Image Anal Stereol 2001;20(Suppl 1):306-312

COMPUTATION OF THE SURFACE AREA OF NATURAL SNOW 3D IMAGES FROM X-RAY TOMOGRAPHY: TWO APPROACHES

JEAN-BRUNO BRZOSKA[1], FREDERIC FLIN[1], BERNARD LESAFFRE[1], CECILE COLEOU[1], PASCAL LAMBOLEY[2], JEAN-FRANÇOIS DELESSE[3], BERTRAND LE SAËC[3] AND GERARD VIGNOLES[4]

[1]Météo-France,CNRM/CEN, 1441 rue de la Piscine, 38406 St-Martin d'Hères, France, [2]Météo-France,CNRM/SCEM,42 Avenue G. Coriolis, 31057 Toulouse Cedex 1, France, [3]LaBRI, UMR 5800 CNRS-Université Bordeaux 1, 351 Cours de la Libération, 33405 Talence Cedex, France, [4]LCTS, UMR 5801 CNRS-SNECMA-CEA-Université Bordeaux 1, 3 Allée La Boëtie, 33600 Pessac, France
e-mail: firstname.se-condname @meteo.fr, Pascal.Lamboley@meteo.fr, {delesse,lesaec}@labri.u-bordeaux.fr, vinhola@lcts.u-bordeaux.fr

ABSTRACT

Once fallen on the ground, snow undergoes a structure metamorphism governed by local temperature and humidity fields. Local 3D curvature is a governing parameter of metamorphism, whereas surface area, directly related to surface energy, can provide a valuable bridge between 3D fine-scale and 1D field-scale snow cover models. X-ray tomography was applied to natural snow at ESRF in order to parameterise the fine-scale behaviour of snow in avalanche risk prediction models. Starting from raw data consisting of a B/W 3D data file, two main approaches are considered. The former uses a triangulation procedure derived from the "marching cube" algorithm. Before triangulation, the B/W data are converted into grey levels and preprocessed in order to smooth the final mesh. The other uses a distance map of the object, from which the field of normal vectors is computed. Assuming the tangent plane approximation, the effective projected area of each surface voxel is derived from simple geometric laws. A cross-validation of both methods is provided on a natural snow image.

Keywords: discrete geometry, distance map, internal surface area, marching cube, normal vectors, snow tomography.

INTRODUCTION

Snow metamorphism is mainly governed by surface parameters, such as the surface area or the local curvature (Colbeck, 1998). The recent availability of high resolution 3D images of snow microstructures (Schneebeli and Krüsi, 2001), especially using X-ray microtomography (Coléou *et al.*, 2001) now allows to address the 3D computation of transfer or mechanical phenomena in realistic situations. In the present work, we will deal with the computation of the surface area, as accessible through X-ray CMT, that is, excluding any feature dimension lower than the acquired image resolution.

The standard evaluation of surface area is to obtain first a mesh of the void/matter interface from the raw data, and then to sum up all the facets' areas. A convenient way to produce such a mesh is to use the "marching cube" algorithm (Lorensen and Cline, 1987), which provides an isosurface separating void and matter in a greyscale density matrix image for a given threshold. Actually, the greyscale image is obtained by applying a classical smoothing filter to the starting B/W image (3^3-sized Gaussian kernel). We will present here an optimised area calculator based on this algorithm.

Another approach is to start from the normal vector field of the object surface. It can be obtained by several ways (Papier and Françon, 1998; Braquelaire and Vialard, 1999; Lenoir *et al.*, 1996). The method presented in the following is original and is detailed elsewhere (Flin *et al.*, these proceedings). Providing the normal vector $\boldsymbol{n}$ for each surface voxel of a discrete object allows the assessment of

surface area as the sum of the elementary areas each surface voxel projects on its local tangent plane. The discrete approach is validated on spheres and cubes.

A cross-validation between the two approaches is given for the tomograph of a natural snow sample.

MARCHING CUBE APPROACH

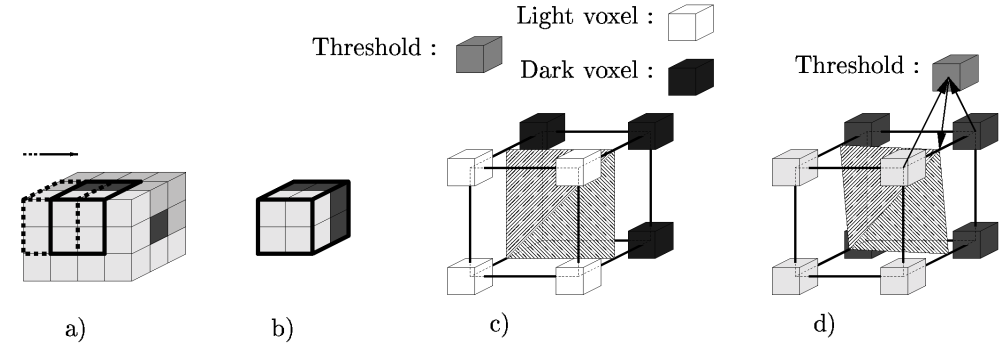


Fig. 1. *The 4 steps of the marching-cube algorithm.*

The starting point (Fig. 1a) is a $n_1 \times n_2 \times n_3$ greyscale image. The image can be split into $(n_1\text{-}1) \times (n_2\text{-}1) \times (n_3\text{-}1)$ sub-images of size $2 \times 2 \times 2$, with a suitable scanning (Fig. 1b). Each of them is treated independently by the algorithm, and leads to a list of at most 5 facets whose vertices lie on the edges of a cube. A threshold (a given greyscale level) discriminates voxels into two sets. By convention, we call "dark" voxels those that have their values below the threshold, and "light voxels" the other ones (Fig. 1c). The appropriate facets are drawn from a table with the help of a «configuration index» which is computed by enumerating the «dark voxels» in the extracted images. Since any of its 8 voxels may be either dark or light, 2^8 different indexes may be encountered. In order to obtain a smooth surface when all the facets are combined, the positions of the vertices are calculated by a bilinear interpolation between the threshold and the values of the concerned voxels (Fig. 1d).

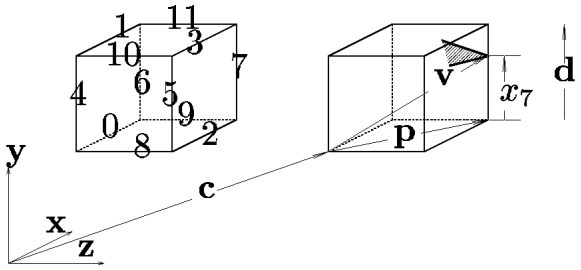


Fig. 2. *The conventions used by the formulae in the text.*

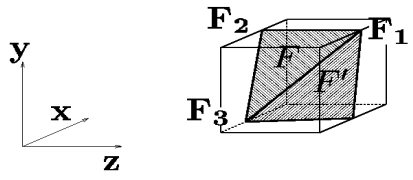


Fig. 3. *A sample configuration with 2 facets.*

The area of a facet F can easily be calculated with a cross product involving the coordinates of its vertices $\boldsymbol{F}_1,\boldsymbol{F}_2,\boldsymbol{F}_3$:

$$A(F) = \tfrac{1}{2} \left| (\boldsymbol{F}_2 - \boldsymbol{F}_1) \wedge (\boldsymbol{F}_3 - \boldsymbol{F}_1) \right| \tag{1}$$

In order to establish a minimal surface formula for each configuration, we define first an orthogonal coordinate system (see Fig. 2) $(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ that verifies $|\boldsymbol{x}| = |\boldsymbol{y}| = |\boldsymbol{z}| = 1$. In this system let $\boldsymbol{c}$ be the lowest corner of a unit cube (the eight corners will have positions $\boldsymbol{c} + \binom{0}{1}\boldsymbol{x} + \binom{0}{1}\boldsymbol{y} + \binom{0}{1}\boldsymbol{z}$, where $\binom{0}{1}$ means either 0 or 1). Assuming that $\boldsymbol{d} \in \{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}\}$, $\boldsymbol{p} \in \{0,1\}^3$, and $\boldsymbol{p}.\boldsymbol{d} = 0$, the notation $(\boldsymbol{p},\boldsymbol{d})$ refers to an edge of such a cube: it means that both $\boldsymbol{c} + \boldsymbol{p}$ and $\boldsymbol{c} + \boldsymbol{p} + \boldsymbol{d}$ are corners. The twelve edges are numbered with $r(\boldsymbol{p},\boldsymbol{d})$ as shown at Fig. 2. A vertex $\boldsymbol{v}$ that lies on the $(\boldsymbol{p},\boldsymbol{d})$ edge of the cube located at the origin verifies:

$$\boldsymbol{v} = \boldsymbol{p} + x_i\boldsymbol{d}, \text{with } i = r(\boldsymbol{p},\boldsymbol{d}) \in [0,11] \cap \mathbb{N} \text{ and } x_i \in [0,1] \tag{2}$$

Since for each marching-cube configuration, no more that one vertex lies on each edge of the unit cube, it is possible to write a formula whose variables are $x_0 \ldots x_{11}$ suitable for the area calculation. This is made very easily by setting up a program that writes out literally Eq. 1 for each configuration, using the above convention. These 256 formulae can be written in separate files, the only parameters that should appear in them being $x_0 \ldots x_{11}$. A basic symbolic computation software* is then able to simplify them easily, using some simple rewriting rules. After simplification, the formulae are recovered, translated in C, and incorporated in a single C source file as an array of functions.

Let us see how the procedure works on an example configuration with 2 facets F and F' (Fig. 3). The vertices of F are $\boldsymbol{F}_1$, $\boldsymbol{F}_2$ and $\boldsymbol{F}_3$. Let a_i, b_i and c_i be the coordinates of the i^{th} vertex of F. The application of Eq. 1 on F provides:

$$A(F) = \tfrac{1}{2} \left[\left((b_2 - b_1)*(c_3 - c_1) + (c_1 - c_2)*(b_3 - b_1) \right)^2 + \right.$$

$$\left. \left((c_2 - c_1)*(a_3 - a_1) + (a_1 - a_2)*(c_3 - c_1) \right)^2 + \left((a_2 - a_1)*(b_3 - b_1) + (b_1 - b_2)*(a_3 - a_1) \right)^2 \right]^{\frac{1}{2}} \quad (3)$$

Using the above convention, and after simplification, we obtain:

$$A(F) = \tfrac{1}{2} \sqrt{\left((x_1 - x_0)^2 + (x_3 - x_1)^2 + 1 \right)} \quad (4)$$

It can be seen that some terms have vanished, just because some vertices lie on edges belonging to the same face of the cube. In this case, only 2 subtractions, 2 additions, 2 square powers, and one square root are needed. These numbers were respectively equal to 12, 3, 3 and 1 with the basic formula and there were 6 multiplications: the above procedure is thus a valuable optimisation in terms of computer time saving.

DISCRETE APPROACH

Presentation

In this approach, the first step is to compute the surface normals from a distance map of the volume object (chamfer discrete distance (d) using a 5^3 chamfer kernel (Borgefors, 1984)). Instead of using the raw data of the discrete distance map, which contains many digitization artefacts, the gradient vector field (**grad** d) was built from the map using the first neighbours (3^3 kernel). Inside the working neighbourhood (in our case, a sphere 5 voxel units in radius), P being the current surface voxel, the gradient vector average of every voxel Q for which

$$\frac{\mathbf{grad}\, d(P) \cdot \mathbf{grad}\, d(Q)}{\left\| \mathbf{grad}\, d(P) \right\| \left\| \mathbf{grad}\, d(Q) \right\|} \leq \cos\alpha \quad (5)$$

was taken. The threshold angle α was set to 30° so that the largest facet angle that could be accounted for be 120°, *i. e.* the angle of hexagonal ice crystals. Once the normals are known, it becomes possible to assess the surface area, using a tangent plane remapping.

For a well-triangulated surface, the sum of areas of each triangle provides a good assessment of the surface area: this is indeed the method generally used for computing the area of discrete objects (Frey and George, 2000); it is highly dependent on the quality of the meshing procedure. This method, as well as most existing others, intrinsically assumes the tangent plane approximation: each triangle is considered as a flat surface. Assuming again the tangent plane approximation, a similar approach can be applied to a raw set of voxels by taking for each voxel the area of its *visible* projection along $\boldsymbol{n}$ - that is - on its tangent plane. The main difficulty is to decide which part of the projection is visible from the normal vector, which can take any orientation with respect to the object grid. Doing the tangent plane approximation means that we consider here the discrete plane that is tangent to the current surface voxel P (see Fig. 4a): this plane appears as a sort of stairway with uneven steps. The complexity of these successions of flats and steps was recently addressed by (Vittone and

*calc, an emacs package written by Dave Gillespie, available on http://www.gnu.org/, has been used.

Chassery, 1997). By using a paving change, it is possible to bypass these difficulties in the practical case of discrete area assessment (manuscript submitted to *IEEE Trans. Im. Proc*).

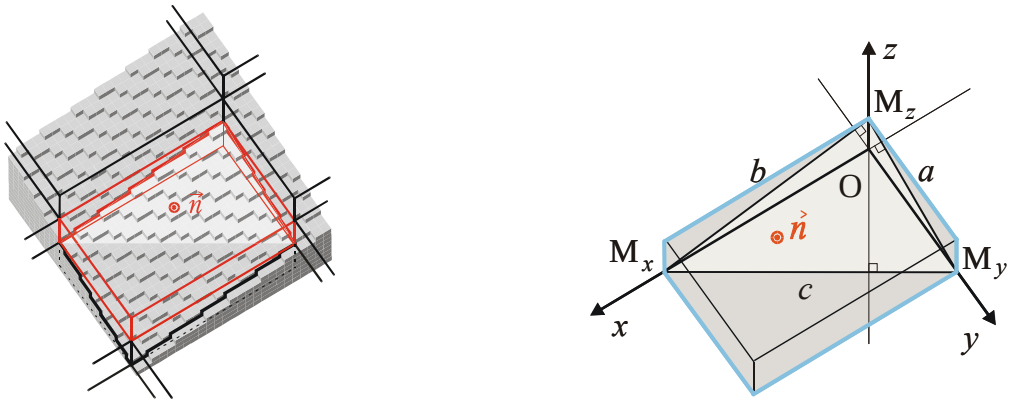


Fig. 4. *a) Discrete plane paving: original and regular, b) Derivation of regular paving parameters.*

Depending on its location, each surface voxel of a discrete plane can present one, two (step edge) or three (step vertex) facets when observed from the normal direction. On a *regularly* paved discrete plane, *i.e.* when there is no flat between steps in any direction, each voxel fully exhibits its three visible facets. The discrete tangent plane of each voxel can then be regularly paved without changing the grid orientation, using rectangular elements («bricks») of suited proportions (Fig. 4a) instead of the original cubic voxels of the 3D image.

The only parameters required to achieve a regular paving on a discrete plane are the proportions of the paving brick, not its size. On the other hand, once a paving element is chosen, its size should be related to the original cubic voxel size (set by convention to unity in the following). This can be achieved by noticing that for each voxel of a discrete plane observed from its normal vector ***n***, the facet that contains ***n*** is fully visible. In the local frame *Oxyz* of Fig. 4b, $M_x(x_1,0,0)$, $M_y(0,y_1,0)$,and $M_z(0,0,z_1)$, respectively denote the intersection of an affine plane arbitrarily set by choosing x_1 parallel to the tangent plane. This facet has the largest area $S_{facet} = \max(|x_1y_1|, |x_1z_1|, |y_1z_1|)$.

Besides, both pavings have the same orientation. The right way to consider these pavings is to compare elements (facets) that remain visible on both pavings whatever the orientation of the plane, that is, to consider the magnification ratio $r_p = S_{facet}/S_{voxel} \equiv S_{facet}$. Given that magnification ratio, we have then to compute the coefficient (called m in the following) between the elementary voxel facet area (unity) and the area $S_{outline}$ of the projected outline of that voxel along ***n*** (the region inside the thick grey contour). This area can be obtained from simple geometrical constructions shown in Fig. 4b. By construction, the parallelepiped defined by O, M_x, M_y, M_z generates by itself a regular paving of the tangent plane, and the triangle (M_x,M_y,M_z) is seen perpendicularly. Then, a, b, c being respectively the side lengths, p the perimeter and S_t the area of the triangle, one has:

$$a = \sqrt{y_1^2 + z_1^2} \quad b = \sqrt{x_1^2 + z_1^2} \quad c = \sqrt{x_1^2 + y_1^2}$$

$$p = \frac{1}{2}(a+b+c) \quad S_t = \sqrt{p(p-a)(p-b)(p-c)} \tag{6}$$

By symmetry with respect to triangle sides, $S_{outline} = 2S_t$. The desired coefficient m is then $m = S_{outline}/S_{facet}$. This formula was directly implemented for the computation of the effective area of each projected voxel, the sum over the object surface providing the surface area assessment.

Tests on spheres

The spherical shape exhibits all possible orientations and allows to test the algorithm without grid orientation effects. The main sources of error remain: *i)*the original digitization of the shape itself, especially for small spheres, and *ii)* the finite number of orientations and positions for a given search sphere radius *r*.

The definition of the effective radius r_{eff} of a discrete sphere *S(r)* of theoretical radius *r* is not straightforward. A good approximation (Papier and Françon, 1998) in terms of volume (number of voxel centres inside the theoretical Euclidean sphere) is $r_{eff}=\sqrt{r(r+1)}$. However, the value to approximate is the area of the original Euclidean sphere, *i. e.* $4\pi r^2$. Computations are done on *digitised* spheres centred on a given voxel *C* obtained using the above approximation; *Q* being any voxel of the working space: $Q \in S(r) \Leftrightarrow d_e^2(Q,C) < r(r+1)$. For *r* greater than 12 voxels, both errors are of the order of ± 1%, within unavoidable digitization uncertainties of image acquisition and thresholding of real snow samples. The angular error between $\boldsymbol{n}$ and $\overrightarrow{PC}$ is less than ± 1% for $r > 12$.

Tests on tilted cubes

Unlike spheres, the cubic shape selects one given orientation with respect to the grid. The tilted cube test is demanding for discrete geometry models because it stresses on orientation problems. Fig 5a shows the result for a tilted cube of edge length 70 voxels. The rendering was directly obtained from the calculated normal vector field ($\boldsymbol{v}$ being an arbitrary illumination direction, grey level = $\boldsymbol{n}.\boldsymbol{v}$). Fig. 5b displays the coefficient *m* for any visible voxel, showing good uniformity and edge detection all over the shape. The computed value of *A* is indeed 28587 voxel^2, *i. e.* such that $6 \times 70^2 \geq A \geq 6 \times 69^2$. Our algorithm was then tested on a series of 25 discrete cubes of same size (theoretical edge length 40 voxel units), and evenly spaced orientations. We used a standard code∗ which recursively subdivides each triangular facet of a regular polyhedron (tetrahedron, octahedron or icosahedron), selecting only points that belong to a same octant because of the symmetries of the cube. Redundant cubes (aligned with the grid) were removed.

The visual rendering of both normal vectors and voxel magnification ratio *m* were found to be comparable, and for each test cube the condition $6\times a^2 \geq A \geq 6\times(a\text{-}1)^2$ was fulfilled (*a* being the cube edge length in voxel units). However, a direct assessment of the accuracy of surface area computation remains difficult. An intuitive definition of the area of a *discrete* cube would be the sum of areas of each facet outline, individually seen along its normal vector. Unfortunately, these projected areas strongly depend on the cube orientation with respect to grid orientation (see Fig. 5a). We did not hear of any formalism taking this relative orientation into account. It was then planned to do the same calculations for larger cubes ($a \sim 300$ voxels) to approach the continuum limit.

CROSS-VALIDATION ON A SNOW SAMPLE

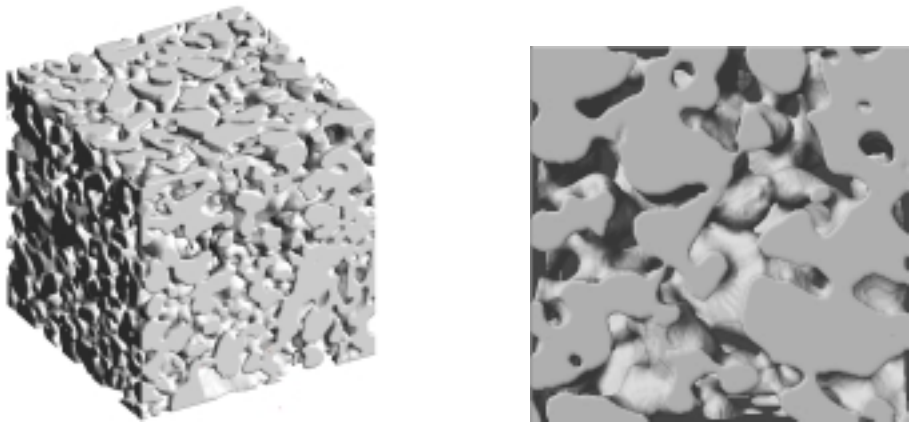
The algorithms were first tested on geometrical shapes (spheres, tilted cubes), then applied to a natural snow image obtained by X-ray computerized microtomography at ESRF. Snow was sampled on a slab that did not undergo melt-freeze cycles (sintered structure, tough and fine) at l'Alpe d'Huez, French Alps (Fig. 6).

Preliminary image treatments performed on the reconstruction led to a 441^3 voxel binary image. Fig. 6b is a VRML visualization of an extract from this image, as obtained by the "smoothing + marching cube" program. It can be seen from this figure that some roughness remains from the original binarization procedure, which explains the positive difference between methods 2 and 3 at Table 1.

∗points.tar.gz, by Jon Leech. Available at: ftp//cs.unc.edu/pub/users/Jon

a) b)

Fig. 5. *a) Normal vector field rendering Normal* **n**, *illumination* **v**, *grey_level =* **v. n,** *b) Tilted cube of edge 70: Magnification ratio m (purple m = 1, to cyan m = 1.7).*



a) b)

Fig. 6. *a) Snow sample, cube size 4.5 mm, b) Inner detail after tessellation*.

Table 1. *Comparison of different methods for surface area assessment from a 441*^{3} *voxel snow sample.*

Method	**Porosity**	*Surface Area* (pix^{-1})	*Diff. w/ method #3*
#1 Marching cubes directly from binary image	0.4589	0.06806	+ 10.0 %
#2 Image smoothing + Marching cubes	0.4590	0.06321	+ 2.2 %
#3 Voxel effective projection	0.4589	0.06185	

CONCLUSION

Two approaches have been developed in order to evaluate the internal surface area in 3D images of snow samples, as obtained by X-ray CMT: an optimised “marching-cube” scheme, and a newer discrete approach. Results of the first tests show that the latter approach gives slightly lower, hence more accurate, estimates; however, it costs more computer time.

REFERENCES

Borgefors G (1984). Distance transformations in arbitrary dimensions. Comp Vis Graph Image Proc 27:321-4.

Braquelaire JP, Vialard A (1999). Euclidean paths: a new representation of boundary of discrete regions. Graphical models and image processing 61:16-43.

Colbeck SC (1998). A review of sintering in seasonal snow. CRREL Report, 97-10.

Coléou C, Lesaffre B, Brzoska JB, Ludwig W, Boller E (2001). 3D snow images by X-ray microtomography. In press in Ann Glaciol.

Frey PJ, George PL (2000). Maillages, application aux éléments finis. Paris: Hermès.

Lenoir A, Malgouyres R, Revenu M (1996). Fast computation of the normal vector field of the surface of a 3D discrete object. Lecture Notes in Computer Sciences 11:101-12.

Lorensen WE, Cline HE (1987). Marching cubes: high resolution 3D surface construction algorithm. Computer

Graphics 21:163-9.

Papier L, Françon J (1998). Evaluation de la normale au bord d'un objet discret 3D. Revue de CFAO et d'informatique graphique 13:205-26.

Schneebeli M, Krüsi G (2001). Three-dimensional reconstruction of snow. In press in Ann Glaciol.

Vittone J, Chassery JM (1997). Coexistence of tricubes in digital naive planes. Lecture Notes in Computer Science 1347:99-110.