Electronic Notes in Theoretical Computer Science 46 (2001)
URL: http://www.elsevier.nl/locate/entcs/volume46.html 24 pages

The Geometry of the Intersection of Voxel Spaces

Jean-Pierre Reveillès [1,2]

LLAIC
Université d'Auvergne
Clermont-Ferrand, France

Abstract

Views rendered by the numerous current digital medical facilities and 3D technologies (Positron Emission Tomography, Magnetic Resonance Imaging, Synchrotron Radiation, Radars, Stereography, etc.) of a 3D object may often be assimilated to tilings of $\mathbb{R}^3$ space by identical cubes (or voxels). Relating two such views of a single object obtained by two distinct processes, in order to fusion their information on a new image, requires a processing of these two tilings specially for objects whose size is close to the resolution of the employed technology.
Keywords: Cubes Intersection, Continued Fractions, Digital Geometry, Medical Imaging.

1 Introduction

3D digital images being often considered as tilings of $\mathbb{R}^3$ space by identical cubes, the *fusion problem* of Medical Imaging tries to relate two tilings of a same area when these tilings are built with voxels of different orientations and resolutions. We propose a geometrical approach to this problem by computing the intersection of overlapping voxels of both families.

This question of fusion of two 3D images is still an issue in Medical Imaging, mainly when small crucial parts of these images have to be matched, as it is the case, for example, with brain areas like caudate nuclei and putamen (see [2]). Several similar areas of interest for the pathologist are contained in rather small boxes, sometimes not larger than 10 pixels in each direction and the precision of medical facilities being fixed for technological reasons, there is no way of zooming in to get better views. Only a detailed mathematical study of the situation may help the surgeon.

[1] Contrat ACI n^o23 du Ministère de la Recherche et de la Technologie
[2] Email: reveil@llaic.u-clermont1.fr

Up to now this question is generally solved using simple Image Processing techniques whose inaccuracy is well known.

Formulas and algorithm presented in this paper should become a reference tool to evaluate the quality of existing fusion methods and particularly should help designing new ones. Fusion of different 3D images depends on a more general problem which is that of *digital coordinates change* in fixed accuracy discrete spaces and such situations abound in our present digital era.

Given two regular grids of different resolution and size placed on a same nD object, the *digital coordinates change* question consists in finding formulas relating integer coordinates, and values attached to overlapping voxels, of both parameterizations. If formulas relating coarse and fine coordinates can be easily established, their accuracy is acceptable only if both image resolutions are pretty distinct; they are not sufficient when resolutions are close to each other. In this latter case actual volumes of intersection polyhedra have to be determined and mixed into interpolation formulas.

Let us describe our approach and the content of this paper.

Digital coordinates change problem mixes polyhedra intersection with lattice periodicity. This problem is trivial in dimension 1, much richer in dimension 2 and becomes more and more complicated as dimension increases. This is mainly due to the increasing complexity of the intersection problem and also because no clear way of using lattice periodicity and cubes simple structure appears to reduce the overall amount of work as soon as $n \geq 3$.

The 2D situation is instructive because the underlying intersection problem is simple and the benefit of lattice periodicity can be easily explained with the help of continued fractions. Unhappily such lattice properties do not generalize such as in higher dimensions. This 2D case nevertheless gives precious indications about which higher dimensions notions are worth introducing, like plane sections of voxels spaces studied in parts 5,6, formulas giving plane sections of cubes of part 8 and *quadratic cubes* of part 13.

Cubes of this kind have quadratic coordinates vertices, which allows Algebraic Computer Systems to exactly compute intersection points of edges and faces. This, in turn, gives processes to express formally the vertices of the intersection of two quadratic cubes; this is exposed in part 14.

But, coming to this point, the seemingly simple problem of intersecting two cubes in general position, puzzling questioning arises: should one use algorithms designed for any convex polyhedra or should one try, on the contrary, to take advantage of cubes especially simple structures and numerous symmetries? Immediate considerations on this intersection problem show that a naive approach along either of these ways results in a large amount of computations and, particularly, of inequalities checking.

It appears that the specific structure of cubes, mainly their symmetries, (used in parts 9, 10, 11) results in a very significant reduction of the number of inequalities to be checked. More precisely exact formulas containing no inequality, are obtained for plane sections of cubes, leading, combined with a

3D convex facets intersection algorithm (cf. part 15), to an efficient procedure for the intersection of two voxels. A nice consequence of the formal approach is the removal of the treatment of all particular cases from the programming these degenerate cases being contained into the analytic formulas.

2 The 2D digital coordinates change problem

This problem occurs when information (for example gray levels) of two images of different orientation and resolution of a same object, have to be matched. Both kinds of pixels can be described as small and large. We suppose that

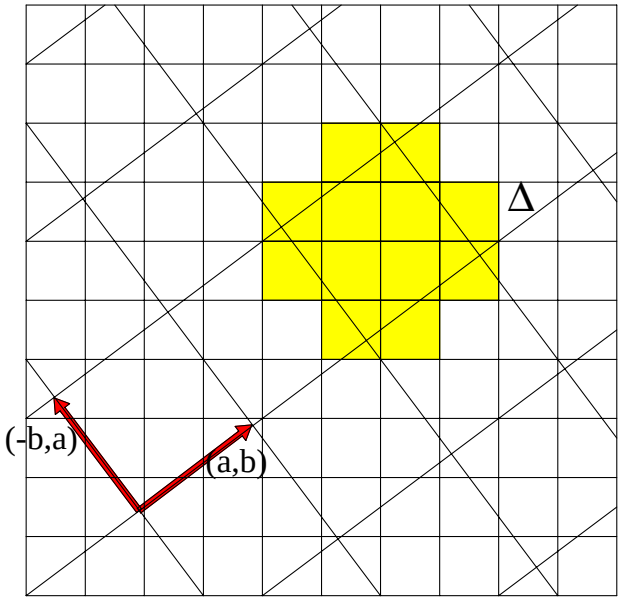

Fig. 1. Digital coordinates change.

small pixels are defined by integer vectors $(m, 0)$ and $(0, m)$ (i.e. they are $m \times m$ blocks of unit squares), and large pixels are defined by integer vectors (a, b) and $(-b, a)$. It is easy to see that large pixel whose coordinates are (x, y) roughly correspond to small pixel $(\left[\frac{ax-by}{m}\right], \left[\frac{bx+ay}{m}\right])$. Similar formulas giving coordinates of large pixels as functions of coordinates of small ones can be written.

Accuracy of these simple digital coordinate change formulas is sufficient when the euclidean norm $\|(a, b)\|/m$ is not to small (i.e. greater than 10), but is to rough when $\|(a, b)\|/m$ is small, which means that both grids have similar sizes. In this case one has to compute areas of intersections of small and large pixels and take account of them into interpolating formulas.

But computing many intersections of pixels by brute force would be demanding; it is wise trying to optimize this task and continued fractions allow to do so.

Vector (a, b) defining large pixels being integer, let us suppose, moreover, that a and b are relatively prime. Let Δ denotes euclidean line directed by

(a, b). All integer points $k(a, b)$ where $k \in \mathbb{Z}$ belong to Δ and similarly for the euclidean line directed by $(-b, a)$, (see fig. 1).

In fact square $\mathcal{L}$ whose vertices are $(0, 0), (a, b), (a-b, b+a), (-b, a)$, (this is one of the large pixels), tiles the digital plane $\mathbb{Z}^2$, and this tiling is doubly periodic. Thus intersection scheme of small and large pixels is already contained in intersections of $\mathcal{L}$ and all small pixels it overlaps. This reduces considering small pixels which cover $\mathcal{L}$ edges. Obviously these coverings are 4-connected digital lines whose parametrization can be obtained easily; example of such covering is illustrated by fig. 2.

Let us say a few words of what occurs if vector (a, b) is replaced by any real vector $\overrightarrow{v}$. First this does not restrict applications, as a real vector $\overrightarrow{v}$ may be arbitrarily approximated by integer vectors, (see [7]); and it is well known that such approximations can by obtained with continued fractions associated to $\overrightarrow{v}$, (cf. [8]). Second if slope α of $\overrightarrow{v}$ is irrational, previous bounded square tiles become infinite and the intersection scheme is given by the covering of whole line Δ, and its perpendicular line, by small pixels.

Thus intersection schemes occurring in *digital coordinate changes* are always given by 4-connected digital lines covering of euclidean segments; these are bounded if $\overrightarrow{v}$ is integer and unbounded in the opposite case.

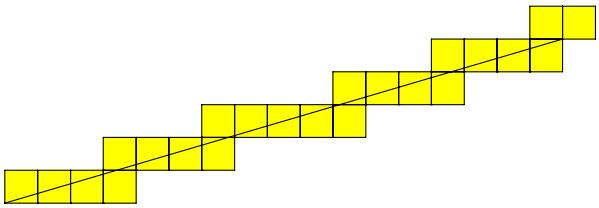

Fig. 2. Covering pixels.

If α if an irrational or a large rational number, this scheme may be difficult to describe because no obvious parametrization of covering pixels exists in the first case, and the 4-connected parametrization may involve large integers in the second one. The so called *Klein's diagram* gives the intersection scheme if α is irrational and reduces its complexity when α if rational. This diagram is simply a couple of two convex hulls, that of integer points $\{(x, y) \mid x \geq 0 \quad \text{and} \quad y \geq \alpha x\}$ and similarly that of integer points $\{(x, y) \mid x \geq 0 \quad \text{and} \quad 0 \leq y \leq \alpha x\}$. The two polygonal lines bounding line $y = \alpha x$ given by Klein's diagram are often called *Klein's funnel*, see fig. 3.

It is well known (see [4], [8]) that Klein's funnel vertices correspond to *partial quotients* of continued fraction associated to number α. The nice property of Klein's funnel lies in the following.

Proposition 2.1 *Intersection scheme of line $y = \alpha x$ with unit pixels is the same as the intersection scheme of Klein's funnel.*

This results says that in order to get intersection scheme of line $y = \alpha x$ with unit pixels, it suffices to construct that of its Klein's funnel, (one of the

two polygonal lines of the funnel suffices). As Klein's funnel is made of integer segments, its covering is a union of 4-connected digital segments. This shows how continued fractions allow to determined easily the intersection scheme of two different 2D views of a same object.

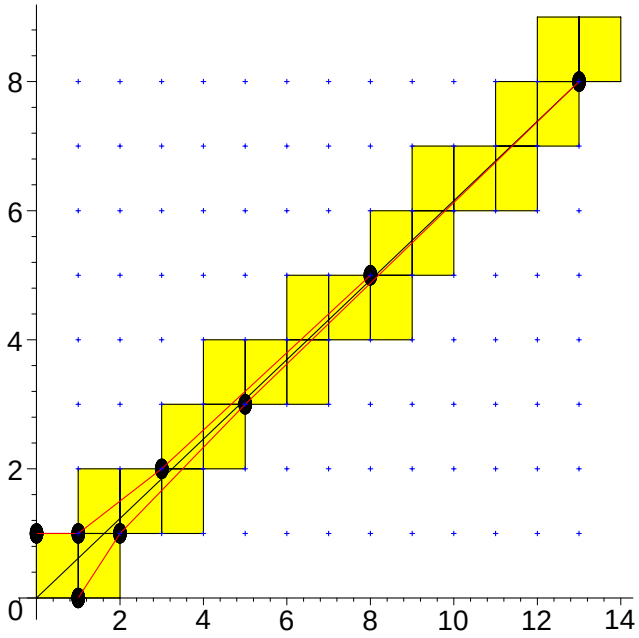

Fig. 3. Klein's funnel.

Formulas giving areas of intersections of any small and large pixels may be obtained; a close question is studied in §11 and 12.

3 The 3D digital coordinates change problem

Each voxel of a 3D image being assimilated to a unit cube, such an image is naturally a tiling of space $\mathbb{R}^3$ which may be called *voxels space* or *voxels tiling*.

This problem could be solved applying a double loop to a procedure giving the intersection polyhedron of two cubes, each of them running into one of the collections of voxels.

But this approach could miss eventual periodicities existing among intersection polyhedra of cubes analogous to that sawn for the 2D digital coordinates change problem. Although all aspects of lattice periodicities will not be studied in this paper (those which are relevant of multidimentional continued fractions will only be sketched in §16), those which are taken into account, known as *voxels space plane sections*, need to depart somehow of the obvious way of doing.

Study of the geometry of plane sections of voxels tilings and its relationships with the computation of the intersection of two cubes will now occupy the rest of this paper.

4 Intersection of voxels

If the question of the intersection of two (and only two) cubes may be considered as worthless from a theoretical point of view, this is not the case if they belong to two voxels tilings because, in this case, they possess strong regularity properties which reappear in their intersections.

As already mentioned, intersection of two cubes can either be considered as a particular case of the intersection of two convex polyhedra or as the intersection of two objects possessing really special structures. If the first point of view is well known in Computational Geometry ([5], [9], [11]), the second one has, to our knowledge, never been published, even if some technics of Computer Imagery like 3D clipping, ([6]), seem close to it.

General Computational Geometry algorithms, which use rather sophisticated data structures to describe convex polyhedra are not well adapted to the present situation for two principal reasons. The first is that they avoid degenerate cases: vertex of the first cube belonging to a face or an edge of the second, colinear edges, edges coplanar to a face, coplanar faces and so on. The second is that implementing them in exact arithmetic or in Computer Algebra Systems is still ahead ([11]) and current implementations with floating point systems destroys lattice properties, which forbids their study.

This explains main characteristics of our approach which aims to a *formal* description of the intersecting polyhedron of two cubes, that is to get the exact algebraic values of its vertices as it is the place where some lattice periodicities hide. But this could be very costly if cube specificities like their small number of faces and their rich symmetry group were not used.

Working Computer Scientists know the long way between theoretical algorithms and their coding. This paper is not only concerned with the geometry of voxels intersection algorithm but also with the issues in its coding. Efficiency of this code, mainly the deletion of almost all control tests occurring in main loop, is the first reason of our way of doing; the second one is to stay within a convenient numbers system: *quadratic algebraic* ones.

Main idea of our algorithm consists in determining faces of the intersection polyhedron of two cubes. If C and C' are two cubes in general position and $\mathcal{F}$, $\mathcal{F}'$ denote collections of their faces, any face of the polyhedron $K = C \cap C'$ is supported either by a face $f \in \mathcal{F}$ or by one $f' \in \mathcal{F}'$. If a face of K is supported by a face $f \in \mathcal{F}$, it is the intersection of two polygons, namely $supp(f) \cap C'$ and the square f, and vice versa, (more precisely $supp(f) \cap C'$ denotes the intersection of the support plane of f with cube C').

In fact support planes of voxels of one of the two families cuts an infinite number of cubes of the other family. But these support planes are periodically spaced, thus the periodicity of a voxels tiling should appear in its plane sections which are 2D ordinary tilings. Rationality hypothesis about section planes simplifying greatly these 2D induced tilings, we are able to give explicit formulas for the vertices of their tiles which, in turn, reduces considerably the

complexity of our intersection algorithm, this last one being reduced to polygons intersections computations, a rather simple task.

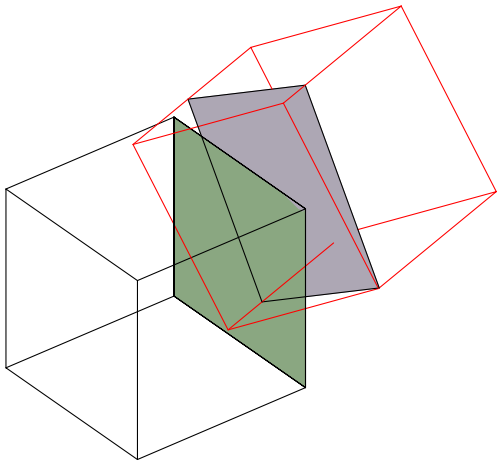

Fig. 4. Principle of the voxels intersection algorithm.

Observing that the intersection polyhedron K of two cubes C and C' has at most 12 faces, these being common parts of square faces of one of the cubes with plane sections of the other one induced by their support planes, intersection algorithm is:

Loop on all 12 faces of both cubes
    Determine plane section of other cube by the support plane of this face.
    Compute intersection of both polygons: plane section and square face.

Figure 4 shows one face of the left cube and the intersection of its support plane with the right side one. Common part of the square face and the intersection polygon is a face of polyhedron $K = C \cap C'$.

5 Plane sections of the voxel space

The study of the geometry of plane sections of voxels tilings is also called *information extraction* in Medical Imaging; it is a crucial part in our intersection algorithm.

As can be seen on picture 5, the question is to describe all intersections of a given plane with the tiling made of all voxels filling space $\mathbb{R}^3$. That is we want to describe all intersection polygons of this plane with each voxel it meets.

But, as is well known, approximate knowledge of geometrical objects (as polygons and polyhedra) destroys most of their theoretical properties avoiding a complete study and implementation of degenerate cases. This is why we do not look only for mere numerical approximations of the vertices of these polygons, but for analytical expressions of their coordinates.

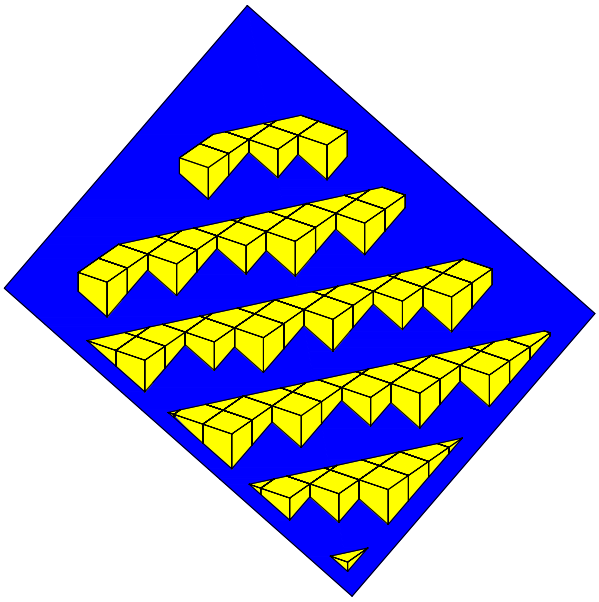

Fig. 5. Plane section of voxels tiling.

Figure 5 shows some voxels cut by a plane (P) on which they induce an ordinary 2D tiling (this tiling of the plane must not be confused with the tilings of $\mathbb{R}^3$ induced by voxels).

This 2D tiling, illustrated in the following picture 6, is built by intersecting the plane with three families of isothetical planes whose equations are $x = k, y = l, z = m$, and where k, l, m are integers (here we consider *small* voxels are unit cubes).

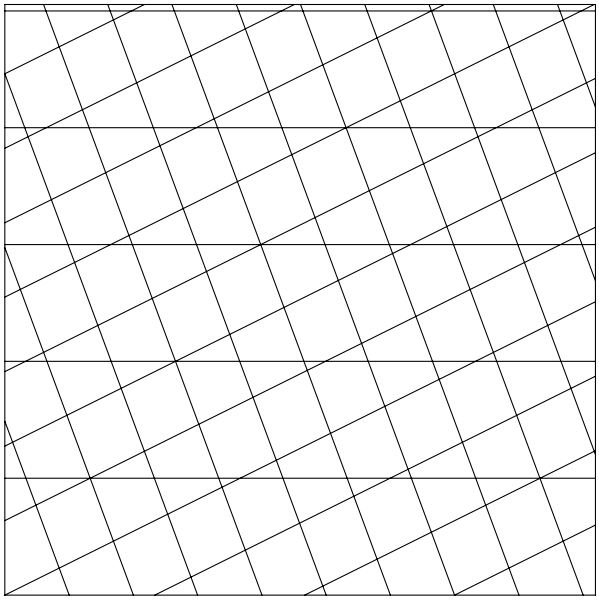

Fig. 6. 2D tiling induced by voxels on section plane.

Let us suppose our intersecting plane (P) goes through 0 and has an integer

normal vector (a, b, c), where $0 \leq a \leq b \leq c$, and $\gcd(a, b, c) = 1$. Thus equation of (P) is $ax + by + cz = 0$. Constrains prescribing (a, b, c) being integer may seem surprising, but we recall that any real direction u in $\mathbb{R}^3$ possesses arbitrarily close diophantine approximations, and, moreover, it is easy to find integers a, b, c such that $angle(u, (a, b, c))$ is arbitrarily small. We shall also see that constrains $0 \leq a \leq b \leq c$ can be relaxed without producing mountains of control tests if octahedral group O_h is used. This last point will be treated below.

A simple computation, made in a basis attached to plane (P), shows that above mentioned tiling is induced by the following family of four parameters lines, x being the variable as usual.

$d_1(x, a, b, c, k) = k\sqrt{(}a^2 + b^2 + c^2)/sqrt(a^2 + b^2)$

$d_2(x, a, b, c) = (ax - k\sqrt{(}a^2 + b^2))sqrt(a^2 + b^2 + c^2)/bc$

$d_3(x, a, b, c) = -(bx + k\sqrt{(}a^2 + b^2))sqrt(a^2 + b^2 + c^2)/ac$

Figure 6, which shows tiling induced by space voxels on the plane (P) whose equation is $3x+7y+13z = 0$, has been drawn using these three families of lines.

6 Arithmetization of the tiling of the plane. Expressing periodicity.

A closer look at figure 6 shows existence of periods, that is of identical tiles. This is one of the (nice) consequences of our arithmetical hypothesis concerning (P) normal vector (a, b, c). The problem is to express these periods, that is *two translation vectors* and *all tiles* of one period; this is where arithmetics comes in.

Each voxel being attached to its lower, left, bottom vertex (x, y, z), we denote by $f(x, y, z)$ the linear form $ax + by + cz$. It is clear that distances of integer points $M = (x, y, z)$ and $M' = (x', y', z')$ to plane (P) are equal if and only if $|f(x, y, z)| = |f(x', y', z')|$ and that M and M' are in the same half-space delimited by (P), and are at the same distance to (P) if and only if $f(x, y, z) = f(x', y', z')$.

We deduce from this that to cubes C and C' respectively attached at M and M' have the same intersection polygon with plane (P) if and only if $f(x, y, z) = f(x', y', z')$.

Periodicity in the tiling of plane (P) is thus equivalent to solving the diophantine problem

$$ax + by + cz = 0 \tag{1}$$

that is of finding integer solutions (x, y, z) of this equation.

While several treatises give complete proofs and algorithms to solve general systems of linear diophantine equations, (see for example [7]), we shall briefly describe one algorithm due to Blankinship (see [3]), which solves the restricted

case of equation (1). Understanding and use of this algorithm should be clear once we remind that solutions of (1) make a 2D lattice contained in (P) and that a lattice has many basis, but that a free family of 2 vectors is not necessarily a basis.

Given diophantine equation $ax + bx + cz = 0$ we build the matrix

$$\begin{pmatrix} a & b & c \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

and apply columns operations driven by first line entries a,b,c in order to cancel two values among them. Values in lines $2, 3, 4$ located below *both zeros of line* 1 of the resulting matrix give two vectors of $\mathbb{Z}^3$ making a basis of the solutions of (1).

Let us solve diophantine equation $3x+7y+13z = 0$ to illustrate this process which is a simple integer clone of Gauss algorithm. Following matrices show successive steps which have to be done.

$$\begin{pmatrix} 3 & 7 & 13 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 3 & 1 & 1 \\ 1 & -2 & -4 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 7 & 2 & -4 \\ -3 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix}$$

A possible base is thus $(7, -3, 0), (2, 1, -1)$ as the reader may verify.

Using the same ideas he (she) shall be able to devise algorithms to solve diophantine equation $ax + by + cz = k$, where k is another integer parameter (*hint:* start solving $ax + by + cz = 1$).

Knowing a basis $\{v_1, v_2\}$ of the lattice associated to equation (1) solves the periodicity question for the plane lattice in the following way. If $M = (x, y, z)$ is an integer point such that voxel attached at M cuts plane (P) then all integer points where attached voxel cuts (P) with the same polygons (within translation) are points

$$M + k_1 v_1 + k_2 v_2 \tag{2}$$

where k_1 and k_2 are arbitrary integers.

7 Explicit description of intersecting polygons.

Once period vectors of the tiling of (P) are obtained, it remains to describe precisely each of the tiles appearing within one period. All these tiles are plane sections of a given unit cube. Let us suppose for the moment that this cube (C) is $[0, 1]^3$, $[0, 1]$ being the unit interval of $\mathbb{R}$. We denote by F_1 the

subset of directions of $\mathbb{R}^3$ satisfying

$$F_1 \quad \begin{cases} 0 \leq a \leq b \leq c \\ a+b \leq c \end{cases} \tag{3}$$

and similarly by F_2 the subset

$$F_2 \quad \begin{cases} 0 \leq a \leq b \leq c \\ a+b \geq c \end{cases} \tag{4}$$

Theorem 7.1 *The sequence of vertices of the unit cube crossed by the planes $ax+by+cz=k$, when k varies from $-\infty$ to ∞, is independent of (a,b,c) as long as it stays in F_1.*

Of course this result is also true when $(a,b,c) \in F_2$, but order of crossed points changes.

A few words about this theorem, before we give the precise order in which vertices are crossed, should explain its object.

Whatever the plane cutting a cube, vertices of the resulting intersection polygon are taken among the 12 intersections of this plane with the 12 support lines of the cube edges.

But the number of edges of plane (non degenerate) sections of a cube is greater than 3 and lower than 6, meaning that some of the previous intersections points must be eliminated to get the right polygons. For each polygon this theorem will first help determining its vertices among the 12 possible points, (remark that the choice changes with the polygon type), and second, but this is a programmer's consideration, also allow to list these vertices in the order of their convex hull.

If, for example, just triangles are considered among all polygons, it is clear that there will be only 8 cases (close to cube vertices); but the number of cases for other polygons types would be much more difficult to find without former theorem, and worse, the whole programming of the formulas giving coordinates of their vertices would be very boring and difficult to debug.

If plane (P) is moved parallel to itself, under the constrain $a+b \leq c$, it is immediately seen that the order in which (C) vertices are crossed is given by the *sorting in increasing order* of the eight values $f(v_1), f(v_2), \ldots, f(v_8)$ taken by form f on (C) vertices. Definition of cube $(C) = [0,1]^3$ gives the evaluation of these 8 values which are $0, a, b, c, a+b, a+c, b+c, a+b+c$, and hypothesis on $(a,b,c) \in F_1$ shows immediately that their increasing sorting is:

$$0 < a < b < a+b < c < a+c < b+c < a+b+c \tag{5}$$

As f is proportional to the distance to plane (P), this set of inequalities says that the order in which (C) vertices are crossed by (P) displacement is:

$(0,0,0)$, $(1,0,0)$, $(0,1,0)$, $(1,1,0)$, $(0,0,1)$, $(1,0,1)$, $(0,1,1)$, $(1,1,1)$.

As (C) edges meeting at $(0,0,0)$ are $[(0,0,0),(1,0,0)]$, $[(0,0,0),(0,1,0)]$ and $[(0,0,0),(0,0,1)]$, we deduce that if (P) is such that f value satisfies

$0 \leq f(x,y,z) < a$, this plane cuts these three edges of (C) and no other one. That is intersection $(C) \cap (P)$ is a triangle. Moreover coordinates of the vertices of this triangle can be computed formally.

8 The fundamental sections of a cube

While present cube (C) is very convenient to explain how we can find all its plane sections, equivalent but more useful formulas are obtained when (C) is centered at origin and when its size is any number; these formulas are given below (eqs(06)). Present definition of cube (C) nevertheless shortens explanations.

Let us go on and find the next intersection polygon. As next crossed vertex is $(1,0,0)$, two new edges, $[(1,0,0),(1,1,0)]$ and $[(1,0,0),(1,0,1)]$ are cut, while $[(0,0,0),(0,1,0)]$, $[(0,0,0),(0,0,1)]$ are also intersected, but segment $[(0,0,0),(1,0,0)]$ is not. Thus when f value satisfies $a \leq f(x,y,z) < b$ intersection polygon $(C) \cap (P)$ is a quadrilateral, which is, after inspection, a trapezoid because planes associated to $[(0,0,0),(0,1,0),(0,0,1)]$ and $[(1,0,0),(1,1,0),(1,0,1)]$ are obviously parallel. Such a case is illustrated by fig. 7

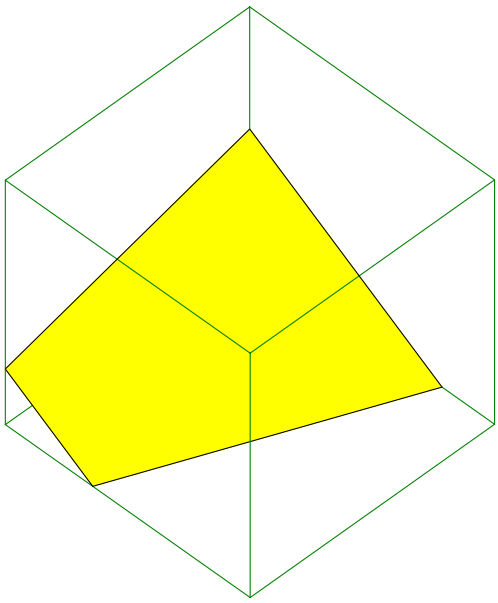

Fig. 7. One plane section of a cube.

Let us still treat the next case ant let the reader treat remaining ones, (all results are gathered in the functions $tri(k)$, etc. presented below in eqs(06)).

It suffices finding which (C) edges are cut when vertex $(0,1,0)$ is crossed. Edge $[(0,0,0),(0,0,1)]$ starting from first vertex $(0,0,0)$ is still cut, as segment $[(1,0,0),(1,1,0)]$ and edge $[(1,0,0),(1,0,1)]$, but two new edges, precisely $[(0,1,0),(1,1,0)]$, and $[(0,1,0),(0,1,1)]$ are also intersected by (P). Thus in case $a \leq f(x,y,z) \leq b$, plane section of cube (C) is a five edges polygon.

Former description of (C) plane sections supposes $(a,b,c) \in F_1$; other case $(a,b,c) \in F_2$ is treated similarly. Going from F_1 to F_2 modifies only slightly the order in which (C) vertices are traversed. Fourth case polygon is changed from parallelogram to hexagon, the 6 crossed edges being $[(1,0,0),(1,1,0)]$, $[(1,0,0),(1,0,1)]$, $[(0,1,0),(1,1,0)]$, $[(0,1,0),(0,1,1)]$, $[(0,0,1),(1,0,1)]$ and $[(0,0,1),(0,1,1)]$. Thus only five kinds of polygons are obtained when a cube is cut by a plane : triangle, trapezoid, pentagon, parallelogram and hexagon. It is easy to collect the five functions giving their vertices into one procedure, the forthcoming *sct(k)*, which is well defined when $(a,b,c) \in F_1 \cup F_2$. We denote $F = F_1 \cup F_2$.

As we already said the most useful formulas are obtained when cube (C) is symmetric with respect to origin, for example when $(C) = [-\ell,\ell]^3$. In this case (C) edges length is 2ℓ, ℓ being a new parameter. Reader can notice procedure *sct(k)* uses very few inequalities to give all vertices of any plane section of cube (C) when $(a,b,c) \in F$. He (she) can also verify that these formulas are continuous functions which give the right limit (degenerated) polygon when parameters tend toward special values. This implies that these formulas also give particular cases of cubes plane sections. For example vertices of a parallelogram parallel to an edge will converge to it if the section plane tends to this edge. Then we shall rely on *cube symmetries* to get (C) plane sections for *any* normal vector (a,b,c), using some elementary group theory.

All these formulas, expressed as functions of parameter $k = f(x,y,z)$, for cube $[-\ell,\ell]^3$ are given below:

$$
\begin{aligned}
tri:\ k \to\ & [[\ell,(k-\ell*(a+c))/b,\ell],[\ell,\ell,(k-\ell*(a+b))/c],\\
& [(k-\ell*(b+c))/a,\ell,\ell]]\\
tra:\ k \to\ & [[\ell,(k-\ell*(a+c))/b,\ell],[\ell,\ell,(k-\ell*(a+b))/c],\\
& [-\ell,\ell,(k-\ell*(-a+b))/c],[-\ell,(k+\ell*(a-c))/b,\ell]]\\
par:\ k \to\ & [[-\ell,-\ell,(k+\ell*(a+b))/c],[\ell,-\ell,(k-\ell*(a-b))/c],\\
& [\ell,\ell,(k-\ell*(a+b))/c],[-\ell,\ell,(k-\ell*(-a+b))/c]]\\
pen:\ k \to\ & [[(k+\ell*(b-c))/a,-\ell,\ell],[\ell,-\ell,(k-\ell*(a-b))/c],\\
& [\ell,\ell,(k-\ell*(a+b))/c],[-\ell,\ell,(k-\ell*(-a+b))/c],\\
& [-\ell,(k+\ell*(a-c))/b,\ell]]\\
hex:\ k \to\ & [[(k+\ell*(b-c))/a,-\ell,\ell],[\ell,-\ell,(k-\ell*(a-b))/c],\\
& [\ell,(k-\ell*(a-c))/b,-\ell],[(k-\ell*(b-c))/a,\ell,-\ell],\\
& [-\ell,\ell,(k-\ell*(-a+b))/c],[-\ell,(k+\ell*(a-c))/b,\ell]]
\end{aligned}
\tag{6}
$$

Any plane section of cube (C), when $(a,b,c) \in F$, may be given by the following algorithm using a simple test control based on k and $a+b \le c$ or

$a+b \geq c$ to call the right function among the former 5 ones. Fig. 9 shows all plane sections of cube (C) giving all tiles of a plane section of a voxels tiling.

```
sct:= proc(k)
local eps,u;
if is(-l*(a+b+c) <= k) and is(k <= l*(a+b+c)) then
  if is(k >= 0) then
    eps := 1: u := k
  else
    eps := -1: u := -k
  end if;
  if is(c < a+b) then
    if is(0 <= u) and is(u < l*(a+b-c)) then
      eps * hex(u)
    elif is(l*(a+b-c) <= u) and is(u < l*(a-b+c)) then
      eps * pen(u)
    elif is(l*(a-b+c) <= u) and is(u < l*(c-a+b)) then
      eps * tra(u)
    else
      eps * tri(u)
    end if
  else
    if is(0 <= u) and is(u < l*(c-a-b)) then
      eps * par(u)
    elif is(l*(c-a-b) <= u) and is(u < l*(a-b+c)) then
      eps * pen(u)
    elif is(l*(a-b+c) <= u) and is(u < l*(c-a+b)) then
      eps * tra(u)
    else
      eps * tri(u)
    end if;
  end if;
end if;
end proc;
```

This procedure gives a formal description of all plane sections of a cube when parameters a, b, c satisfy F_1 or F_2, that is when $(a, b, c) \in F$; set F is the *fundamental domain* of (C) symmetry group. Of course analogous formulas exist for the other cases, but finding them using *control tests* would be very awkward: code writing and debugging would be long and tedious tasks. Using octahedral group, which is the group of symmetries of a cube, will simplify this generalization.

9 The octahedral group of symmetries of a cube.

This group, denoted O_h, can be identified with the product of group $(\mathbb{Z}/2\mathbb{Z})^3$ of order 8 with the group $\mathcal{S}_3$ of permutations on three letters, of order 6. Order of O_h is thus equal to 48.

While O_h is usually interpreted with the help of symmetries of a cube (like (C)), we shall depart from this course to adopt an equivalent one where O_h reduces any space direction to a canonical one belonging to its fundamental domain F defined by inequalities $0 \leq a \leq b \leq c$.

In the sequel, first factor of O_h, that is group $(\mathbb{Z}/2\mathbb{Z})^3$, will correspond to the obvious eight signs arrangements of the three coordinates a, b, c of a $\mathbb{Z}^3$ vector, while second factor $\mathcal{S}_3$ will map to the 6 permutations of these three values.

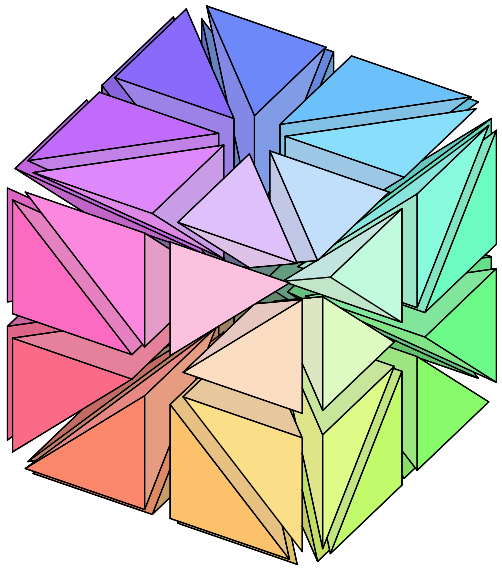

Fig. 8. Geometrical view of octahedral group.

So any direction of space is equivalent to one of F through an element of O_h. This group induces a triangulation of (C) boundary in 48 triangles. The triangle which is the intersection of the fundamental domain F with (C) boundary: $\{(x, y, z) \mid \max(|x|, |y|, |z|) = \ell\}$ will also be denoted by F.

Otherwise said, for any vector v of Z^3 there is a $g \in O_h$ such that $g.v$ belong to F. Of course $g^{-1}.g.v = v$, that is v can be easily recovered.

10 Implementing group O_h

We detail this point, which may be puzzling to reader not acquainted with formal computing systems, and also because several solutions exist, some of which leading to inefficient coding.

The solution we retain uses the *linear representation* of group O_h which is given by the set of 3×3 matrices where each line and each column has one and

only one non-zero entry equal to 1 or -1. It is a simple verification to show that this subgroup of $O(3)$ (orthogonal group of dimension 3) is isomorphic to O_h. Important point is to find element g which will reduce a given vector v as claimed above.

Algorithm giving matrix g associated to $v = (a, b, c)$ works as follow:

- build the 4×3 table bounding the 3×3 unit matrix with a first line equal to (a, b, c), *(same construction as the previous one starting Blankinship algorithm).*
- multiply columns by the sign of their first element (if $a = -6$, multiply column 1 by -1). Thus all elements of first line become positive (or null) and the 1 below will keep the precious sign. Take care giving sign 1 to null elements *(and not sign* 0 *as some compilers do).*
- then exchange columns so that first line is sorted in increasing order. Resulting 3×3 matrix below first line, denoted by h, is the inverse of the looked for g element, that is $transpose(h).v = w$ belongs to F and $h.w = v$. As O_h is contained in $O(3)$ we have $g = h^{-1} = transpose(h)$.

Keeping matrix h or $transpose(h) = g$ is mathematically equivalent, but from the coding point of view, w being given as the first line of previous algorithm there is no need for g; on the contrary it is h which is useful to transform w (and all data found while $(a, b, c) \in F$) back to the domain of v.

Pseudo-code for this algorithm giving the linear operator h associated to vector $v = [v[1], v[2], v[3]]$ may be written as follows.

LinearRepr:= **proc**(v)
local M;
$M := [[v[1], sgn(v[1]), 0, 0], [v[2], 0, sgn(v[2]), 0], [v[3], 0, 0, sgn(v[3])]]$;
$M := sort(M, (x, y) \rightarrow is(abs(x[1]) \leq abs(y[1])))$;
$[abs(M[1][1]), abs(M[2][1]), abs(M[3][1])]$,
  $array([[M[1][2], M[2][2], M[3][2]],$
  $[M[1][3], M[2][3], M[3][3]],$
  $[M[1][4], M[2][4], M[3][4]]])$;
end proc:

11 Relation between sections of voxels space and digital planes.

There are several way of extracting plane sections from a 3D image. First one was indicated above and starts cutting all voxels with a given plane leading to the plane tiling already studied. Second step of this approach consists in mapping these tiles to pixels overlapping this tiling. A precise treatment of this correspondence supposes determining all intersections of tiles and pixels, which can be done using an algorithm for the intersection of two convex polygons

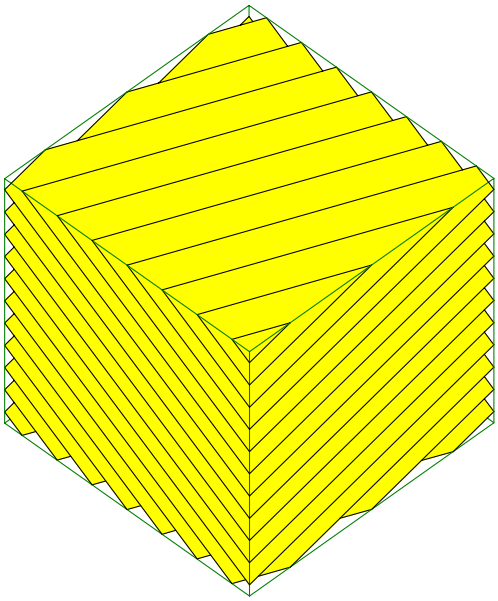

Fig. 9. Plane sections of a cube are the tiles of one period.

(see §15). We do not go into all the details of this first extract method but rather present a simpler one using digital planes.

Correspondence between voxels and integer points has to be specified. Voxel attached to point (x, y, z), denoted $V(x, y, z)$, is the cube which is the cartesian product of intervals $[x, x+1] \times [y, y+1] \times [z, z+1]$, so that (x, y, z) is its left, lower, bottom vertex. Consequently voxel $V(x, y, z)$ cuts plane (P) whose equation is $ax+by+cz = 0$, (condition $(a, b, c) \in F$ being still fulfilled) if and only if:

$$ax + by + cz \leq 0 \quad \text{and} \quad 0 \leq a(x+1) + b(y+1) + c(z+1)$$

Thus voxels intersecting (P) belong to the *thick digital plane* defined by

$$-(a + b + c) \leq ax + by + cz \leq 0.$$

This set of voxels is 6-connected and thicker (whose name) than the naive 18-connected digital plane defined by $-c \leq ax + by + cz \leq 0$ (remark $c = \max(a, b, c)$). This shows that there are at most $a + b + c$ different tiles and that using naive digital planes is not sufficient to extract slices in 3D images; better results are obtained with 6-connected thick digital planes.

It is nevertheless possible to get good approximations using judicious 18-connected digital planes centered on the euclidian plane defined by $-c/2 \leq ax + by + cz < c/2$ because these will give the largest tiles.

12 Tiles areas

As the number, five, of tiles of voxels space plane sections is rather small, it is conceivable looking for the formulas expressing their areas in function of the parameters a, b, c (we choose $\ell = 1$ for simplicity).

Let us denote $A_0 = sqrt(a^2 + b^2 + c^2)/abc$, then we have

Type	Section	Area
triangle	$tri(k)$	$k^2 A_0$
trapezoid	$tra(k)$	$(2k - a)aA_0$
parallelogram	$par(k)$	$2abA_0$
pentagon	$pen(k)$	$(2ab - (a + b - k)^2)A_0$
hexagon	$hex(k)$	$(2ab - (a + b - k)^2 - (k - c)^2)A_0$.

With these formulas it is easy, for example, to bound errors made if triangles are neglected.

Most interesting result concerning these tiles areas formulas is that they define a piecewise continuous function of variable k which is, surprisingly, everywhere differentiable.

13 Intersection of two voxels tilings and quadratic cubes.

As mentioned in the introduction our main interest lies in the intersection of two sets of voxels each one tiling $\mathbb{R}^3$ space. Tiles of each partition are identical cubes. First one can be seen as unit cube of $\mathbb{R}^3$ and second one as a cube of arbitrary size and orientation.

In order to find periodicities among intersection polyhedra of voxels of each family, second cube must possess rather special properties to be explained right now.

Contrary to the 2D situation where exist an infinite number of squares of arbitrary orientation having integer vertices, there does not exist cubes of any orientation having integer vertices. But it is easy to see that we can construct cubes of almost any direction using quadratic real numbers, called *quadratic cubes*; this is, in a certain sense the simplest family after integer cubes which have to be isothetic or parallel to the bissector planes.

Quadratic cubes offer a second advantage : all computations can be made exactly, at least with a computer algebra system (Maple, Mathematica, etc.) and this allows an exact study of the geometry of the intersection polyhedron of two such cubes.

It is not difficult to see that given an integer vector $v_1 = (a, b, c)$ approximating a direction in $\mathbb{R}^3$ space, arbitrary quadratic directions can be found in plane $ax + by + cz = 0$; let v_2 be one such vector. Cross product $v_1 \wedge v_2 = v_3$ gives a third quadratic vector such that v_1, v_2, v_3 is an orthogonal basis. Di-

viding v_1 by its norm, $\|v_1\|$, does not change the algebraic property of this basis which becomes orthonormal.

So any real cube in general position may be arbitrarily approximated by quadratic cubes.

14 Intersection of cubes.

Now comes the main part which is to design a reasonable algorithm to compute the intersection polyhedron of two given cubes; first one may be a unit cube and second one is supposed to be a quadratic one as above. The requirement is more involved than can be thought at first because we want exact results and an efficient algorithm with the lowest possible complexity.

A very superficial analysis of the problem shows that even for two simple cubes the number of computations to be carried out in order to get their intersection is rather high. A brute force algorithm would begin finding all vertices, then build their convex hull. Clearly these vertices can be obtained intersecting edges of one cube with the faces of the other one. This leads to $144 = 2 * 6 * 12$ vertices. Most of them have to be deleted because one intersection point must belong to the edge and to the square face considered and not only to the support line or plane. This leads to 2 control tests for each edge and to 4 control tests for each face; thus 6 control tests have to be verified for each intersection point, giving, on the whole, 864 control tests; adding the 144 intersections and some more trifle shows that the order of magnitude is not far from one thousand elementary computations.

This is why a more efficient solution, one which specially avoids control tests, is welcome. This explains our use of octahedral group to get plane sections of cubes. See above $sct(k)$ algorithm where only 2 control tests are carried to get one plane section polygon when (P) normal vector (a, b, c) belongs to the fundamental domain F. But with any (a, b, c) normal vector computation overhead is small because a simple sorting of $|a|, |b|, |c|$ has to be conducted to give the reducing $h \in O_h$ element.

As explained at the end of §4, our intersection algorithm consists in first finding the polygons cut on each cube by the support planes of the faces of the other one and, second, intersecting each polygon with the square face of its support plane. Chance to cube plane sections formulas of §8, the polygons are obtained very efficiently; this is the main point of our approach. It only remains intersecting these polygons with their corresponding square faces.

15 Intersection of convex polygons

Once plane section polygons are obtained, and their are at most 12 of them, it remains intersecting each of them with the square face which defined its support plane. This is a particular case of the well known intersection of convex polygons problem. A clever linear algorithm (in $O(m + n)$, m, n respective

edges numbers of each polygon) was given by O'Rourke (cf. [10]). We shall briefly recall its principle mainly to insist on particular points O'Rourke left out but which are crucial for our kind of data.

Clever notion O'Rourke introduced is that of an oriented edge, let us say $B = b_0b_1$, *aiming toward* another one, let say $A = a_0a_1$. Let $u \to f_A(u)$ denotes the linear form associated to A (or its support line) which is $f_A(u) = 0$ if u belongs to A support line, $f_A(u) > 0$ if $det(A, a_0u) > 0$ etc.

We say that $B = b_0b_1$ aims toward $A = a_0a_1$ if an only if ($det(A, B) \geq 0$ and $det(A, a_0b_1) < 0$) or ($det(A, B) <= 0$ and $det(A, a_0b_1) > 0$).

Picture 10 shows some cases of vectors B aiming toward A and some others (the B's), which do not aim toward A.

Both convex polygons being positively oriented, one edge is chosen on each curve and an alternate pursuit is opened. Each time the pursuit is stopped, the intersection point of the two current edges is computed and if it is non trivial, (segments may have a void intersection even if their support do not), it is supposed that curves cross at this point. It is easy with the help of a boolean variable to follow which of the two curves lies inside the other one. Once all edges have been traversed the algorithm is over.

Unfortunately the case where both curves may be tangent was not treated by O'Rourke and this occurs in our application. This does not need tremendous changes in his code but it has to be done. Other issues with this algorithm concern non intersecting curves either because they are disjoint or because one in contained in the interior of the other one. These cases are also easily treated.

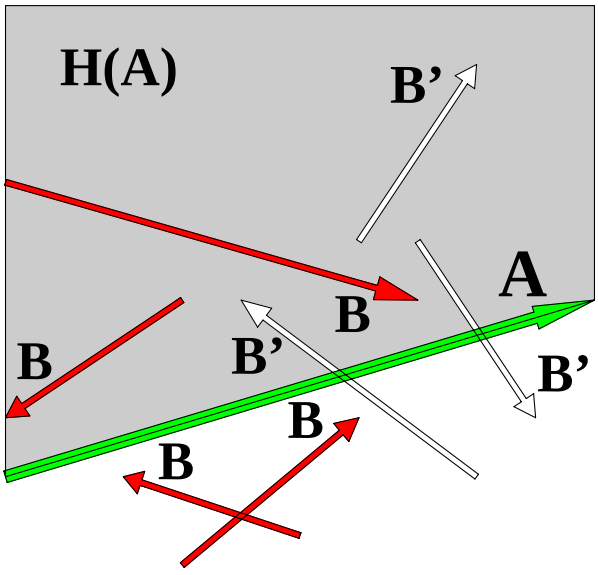

Fig. 10. O'Rourke's notion of vector B tending toward A.

Main algorithm implementing O'Rourke's algorithm and necessary modifications follows. It uses two readily written functions: $left(M, V)$ which says if

point M is located on the left side of vector M and procedure interSgt(Sg1,Sg2) which gives the intersection point of segments Sg1 and Sg2 or *void* if they do not intersect.

```
interCb:=proc(C1,C2)
  local a,b,a1,b1,A,B,aadv,badv,aHB,bHA,
    inside,lst,orient,pt;
  a,b:= 1,1;
  inside:="unknown";
  aadv,badv:=1,1:
  lst:=NULL;
  while (aadv <= nops(C1) or badv <= nops(C2)) do
    a1:=(a+nops(C1)-2 mod nops(C1))+1;
    b1:=(b+nops(C2)-2 mod nops(C2))+1;
    A,B:=C1[a]-C1[a1],C2[b]-C2[b1];
    orient:=A[1]*B[2]-A[2]*B[1];
    bHA:=left(C2[b],[C1[a1],C1[a]]);
    aHB:=left(C1[a],[C2[b1],C2[b]]);
    pt:=interSegt([C1[a1],C1[a]],[C2[b1],C2[b]]);
    if pt <> NULL then
      if not type(op(2,pt),string) then
        lst:=lst,pt;
        if inside="unknown" then aadv,badv:=1,1 end if;
        if aHB then inside:="C1in" else inside:="C2in" end if;
      else
        lst:=lst,op(1,pt); inside:=op(2,pt);
      end if
    end if;
    if is(orient >= 0) then
      if bHA then
        if inside="C1in" then lst:=lst,C1[a] end if;
        a,aadv:=(a mod nops(C1))+1,aadv+1;
      else
        if inside="C2in" then lst:=lst,C2[b] end if;
        b,badv:=(b mod nops(C2))+1,badv+1;
      end if;
    else
      if aHB then
        if inside="C2in" then lst:=lst,C2[b] end if;
        b,badv:=(b mod nops(C2))+1,badv+1;
      else
        if inside="C1in" then lst:=lst,C1[a] end if;
        a,aadv:=(a mod nops(C1))+1,aadv+1;
      end if;
    end if;
```

```
    end do:
#answer is prepared
  if lst <> NULL then
    [lst]; # C1 and C2 intersect non trivially
  else
    if (not left(C1[1],[C2[1],C2[2]])) and (not left(C2[1],[C1[1],C1[2]])) then
      [lst]; # C1 and C2 are disjoint
    elif left(C1[1],[C2[1],C2[2]]) then
      C1; #C1 is inside C2
    else
      C2; # C2 is inside C1
    end if;
  end if;
end proc:
```

This algorithm is clearly linear; procedure InterCb being the longest among those of our program, (written in Maple language), it gives an idea of the very reasonable size of the whole code for the intersection of two cubes after our approach. Picture 11, showing an example of cubes intersection, illustrates its use.

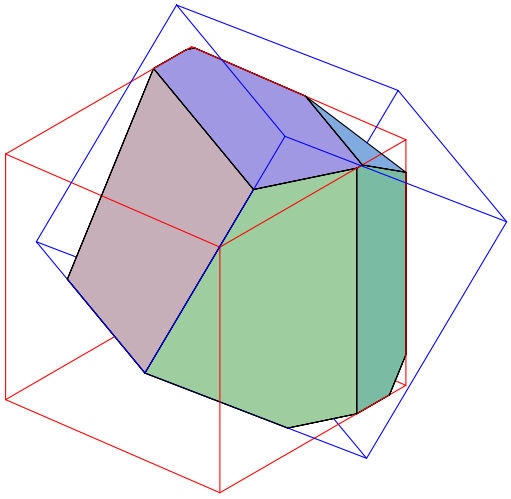

Fig. 11. Intersection of two voxels.

16 3D intersection scheme and generalized continued fractions

Plane sections introduced in §5 showed a first use of voxels tilings underlying lattices. This use of lattices (as subgroups of $\mathbb{Z}^3$) is still limited until nothing is said about the intersection scheme of two voxels tilings and its relationship with higher dimensional continued fractions. This last subject being rather

difficult and long to explain, we just mention which of its features are obviously related to the 3D digital coordinates change problem.

As seen in §2, optimizing intersections of voxels of two tilings leads considering the intersection scheme of small voxels (supposed to be usual unit cubes) with the boundary of an orthogonal cone induced by three adjacent faces of a large voxel (i.e. 3D analogue of fig. 1). A 3D extension of prop. 2.1 is obviously true when Klein's funnel is replaced with the convex hull of integer points (small voxels) contained in this cone. That is all kinds of intersections of couples of small and large voxels can be found among intersections of the small voxels belonging to a set of *rational digital planes* with the orthogonal cone.

It occurs that Arnold (cf. [1]), for much different motivations, showed that convex hulls of integer points contained in 3D cones, (his theory has even sense for any dimension), are the natural generalization of ordinary continued fractions. Such hulls are now called *Arnold-Klein sails*. Thus intersection schemes of two voxels spaces and Arnold-Klein sails are in fact the same thing.

17 Conclusion

Implementation of intersection of two digital views of a same object having neighboring resolutions needs to compute intersections of overlapping voxels. A formal approach of this question using as much as possible particular structure of cubes, and specially their symmetries, shows that their plane sections can be described by exact formulas which are still valuable for degenerate cases. This avoids the complicated coding generally met in such situations.

New directions for optimization of intersection of large numbers of overlapping voxels using multidimensional continued fractions are indicated. Interference of this theory into 3D Imaging should deserve much attention in the future.

Though first application of our work concerns the fusion of distinct modalities quantities in 3D Medical Imaging, it should be useful in many domains where digital coordinates changes are present.

References

[1] Arnold, V. I., *Higher dimensional continued fractions*, Regular and Chaotic Dynamics (1998).

[2] Barra, V., "Fusion d'images 3D du cerveau: études de modèles et applications," Ph.D. thesis, ERIM, Université d'Auvergne (1993).

[3] Blankinship, W. A., *A new version of the euclidean algorithm*, Amer. Math. Monthly (1963).

[4] Davenport, H. and J. H. Davenport, "The Higher Arithmetic: An Introduction to the Theory of Numbers, Seventh Edition," Cambridge University Press, 1999.

[5] Dobrindt, K., K. Mehlhorn and M. Yvinec, *A complete and efficient algorithm for the intersection of a general and a convex polyhedron*, Rapport de recherche INRIA (1993).

[6] Foley, van Dam, Feiner and Hughes, "Computer Graphics, Principles and Practice," Addison-Wesley, 1990.

[7] Grötschel, M., L. Lovász and A. Schrijver, "Geometric Algorithms and Combinatorial Optimization, Second Corrected Edition," Springer-Verlag, 1994.

[8] Hardy, G. H. and E. M. Wright, "An Introduction to the Theory of Numbers, Fifth Edition," Oxford Press, 1980.

[9] Martin, A. K., "A Simple Primal Algorithm for Intersecting 3-Polyhedra in Linear Time," Ph.D. thesis, Department of Computer Science, University of British Columbia, 2366 Main Mall, Vancouver, B.C. CANADA V6T IZ4 (1993).

[10] O'Rourke, J., "Computational Geometry in C," Cambridge Press, 1990.

[11] Overmars, M. H., *Designing the computational geometry algorithms librairy cgal*, in: *Proceedings Workshop on Applied Computational Geometry*, 1996.