← ic[dno] + max(0, min(lMax, nMax − 1) − max(lMin, nMin − 1) + 1)
                      comment: and move to the next line, either in ls or in nls
                      if nMax ≤ lMax
                      then { nMin ← pos(inl) + length(inl)
                             next(inl)
                             if isNotAtEnd(inl)
                               do nMax ← pos(inl) − 1
                               else nMax ← ∞ }
                      else next(il) } } }
```

Figure 2: The algorithm used to count the intercepts

equivalent d-perimeter is defined as the perimeter of the d-ball with same volume V.

$$roundness = \frac{P_{eq}}{P} \tag{6}$$

$$P_{eq} = \frac{d \cdot V}{R_{eq}} \tag{7}$$

$$R_{eq} = \sqrt[d]{\frac{V \cdot \Gamma(\frac{d+1}{2})}{\pi^{\frac{d}{2}}}} \tag{8}$$

$$\Gamma(\frac{d+1}{2}) = \begin{cases} \frac{d}{2}! & \text{if } d \text{ is even} \\ \frac{\sqrt{\pi} \cdot d!!}{2^{\frac{d+1}{2}}} & \text{if } d \text{ is odd} \end{cases} \tag{9}$$

$$d!! = \begin{cases} 1 & \text{if } d \leq 1 \\ d \cdot (d-2)!! & \text{otherwise} \end{cases} \tag{10}$$

Roundness takes values between 0 and 1. It is equal to 1 for circular or spherical shapes, and decreases for more complicated shapes. For 3D shapes, equivalent decriptive parameters can be defined using ratio of surface area and volume [1].

3 Implementation

The implementation of this algorithm has been done in the `itk::ShapeLabelMapFilter` class, which is already in charge of the computation of several shape descriptors. The run-length encoding used in the `itk::LabelObject` representation is reused. The implementation is N-dimensional and thus is usable for any image dimension. The most useful cases, 2D and 3D have been specialized to provide a more accurate estimation by also using the diagonals – 4 directions in 2D and 13 directions in 3D. In the 4D case and greater, the diagonals are not used.

The perimeter estimation is now turned on by default, but can still be disabled with `SetComputePerimeter(false)` in `itk::ShapeLabelMapFilter` if not needed. This should enhance the user experience, especially for the attributes non obviously derived from the perimeter like the roundness.

3.1 N-Dimensional name

The attribute is called *Perimeter* independently of the dimension of the image and is available in the `itk::ShapeLabelObject`.

3.2 Border management

The borders of the image is considered to be in the background, so the contour of an object touching the border of the image is measured. This was not the case in the previous (undocumented) implementation and has been proven to be misleading for many users. It is possible to subtract the perimeter on the border to the full perimeter if the measure without the part touching the border is needed.

3.3 Multithreading

The architecture implemented in `itk::ShapeLabelMapFilter` use one thread per LabelObject. The perimeter estimation has been integrated in this architecture, providing a multithreaded implementation as long as there are several LabelObjects in the input LabelMap.

4 Evaluation method

Perimeter and surface area are measured on discretization of various 2D and 3D shapes whose true perimeter or surface area is known. Planar test shapes include disks, rings obtained by the difference of several disks, trefoil shape, ellipses and rectangles with various sizes aspect ratios. Test shapes for 3D measurements include balls with various radii, hollow balls, prolate and oblate ellipsoids, and cuboids. Only significant results are presented here.

Shapes were discretized following the Gauss discretization scheme [3, 5]. Binary images can be considered as a subset of a rectangular grid L_d where d = 2,3. Such a grid can be written as

$$L_d = \Delta_1\mathbb{Z} \times \ldots \times \Delta_1\mathbb{Z}, \tag{11}$$

where the d-uple $(\Delta_1, \ldots, \Delta_d)$ defines the pixel or voxel size. A grid point x belongs to the reconstructed structure if the grid cell centered on x hits X (See Fig. 3).

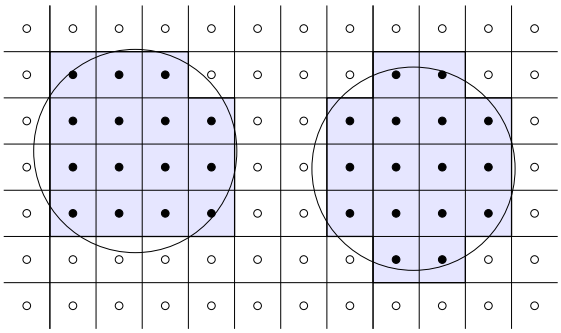

Figure 3: Discretization of two disks with same radius and different positions. A grid cell belong tho the reconstructed shape if its center is inside the original shape. The reconstructed shapes have different size and shape.

For each test shape, measurements were made on several discretisation of the shape, with various orientations and various origins. The center of each shape was chosen at random within the center pixel or voxel of the grid.

Perimeter of 2D discretized shapes was measured using Crofton method using 2 and 4 directions, and by computing the length of the reconstructed contour (Matlab Software was used). Surface area of 3D discretized shapes was measured using Crofton method using 3 and 13 directions, and by computing the surface area of the reconstruced shape obtained by the marching cubes algorithm. The average and the standard deviation of measurements were computed for all shapes discretized with the same resolution. Effect of orientation was investigated by computing average and standard deviation of shapes having the same orientation.

5 Accuracy

5.1 Perimeter of disks at various resolutions

Figure 4 shows average and standard deviation of perimeter measured on disks with various resolutions. Perimeter measured using Crofton method converges towards the real value when the resolution increase. The convergence speed is faster when four directions are used instead of two, and the variability is lower.

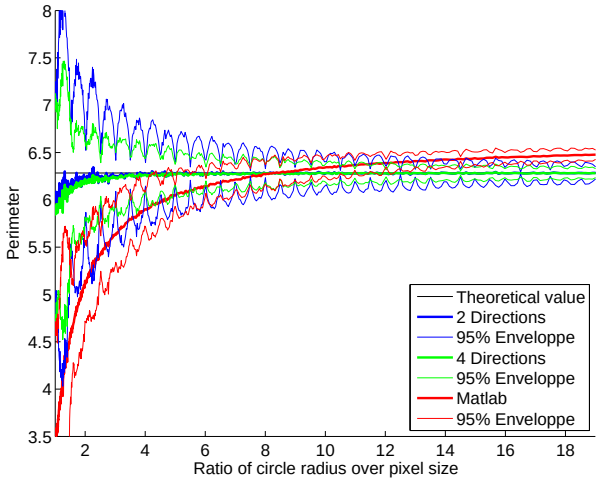

Figure 4: Perimeter measure of disks with various resolutions and at random positions

Perimeters measured with Matlab show similar variability, but the measured values do not converge toward the true value. This shows the inaccuracy of the method consisting in counting boundary pixels.

The oscillations of the variability is due to the fact that for disks with diameters equal to an integer, the number of intersections with horizontal or vertical lines is always the same, thus greatly reducing the variability of measures.

5.2 Perimeter of ellipses with various orientations

Figure 5 shows measures obtained on discretized ellipses (semi axis lengths equal to 30 and 10) with various orientations. Using Crofton method with only 2 directions produces greater errors on the measure than with 4 directions. However, in both cases, the average over the orientations of the mean values is very close to the actual perimeter.

The perimeter by counting boundary pixels, for example by using Matlab's "regionprops" methods, leads to a systematic bias: whatever the orientation, measure is different from the actual perimeter. Note that the difference does not decrease with the resolution.

5.3 Roundness of random ellipses

Figure 6 shows the roundness measured on a population of 100 discretized ellipses with semi axis lengths equal to 4 and 2 pixels. The center of the ellipses is chosen at random within the center pixel, and orientation is chosen at random between 0 and 180 degrees.

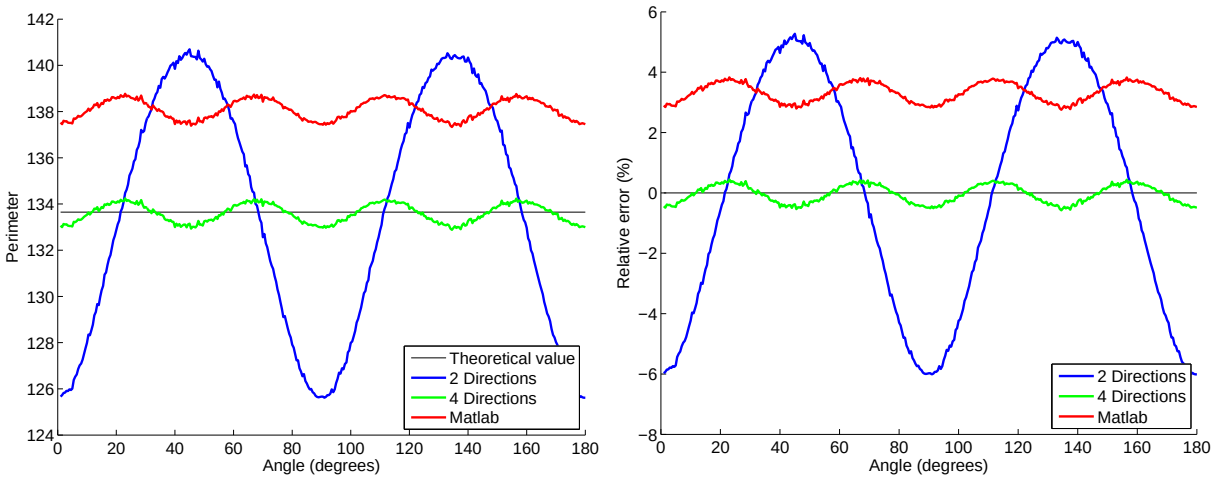

Figure 5: Variations of perimeter measure of a digitized ellipse depending on the orientation. Left: perimeter measures. Right: relative errors (in percent).

The variations in the perimeter measure cause variation in the roundness measure. The variability of roundness is less when measured with the Crofton method using 4 directions. Not that in some cases, the computed roundness may be greater than 1, due to the error in perimeter measure.

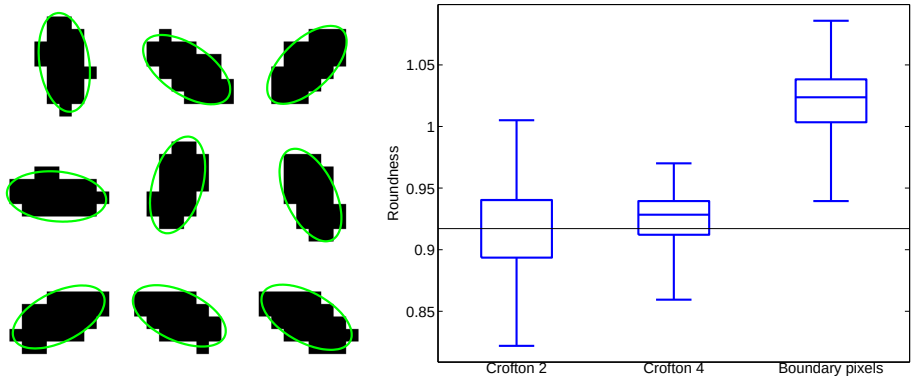

Figure 6: Measure of roundness on random ellipses. Left: some examples of test ellipses and their discrete reconstructions. Right: box plots of roundness computed from perimeter measured from different methods.

5.4 Surface area of 3D shapes

Comparison of surface area measured on several discrete reconstructed shapes is given in Table 1. Test shapes comprise a ball with radius 30 voxels, prolate ellipsoids (semi axis lengths equal to 30, 10 and 10 voxels), oblate ellipsoids (semi axis lengths equal to 10, 10 and 30 voxels), and torus (outer and inner radius equal to 50 and 10 voxels).

As for the planar case, the measure is closer to the actual value when using more directions. The error is around 2 percent in average for 3 directions, and less than 1 percent for 13 directions.

Shape	(φ,θ)	S	$S_{crofton^3}$	$S_{crofton^{13}}$	$S_{VTKmarch}$	S_{ITKiso}
ball	*n.a.*	11309.8	11312.0 (+0.0%)	11306.6 (−0.0%)	12298.2 (+8.7%)	12299.5 (+8.8%)
prolate	$(0,0)$	3082.9	2938.7 (−4.7%)	3084.6 (+0.1%)	3354.1 (+8.8%)	3354.9 (+8.8%)
-	$(45,0)$	–	3137.3 (+1.8%)	3086.3 (+0.1%)	3353.9 (+8.8%)	3354.7 (+8.8%)
-	$(45,45)$	–	3150.6 (+2.2%)	3083.3 (+0.0%)	3351.5 (+8.7%)	3352.2 (+8.7%)
oblate	$(0,0)$	6856.8	6290.7 (−8.3%)	6807.6 (−0.7%)	7410.2 (+8.1%)	7411.3 (+8.1%)
-	$(45,0)$	–	6872.0 (+0.2%)	6789.0 (−1.0%)	7369.7 (+7.5%)	7370.7 (+7.5%)
-	$(45,45)$	–	7154.7 (+4.3%)	6808.5 (−0.7%)	7385.9 (+7.7%)	7386.7 (+7.7%)
torus	$(0,0)$	11843.5	11417.3 (−3.6%)	11792.9 (−0.4%)	12826.6 (+8.3%)	12828.7 (+8.3%)
-	$(45,0)$	–	11809.3 (−0.3%)	11766.8 (−0.6%)	12787.4 (+8.0%)	12789.3 (+8.0%)
-	$(45,45)$	–	12086.0 (+1.9%)	11822.4 (−0.2%)	12849.1 (+8.5%)	12851.1 (+8.5%)

Table 1: Differences between actual surface area and its measures with different methods, on shapes with various orientations. The orientation is given by the direction of the shape rotation axis, defined by the azimut φ (between 0 and 360 degrees) and the colatitude θ (between 0 and 180 degrees).

When using Crofton method, the measured perimeter is oscillating around the actual value. If 3 directions are used, relative error is usually less than 5 percents. Errors are smaller when 13 directions are used (maximum relative error equal to 1 percent).

When using isosurface reconstruction, the surface area is systematically over estimated. The difference is around 8 percent in average.

6 Timing

The performance of the Crofton based method as well as several other methods available in ITK or VTK are shown in Table 2. The timings were obtained using the `./perf3D ../images/ball450.nrrd` command on an Intel(R) Xeon(R) CPU X5570 processor at 2.93GHz with 8192Kb cache, 24Gb of RAM running Ubuntu linux 11.04 64 bits with gcc 4.5.2. The input image have a size of $1000 \times 1000 \times 1000$ and contains a single binary object: a ball of radius 450.

Method	Timing
Proposed method	1.80882 s
VTK marching cubes	15.3972
VTK marching cubes + gaussian filtering	26.6482
ITK BinaryMask3DMeshSource	29.1124

Table 2: Excecution times in seconds for measuring the surface area of a ball with radius 450 voxels in a $1000 \times 1000 \times 1000$ voxels image.

References

[1] Philippe Andrey, Kiên Kiêu, Clémence Kress, Gaëtan Lehmann, Leïla Tirichine, Zichuan Liu, Eric Biot, Pierre-Gaël Adenot, Cathy Hue-Beauvais, Nicole Houba-Hérin, Véronique Duranthon, Eve Devinoy, Nathalie Beaujean, Valérie Gaudin, Yves Maurin, and Pascale Debey. Statistical analysis of 3d images

detects regular spatial distributions of centromeres and chromocenters in animal and plant nuclei. *PLoS Computational Biology*, 6(7):1–15, July 2010. 2.4

[2] L. Ibanez and W. Schroeder. *The ITK Software Guide*. Kitware, Inc. ISBN 1-930934-10-6, 2003. available at: http://www.itk.org/ItkSoftwareGuide.pdf.

[3] Reinhard Klette and Ariel Rosenfeld. *Digital Geometry*. Morgan Kaufman, 2004. 1, 4

[4] C. Lang, J. Ohser, and R. Hilfer. On the analysis of spatial binary images. *Journal of Microscopy*, 203(3):303–313, Sept. 2001. 2.2

[5] David Legland, Kiên Kiêu, and Marie-Françoise Devaux. Computation of minkowski measures on 2D and 3D binary images. *Image Analysis Stereology*, 26(6):83–92, June 2007. 1, 2.2, 4

[6] W. E. Lorensen and H. E. Cline. Marching cubes: a high-resolution 3D surface construction algorithm. In *Proceedings of the 14th ACM SIGGRAPH on Computer Graphics*, volume 21, pages 163–169, 1987. 1

[7] Joachim Ohser and Franck Mücklich. *Statistical Analysis of Microstructures in Materials Sciences*. J. Wiley & Sons, Chichester, 2000. 2.2

[8] E. Pirard and G. Dislaire. Robustness of planar shape descriptors of particles. In *Proc. IAMG'05, Toronto*, 2005. 2.4