# Planning data structure for space partitioning algorithms

Gábor Fábián and Lajos Gergó

***Abstract*—**In this paper we give a new data structure that can be used efficiently by space partition. We consider an approximation schema introduced recently which is a generalization of space partitioning algorithms. The input of the schema is triangular mesh, in special case it can be a set of space points as well. The approximation schema is an iterative process. The first approximation of the input mesh is its bounding box, or an arbitrary convex polyhedron containing the mesh. The process gives an atomic decomposition of the initial polyhedron. Every iterative step an atom is chosen, and divided by a plane into two disjoint atoms. In the n-th step the decomposition consists n+1 atoms. We give a sufficient condition for the convergence of the method, minimizing a volume based error metric and leaving the atoms that are irrelevant to the approximation. It can be shown, the convergence depends only on the strategy of the choosing and dividing. We define a data structure for the atomic decomposition, where vertices and faces of atoms are contained in a global list for minimizing the redundancy. We give a short survey of geometric operations and properties of polyhedra, the dividing operation of the process will be discussed, as well.

***Keywords*—** data structures, polyhedral approximation, spatial decomposition, volume based approximation

## I. Introduction

THIS paper focuses on approximating algorithms of subsets of real plane or space. As we will see, there are many possible applications of approximation algorithms of these sets. If the input set contains finitely many points, the approximation problem can be seen as a kind of space partitioning [5][16]. If the input set is a polygonal mesh, then it can be considered as boundary of a spatial subset, then mesh repairing [1], mesh simplification, approximate convex decomposition [10][12], can be obtained by the approximation process. Similarly, if input is a closed parametric surface, we can modify the process to obtain mesh generation or finite element methods [15].

The subjects mentioned before are very close to each other, our goal is to unify them into a single method, a general approximation algorithm. Different tuning of parameters fits various approximation problems. First, we give an overview of the process, thereafter we discuss its relation to other methods, finally we show a few examples about how should we tune the parameters to obtain different approximation methods.

Gábor Fábián is with the Eötvös Loránd Science Universtiy, Faculty of Informatics, 1117 Budapest (e-mail: robagnaibaf@ gmail.com).

Lajos Gergó is with the Eötvös Loránd Science Universtiy, Faculty of Informatics, 1117 Budapest (e-mail: gergo@ inf.elte.hu).

## II. Approximation Schema

Let us consider $S$ a bounded subset of the plane or the space $\mathbb{R}^d$ ($d=2,3$). If $S$ is bounded then it can be transformed into the $d$-dimensional unit ball, i.e. we can assume, that $S \in \mathbb{B}^d$. The main idea is constructing atomic decomposition of the ball, which is obviously is a finite dimensional subspace of the unit ball. By Riesz projection theorem we can draw up immediately the best approximation of any $S \in \mathbb{B}^d$, it is the Fourier-series with respect to the subspace spanned by the atoms [3][13]. If the decomposition, $\mathcal{B}_0,\ldots,\mathcal{B}_n,\ldots$ is iteratively refined, we get a sequence of approximations $S_0,\ldots,S_n,\ldots$. In [6] we formulated some statements for the approximation sequence to be monotonic and convergent in a metric defined on "solid" sets of $\mathbb{B}^d$. Let us take a short survey of the mathematical details of the approximation schema:

Let $\mathcal{B}_n := \sigma(B_i^{(n)} \subset \mathbb{B}^d : i = 0,1,\ldots,n)$ be a finite atomic σ-algebra on $\mathbb{B}^d$ [14], i.e. for $n \in \mathbb{N}$

$$\mathcal{B}_n := \sigma(B_i^{(n)} \subset \mathbb{B}^d : i = 0,1,\ldots,n)$$
$$B_i^{(n)} \cap B_j^{(n)} = \varnothing \quad (i \neq j)$$
$$\cup_{i=0}^{n} B_i^{(n)} = \mathbb{B}^d$$

It is easy to check [4][17], that

$$\phi_i^{(n)} := \frac{1}{\sqrt{\mu(B_i^{(n)})}} \chi_{B_i^{(n)}} \quad (i = 0,1,\ldots,n)$$

functions form orthonormed system under the common inner product of function spaces, $\langle .,. \rangle$. Here $\chi_{B_i^{(n)}}$ denotes the indicator function of the set $B_i^{(n)} \in \mathbb{B}^d$. Then for all $f$ function the Fourier-series is defined by

$$\mathcal{F}^{\mathcal{B}_n} f := \sum_{i=0}^{n} \langle f, \phi_i^{(n)} \rangle \phi_i^{(n)}$$

Consequently, the Fourier-series of an indicator function can be written as

$$\mathcal{F}^{\mathcal{B}_n} \chi_S := \sum_{i=0}^{n} \langle \chi_S, \phi_i^{(n)} \rangle \phi_i^{(n)} = \sum_{i=0}^{n} b_i^{(n)} \chi_{B_i^{(n)}}$$

where

$$b_i^{(n)} := \frac{\mu(B_i^{(n)} \cap S)}{\mu(B_i^{(n)})}$$

are the Fourier-coefficients. The main problem, that $\mathcal{F}^{\mathcal{B}_n}\chi_S$ is not an indicator function, therefore we can not assign an $S_n \subset \mathbb{B}^d$ to it. To solve the problem the following operator will be introduced:

$$\mathcal{X}_\alpha f(x) := \begin{cases} 0 & f(x) \le \alpha \\ 1 & f(x) > \alpha \end{cases}$$

where $\alpha \in (0,1)$. Now we can define the *n*-th approximation by

$$S_n := \{\mathcal{X}_\alpha \mathcal{F}_n^{\mathcal{B}} \chi_S = 1\} =$$
$$\{x \in \mathbb{B}^d \mid \mathcal{X}_\alpha \mathcal{F}^{\mathcal{B}} \chi_S(x) = 1\}$$

Let $\mu$ be a measure on $2^{\mathbb{B}^d}$. The distance of the sets $A, B \subset \mathbb{R}^d$ can be measured by

$$\rho(A,B) := \mu(A \Delta B)$$

where $\Delta$ denotes the symmetric difference. We showed, that $\rho$ is a metric on bounded, connected, regular sets, which boundary is set of measure zero. Moreover, Lebesgue-density theorem implies that if $\mu(B_k^{(n)})$ tends to zero $(k \in (0,1,\ldots,n)$ as $n$ tends to infinity, then

$$\lim_{n\to\infty} \rho(S, S_n) = 0$$

i.e. the sequence of approximations converges to $S$. Moreover the convergence is true, if only $\mu(B_k^{(n)})$ tends to zero, where $b_k^{(n)} \in (0,1)$, i.e. atoms with Fourier-coefficients exactly $0$ or $1$ can be ignored. Let

$$\Delta_n := \{i \in \{0,1,\ldots,n\} \mid b_i^{(n)} \in (0,1)\}$$

be the set of relevant indices, and let us define the diameter of a set $B$:

$$\operatorname{diam}(B) := \sup\{\| x - y \|_2 \mid x, y \in B\}$$

Then the convergence theorem can be formulated as follows:

$$\lim_{n\to\infty} \max\{\operatorname{diam}(B_k^{(n)}) \mid k \in \Delta_n\} = 0$$

then

$$\lim_{n\to\infty} \rho(S_n, S) = 0$$

To sum up, our input is an $S \in \mathbb{B}^d$ set, we need to define a measure $\mu$ on $2^{\mathbb{B}^d}$ (or at least outer measure), and choose an initial set $B_0^{(0)} \supset S$, let $n := 0$. Then

- Choose an index $k$, where $0 < b_k^{(n)} < 1$, and divide the set $B_k^{(n)}$ into two non-empty disjoint sets, the resulting algebra is $\mathcal{B}_{n+1}$.
- We can compute $\rho(S_n, S)$ using the measure (more accurately, the Fourier-coefficients).
  If $\rho(S_n, S) \ge \varepsilon$ for some given $\varepsilon > 0$ tolerance, go back to previous step, else the process terminates.

We need to answer two important questions. How can we choose the $k$ index, and how we divide the $B_k^{(n)}$ atom. We introduce two functions, $\mathcal{C}$ for choosing and $\mathcal{D}$ for dividing. $\mathcal{C}$ is a function from possible atomic decompositions to natural numbers, $\mathcal{D}$ maps any spatial subset to a pair of spatial subsets. Accurately, the following properties are required:

$$\forall n \in \mathbb{N} \quad 0 \le \mathcal{C}(\mathcal{B}_n) \le n$$

$$\forall B \in \mathbb{B}^d \quad \mathcal{D}(B) = (B', B'')$$
$$B' \cap B'' = \varnothing, B' \cup B'' = B$$

Using our notations the general approximation schema is the following.

VolumeBasedApproximation$(S, B_0^{(0)}, \varepsilon, \mathcal{C}, \mathcal{D})$

1. $n := 0$
2. $\mathcal{B}_n := \sigma(B_0^{(n)}, \ldots, B_n^{(n)})$
3. $k := \mathcal{C}(\mathcal{B}_n)$
4. $S_n := \{\mathcal{X}_\alpha \mathcal{F}^{\mathcal{B}_n} \chi_S = 1\}$
5. **if** $\rho(S_n, S) \ge \varepsilon$ **then**
6. $\quad \mathcal{B}_{n+1} := \sigma(\ldots, B_{k-1}^{(n)}, \mathcal{D}(B_k^{(n)}), B_{k+1}^{(n)}, \ldots)$
7. $\quad n := n+1$
8. $\quad$ **goto** 3.
9. $\quad$ **else stop**

## III. Relation to space partitioning

In the followings let us consider, that the dividing operation is a splitting by a $d$-$1$ dimensional plane. This is not necessary, but in real world applications this is justifiable by the finite representation of spatial subsets. Therefore the most popular spatial decomposition methods like quadtree, octree- k-d tree or BSP tree can be obtained naturally from the general schema. The space partitioning methods mentioned before operate with hyperplanes, i.e. in every step the space will be divided into two or more disjoint half-spaces by one or more hyperplanes.
The main features and differences of the methods are discussed below, and can be found on Figure 1.

- quadtree/octree: axis-aligned planes at the center of atoms
- k-d-tree: axis-aligned planes anywhere

- BSP-tree: arbitrary plane anywhere
- VBA process: arbitrary surface anywhere

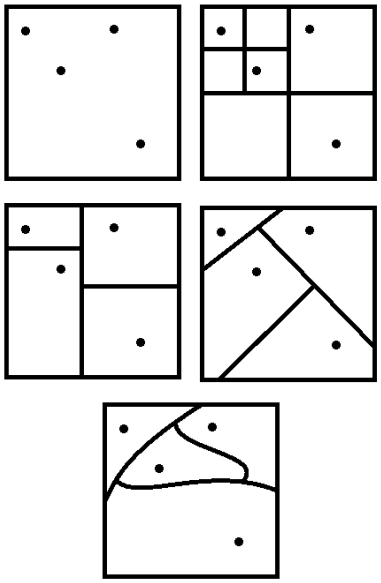

Fig. 1: Space partitioning methods from left to right from top to bottom: input points with bounding box, quadtree, k-d tree, BSP tree, VBA process.

For instance, we will present, how can we get a quadtree using our (volume based approximation) VBA process. Let $S$ be a finite subset of $\mathbb{B}^2$, and $B_0^{(0)}$ is the square of side $2$ centered at the origin. Obviously $S \subset B_0^{(0)}$. We need to define the measure. If $B_k^{(n)}$ and $S$ are disjoint sets, then $B_k^{(n)}$ is not relevant, therefore a good choice is the following:

$$\mu(B) := \begin{cases} 0 & B \cap S = \varnothing \\ \lambda(B) & otherwise \end{cases}$$

where $\lambda$ denotes the common Lebesgue-measure, in this case the area of the $B$ rectangle.

$$d_x(B) := \max\{|x' - x''| \mid (x', y), (x'', y) \in B\}$$

the diameter of $B$ in the direction of x-axis, and similarly $d_y(B)$ the diameter of $B$ in the direction of y-axis. Then let

$$J_n := \{j = 0, \ldots, n \mid d_x(B_j^{(n)}) \neq d_y(B_j^{(n)})$$

i.e. the set of indices of atoms which are not a square, and let

$$K_n := \{k = 0, \ldots, n \mid \forall j \in \Delta_n \quad \lambda(B_j^{(n)}) \leq \lambda(B_k^{(n)})\}$$

i.e. the set of indices of atoms, that have greater or equal volume, than the relevant atoms. Then

$$\mathcal{C}(\mathcal{B}_n) := \begin{cases} \min J_n & J_n \neq \varnothing \\ \min K_n & otherwise \end{cases}$$

Thereafter, let us define

$$c_x(B) := \min_x\{x \mid (x, y) \in B\} + \tfrac{1}{2} d_x(B)$$

the center of $B$ in the direction x, and similarly $c_y(B)$ the center in the direction y. Then we can define the half-space

$$H_x(B) := \{(x, y) \in B \mid \langle (x, y) - c_x(B), (1, 0) \rangle \leq 0\}$$

i.e. the points below the $1$-dimensional plane defined by its normal $(1, 0)$ and a point $c_x$. Similarly, let

$$H_y(B) := \{(x, y) \in B \mid \langle (x, y) - c_y(B), (0, 1) \rangle \leq 0\}$$

Then $\mathcal{D}$ can be defined as

$$\mathcal{D}(B) := (B', B'')$$

where

$$B' := \begin{cases} B \cap H_x(B) & d_x(B) \geq d_y(B) \\ B \cap H_y(B) & d_x(B) < d_y(B). \end{cases}$$

and

$$B'' = B \setminus B'$$

It can be seen, that the approximation process with $\mu, \mathcal{C}, \mathcal{D}$ defined above generates the quadtree of the point set of $S$.

## IV. Experimental Results

Without to claim the completeness, we show some results from our experiments. We used various choosing and dividing functions. Recall, that the output of the general approximation schema strongly dependent on these functions.

In every step of the iteration one atom is chosen, for example

- Random atom
- Maximum volume atom
- Maximum diameter atom
- Atom that containing the most vertices

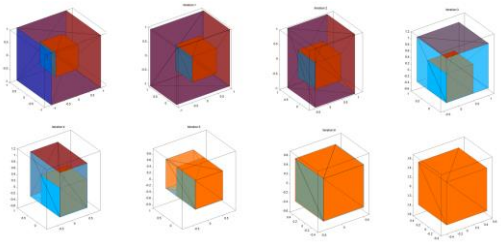

Fig. 2: Approximation of a cube in 6 steps.

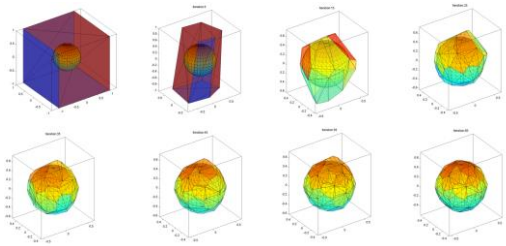

Fig. 3: Approximation of a sphere in 65 steps.

Similarly, the parameters of the splitting plane can be defined in many ways. For example

- Random plane at the centroid of the atom
- Plane lying on a triangle contained in the atom
- Plane that least coplanar to any face of the atom
- Plane that best fitting to a part of the surface

Mesh approximation can be obtained in many ways, we tested our algorithm using various parametrization on some simple objects. Good example for the mesh simplification is a slowly varying surface represented with many vertices and faces. In this case our algorithm performs well even if many faces are missing from the input mesh.

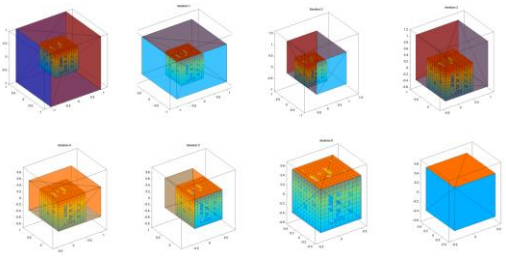

Fig. 4: Approximation of a cube with topological errors in 6 steps. The initial mesh has 10×10×10 segments (2 triangles per segment), several faces are removed from the surface.

The approximation schema with random plane splits can be used for fracture simulation of meshes, moreover it is possible to create BSP-like data structures.

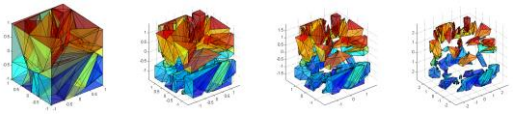

Fig. 5: Fracture simulation using random planes.

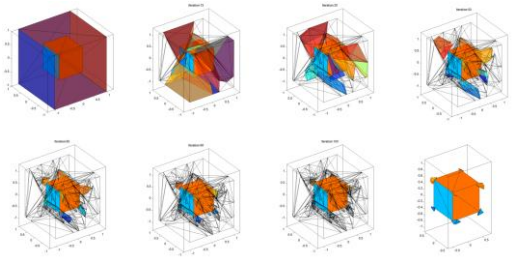

Fig. 6: Space partitioning with random planes, the input is the 8 vertices of the cube.

## V. Decomposition in Details

### A. Planning data structure

The three-dimensional decomposition data structure must contain three data arrays, namely vertices, faces and atoms. It is a similar approach as vertexstreams and indexstreams, but there are two important differences.

- the number of elements in a container can be changed dynamically
- the dimension of the elements in a container are different (except the vertices)

The second condition is necessary, because atoms are convex polyhedrons with arbitrary number of faces, and faces are convex polygons with arbitrary number of vertices. The first condition provides, that the containers can be extended. Let $X_1,\ldots,X_N$ be arbitrary non-empty sets, the set of indexed lists over the sets $\{X_i\}_{i=1}^N$ can be defined by

$$X_1\times\ldots\times X_N=\prod_{i=1}^N X_i$$

i.e. the indexed list is an ordered $N$-tuple from a cartesian product of $X_1,\ldots,X_N$ . Let us denote the lists with square brackets:

$$l\in\prod_{i=1}^N X_i \quad\Leftrightarrow$$

$$l=[x_1,\ldots,x_N] \quad (x_i\in X_i)$$

in this case obviously

$$l_i:=x_i \quad (i=[1..N])$$

where $[a..b]:=[a,b]\cap\mathbb{N}$ denotes the discrete closed interval from $a$ to $b$. We say that $x$ is the element of the list $l$, if there is an $i$ index for which $x=l_i$ , i.e.

$$x\sqsubset l \quad\Leftrightarrow\quad \exists i\in[1..N]:l_i=x$$

In particular, if $X_1=\ldots=X_N=:X$ , then we use the $X^N$ notation. We need to define an operation to extend a list. Let be $y_i\in Y_i$ $(i=1,\ldots,n)$ , then let

$$l':=[l,y_1,\ldots,y_n]\in\prod_{i=1}^N X_i\times\prod_{i=1}^n Y_i$$

be a list obtained from $l$ appending the $y_1,\ldots,y_n$ elements to it, i.e.

$$l'=[x_1,\ldots,x_N,y_1,\ldots,y_n]$$

The length of a list is defined by the number of sets in the product:

$$l\in\prod_{i=1}^N X_i \quad\Rightarrow\quad |l|:=N$$

Now we can define the decomposition data structure as an 8-tuple.

$$(M, N, K, V, l, F, L, A)$$

where

- $M$ is the total number of vertices
- $N$ is the total number of faces
- $K$ is the number of atoms
- $V \in [\mathbb{R}^3]^M$ is the list of vertices, where vertices are 3-length lists of real numbers
- $l : [1..N] \to [1..M]$ such that $l_i$ is the number of vertices of the $i$-th face
- $F \in \prod_{i=1}^{N} [\mathbb{N}^{l_i}]$ is the list of faces, where faces are defined by list of vertex indices
- $L : [1..K] \to [1..N]$ such that $L_j$ is the number of faces of the $j$-th atom
- $A \in \prod_{j=1}^{K} [\mathbb{N}^{L_j}]$ is the list of atoms, where atoms are defined by list of face indices

If we assume that the length of a list is an inner property of the list itself, we could leave the $M, N, K, l, L$ parameters, and decomposition data structure may considered as a 3-tuple formed by list of vertices, list of vertex indices of faces and list of face indices of atoms:

$$(V, F, A)$$

We give a simple example in Figure 7-8. The complete data structure can be described using the terminology of lists introduced above.

V=[ [-1, 1,-1], [-1,-1,-1], [1, -1,-1],[1, 1,-1],
[-1, 1, 1], [-1,-1, 1], [1, -1, 1],[1, 1, 1],
[0,0,0]]

F=[ [9,1,2],[9,2,3],[9,3,4],[9,4,1],
[9,6,5],[9,7,6],[9,8,7],[9,5,8],
[1,2,3,4],[8,7,6,5]]

$$A = \lfloor [1,2,3,4,9], \lfloor 5,6,7,8,10]]$$

*B. Choosing operation*

As we mentioned before, the behaviour of the VBA process is strongly depends on the choosing and dividing functions. Notice, that the choosing operation have to be the function of the $\{b_i^{(n)}\}_{i=0}^{n}$ Fourier-coefficients according to the convergence theorem (recall, in convergence theorem we used relevant atoms, that are defined through Fourier-coefficients).

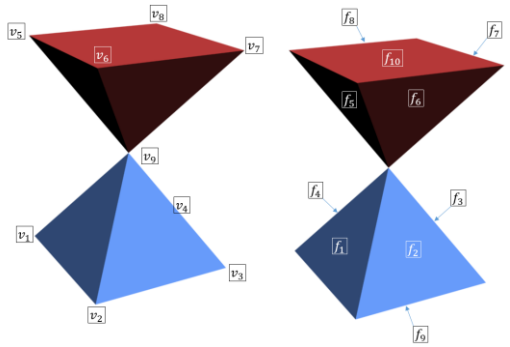


Fig. 7: Example for decomposition with 2 atoms, 10 faces, 9 vertices. Both atoms have 5 faces, 4 triangles and 1 square. The $v_9$ vertex is shared between the atoms.

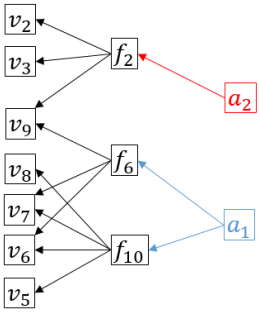


Fig. 8: Example for some elements of the decomposition data structure

Accurately, we should differentiate that if $b_i^{(n)} \in \{0,1\}$ or $b_i^{(n)} \in (0,1)$ . In general it is not enough to estimate the Fourier-coefficients, we should calculate their exact value. We assumed that the $S$ input mesh is a polyhedron, an arbitrary $B_i^{(n)}$ atom is a convex polyhedron, therefore $B_i^{(n)} \cap S$ is a polyhedron, as well. Polyhedron intersection is not an easy problem, but the topic is studied intensively. A possible algorithm can be found in [11]. If we are able to calculate the intersection of polyhedra, then calculating the Fourier-coefficients means only determining the ratio of the volumes of intersecting polyhedron and the original atom. To sum up, calculating the exact value of Fourier-coefficients can be done in two steps

- calculating the intersecting polyhedron
- calculating the volume of two polyhedra (intersecting polyhedron and containing atom.

Due to the content limits of present paper, we discuss only a volume calculation technique of polyhedra in detail.

*C. Calculating volume of polyhedra*

Let us consider an $S$ compact set of $\mathbb{R}^3$ having a piecewise smooth boundary $\partial S$ and let $\mathbf{f} : \mathbb{R}^3 \to \mathbb{R}^3$ be a continously differentiable vector field on $S$. Then Gauss-Ostrogradsky

divergence theorem states that the volume integral of the diveregence of $\mathbf{f}$ can be calculated by a surface integral of $\mathbf{f}$, namely

$$\int_S \operatorname{div}\mathbf{f}\,\mathrm{d}V = \int_{\partial S} \langle \mathbf{f}, \mathbf{n}\rangle \,\mathrm{d}A$$

where $\mathrm{d}V$ is a volume element of $S$, $\mathrm{d}A$ is a surface element of $\partial S$, $\mathbf{n}$ denotes the outward-pointing unit normal field of $\partial S$ and $\langle .,. \rangle$ is the common dot product in $\mathbb{R}^3$. In particular, if
\begin{equation}

$$\mathbf{f}(x,y,z) = \frac{1}{3}\mathbf{id}(x,y,z) := (x/3, y/3, z/3)$$

then

$$(\operatorname{div}\mathbf{f})(x,y,z) = 1/3+1/3+1/3 \equiv 1$$

therefore in this case the left hand side integral means the volume of $S$, i.e.

$$\operatorname{vol}(S) = \int_S 1\mathrm{d}V = \frac{1}{3}\int_{\partial S} \langle \mathbf{id}, \mathbf{n}\rangle \,\mathrm{d}A$$

Let $P \subset \mathbb{R}^3$ be a polyhedron defined by the faces $p_1, \ldots, p_m \subset \mathbb{R}^3$, and let us suppose, that the supporting plane of $p_i$ passes through an $\mathbf{a}_i \in \mathbb{R}^3$ point, it has a unit normal $\mathbf{n}_i$ and its $A_i$ area is known. Then using our preceding results, it is easy to prove, that the volume of $P$ can be calculated using the following formula:

$$\operatorname{vol}(P) = \frac{1}{3}\sum_{i=1}^{m}\int_{p_i} \langle \mathbf{id}, \mathbf{n}_i\rangle \,\mathrm{d}A = \frac{1}{3}\sum_{i=1}^{m} A_i \langle \mathbf{a}_i, \mathbf{n}_i\rangle$$

Notice, that the $\mathbf{a}_i$ is an arbitrary point of the $p_i$ face, eventually that is why we can change the order of integration and dot product. It can be assumed, that $\mathbf{n}_i, \mathbf{a}_i$ are known, because the faces of polyhedrons are simple polygons. A $p_i$ (simple) polygon can be defined by a directed list of its vertices, for example $[\mathbf{v}_1, \ldots, \mathbf{v}_n]$, where $\mathbf{v}_j \in \mathbb{R}^3$, moreover $\mathbf{v}_1^{(i)} = \mathbf{v}_n^{(i)}$. Then we can define the point and the normal of the supporting plane of $p_i$ as

$$\mathbf{a}_i := \mathbf{v}_1^{(i)}$$

and

$$\mathbf{w}_{jkl}^{(i)} := (\mathbf{v}_k^{(i)} - \mathbf{v}_j^{(i)}) \times (\mathbf{v}_l^{(i)} - \mathbf{v}_j^{(i)})$$

$$\mathbf{n}_i := \frac{\mathbf{w}_{jkl}^{(i)}}{\| \mathbf{w}_{jkl}^{(i)} \|}$$

where $j,k,l$ are the indices for which $\| \mathbf{w}_{jkl}^{(i)} \|$ is maximal. Consequently, if a $P$ polyhedron is defined by its vertices and polygonal faces, then we can calculate the volume of the polyhedron if we can calculate the area of its faces.

*D. Area of polygons*

There is a well-known formula to calculate the area of a planar polygon. Let be $(x_i, y_i) \in \mathbb{R}^2$ and let

$$[(x_1, y_1), \ldots, (x_n, y_n)]$$

be the directed vertex list of a two dimensional polygon $p$. Recall, that $(x_1, y_1) = (x_n, y_n)$. Then the area can be calculated by (see e.g. [9])

$$\operatorname{area}(p) = \frac{1}{2}\left| \sum_{i=1}^{n-1} x_{i+1} y_i - x_i y_{i+1} \right|$$

The absolute value can be omitted, if we know the orientation of $p$ (clockwise or counterclockwise), the formula above is right in both case. The problem is, that the polygons in our representation are three dimensional, therefore we define an affine mapping between an arbitrary (supporting) plane and $\mathbb{R}^2$. Let

$$P_{\mathbf{n},\mathbf{a}} := \{x \in \mathbb{R}^3 \mid \langle \mathbf{x}-\mathbf{a}, \mathbf{n}\rangle = 0\}$$

be the plane going through $\mathbf{a}$ and having a unit normal $\mathbf{n}$. At this point we give an orthonormed system, that fits to the plane's normal. Let
\begin{equation}

$$\omega_1(\mathbf{n}) := \begin{cases} \dfrac{1}{\sqrt{n_1^2+n_2^2}}(-n_2, n_1, 0) & n_1^2+n_2^2 \geq n_i^2+n_j^2 \\ \dfrac{1}{\sqrt{n_3^2+n_1^2}}(-n_3, 0, n_1) & n_3^2+n_1^2 \geq n_i^2+n_j^2 \\ \dfrac{1}{\sqrt{n_2^2+n_3^2}}(0, -n_3, n_2) & n_2^2+n_3^2 \geq n_i^2+n_j^2 \end{cases}$$

where $i, j \in \{1,2,3\}$ and $i \neq j$.
\begin{equation}

$$\omega_3(\mathbf{n}) = \mathbf{n}$$

$$\omega_2(\mathbf{n}) = \mathbf{n} \times \omega_1(\mathbf{n})$$

It is easy to see, that $\omega_i(\mathbf{n})$ is an orthonormed system, therefore it is a basis in $\mathbb{R}^3$. Now we define a function, that transforms the points of the $P_{\mathbf{n},\mathbf{a}}$ plane into its local coordinate system.

$$\mathcal{R}_{\mathbf{n},\mathbf{a}} : P_{\mathbf{n},\mathbf{a}} \to \mathbb{R}^2$$

$$\mathcal{R}_{\mathbf{n},\mathbf{a}}(x) := (\langle \mathbf{x}-\mathbf{a}, \omega_1(\mathbf{n})\rangle, \langle \mathbf{x}-\mathbf{a}, \omega_2(\mathbf{n})\rangle)$$

Since $\{\omega_i(\mathbf{n})\}_{i=1}^3$ is orthonormed basis, consequently every $\mathbf{x} \in \mathbb{R}^3$ can be written in the local coordinate system as

$$\mathbf{x} = \sum_{i=1}^{3} \langle \mathbf{x}-\mathbf{a}, \omega_i(\mathbf{n})\rangle \omega_i(\mathbf{n})$$

Notice, that if $\mathbf{x} \in P_{\mathbf{n},\mathbf{a}}$ then

$$\langle \mathbf{x}-\mathbf{a}, \omega_3(\mathbf{n})\rangle = \langle \mathbf{x}-\mathbf{a}, \mathbf{n}\rangle = 0$$

by definition, i.e. the third coordinate is zero for all $\mathbf{x} \in P_{\mathbf{n},\mathbf{a}}$, consequently

$$\mathbf{x} := \mathcal{R}_{\mathbf{n},\mathbf{a}}(\mathbf{x})$$

is the image of $\mathbf{x}$ in the two dimensional coordinate system attached to $P_{\mathbf{n},\mathbf{a}}$.

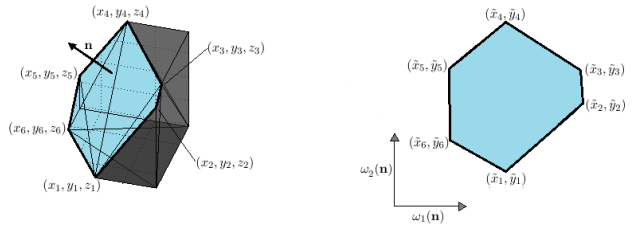


Fig. 9: It is possible to compute the volume of a polyhedron knowing its face normals, points and areas. Area of a simple polygon can be easily computed knowing the two dimensional coordinates of its points on the supporting plane.

To sum up the methods described before, if $P$ is a polyhedron defined by its vertices and face indices, we can calculate its volume in a few steps:

- calculate a normal of every face, and choose one of its vertices
- transform the vertices of a face into the local coordinate system of the supporting plane
- calculate the area of the face, using the transformed $(x, y)$ coordinates
- calculate the volume of the polyhedron, using the normals, points and areas.

### *E. Dividing operation*

As we mentioned before, when designing VBA processes the most important is to define the choosing and dividing operations. In the previous sections the subprocesses of choosing were discussed, now we give a brief overview of dividing in the decomposition data structure.

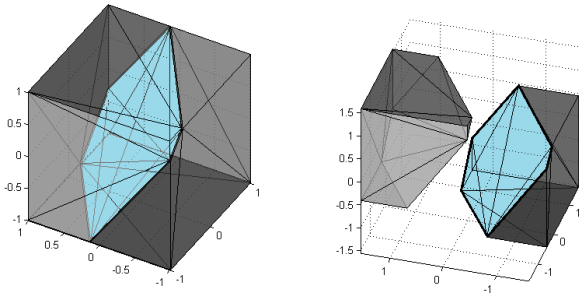

Fig. 10: An atom is divided into two non-empty atoms. The newly created face is always a convex polygon.

We assumed, that the initial atom of the decomposition is a convex polyhedron, which faces are convex polygons. Moreover, let us suppose that $\mathcal{D}$ operator is always can be interpreted as intersection with a half-space, so we have only 3+3 free parameters, a normal vector and a point of the splitting plane. These limitations significantly simplify the problem. In this context, let us note some remarks (see e.g. [2]).

- The non-empty intersection of a convex polyhedron and a half-space is a convex polyhedron.
- If $B_0^{(0)}$ convex polyhedron and $\mathcal{D}$ is defined as splitting by a plane then $B_k^{(n)}$ is convex polyhedron for all $k, n \in \mathbb{N}$.
- The non-empty intersection of a convex polygon and a half-plane is a convex polygon.
- If the faces of $B_0^{(0)}$ are convex polygons and $\mathcal{D}$ is defined as splitting by a plane then the faces of $B_k^{(n)}$ are convex polyhedron for all $k, n \in \mathbb{N}$.

As a consequence of the convexity of $B_0^{(0)}$ we can assume that the atom to be divided is convex, in this case we obtain a much simpler problem.

Let us suppose, that $p$ is a convex polygon, defined by the convex hull of its vertices

$$p = \mathrm{conv}(\{\mathbf{v}_1, \ldots, \mathbf{v}_n\})$$

where $\mathbf{v}_i \in \mathbb{R}^3$. Let $P_{\mathbf{n},\mathbf{a}}$ be a plane, and let us suppose that $\mathbf{v}_i \notin P_{\mathbf{n},\mathbf{a}}$, i.e. the vertices of the polygon do not lay on the plane. If $p \cap P_{\mathbf{n},\mathbf{a}} \neq \varnothing$ then exactly two segments are intersected, i.e. there exist exactly two indices $i, j$ such that

$$\mathrm{conv}(\{\mathbf{v}_i, \mathbf{v}_{i+1}\}) \cap P_{\mathbf{n},\mathbf{a}} \neq \varnothing$$
$$\mathrm{conv}(\{\mathbf{v}_j, \mathbf{v}_{j+1}\}) \cap P_{\mathbf{n},\mathbf{a}} \neq \varnothing.$$

i.e. there exist

$$\mathbf{x}' \in [\mathbf{v}_i, \mathbf{v}_{i+1}] \cap P_{\mathbf{n},\mathbf{a}}$$

and

$$\mathbf{x}'' \in [\mathbf{v}_j, \mathbf{v}_{j+1}] \cap P_{\mathbf{n},\mathbf{a}}$$

The intersecting points can be computed as follows

$$\mathbf{x}' := \mathbf{v}_i + \frac{\langle \mathbf{n}, \mathbf{a} - \mathbf{v}_i \rangle}{\langle \mathbf{n}, \mathbf{v}_{i+1} - \mathbf{v}_i \rangle} (\mathbf{v}_{i+1} - \mathbf{v}_i)$$

$$\mathbf{x}'' := \mathbf{v}_j + \frac{\langle \mathbf{n}, \mathbf{a} - \mathbf{v}_j \rangle}{\langle \mathbf{n}, \mathbf{v}_{j+1} - \mathbf{v}_i \rangle} (\mathbf{v}_{j+1} - \mathbf{v}_j)$$

Therefore, for a given atom the splitting operation can be done in few steps, namely: sort the faces below and above the splitting plane, collect the intersecting faces, divide the intersecting faces into two parts, find the caps of the intersecting surface. The main ideas of a polygonal mesh

splitting algorithm can be found in [7]. In details the dividing operation is the following.

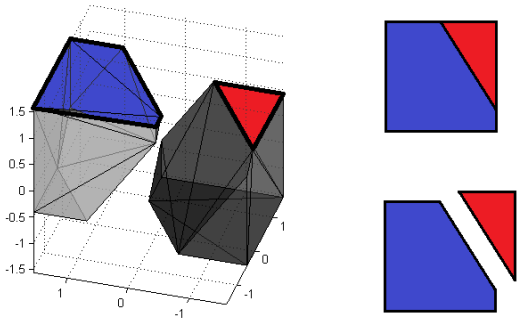

Fig. 11: An atom is divided into two non-empty atoms. The non-intersecting faces are unchanged, the intersecting faces are divided into exactly two convex polygons.

- Find all intersecting faces. Compute all intersecting points. Every intersecting face has exactly two intersecting points $\mathbf{x}', \mathbf{x}''$ .
- Every $\text{conv}(\{\mathbf{v}_1,\ldots,\mathbf{v}_n\})$ intersecting face will be divided into two non-empty faces
$$\text{conv}(\{\mathbf{x}',\mathbf{x}'',\mathbf{v}_{j+1},\ldots,\mathbf{v}_n,\mathbf{v}_1,\ldots\mathbf{v}_i\})$$
and
$$\text{conv}(\{\mathbf{x}',\mathbf{v}_{i+1},\ldots,\mathbf{v}_j,\mathbf{x}''\})$$
using the notations introduced earlier.
- Let us suppose, that all the intersecting vertices $\mathbf{x}_1,\ldots,\mathbf{x}_{2L}$ are computed $(L\in\mathbb{N})$. In this case the plane intersects $L$ pieces of faces, since every convex face adds exactly 2 intersecting vertices to the list. The cap, i.e. the new face created on the intersecting plane is defined the convex hull of the intersecting vertices, i.e.

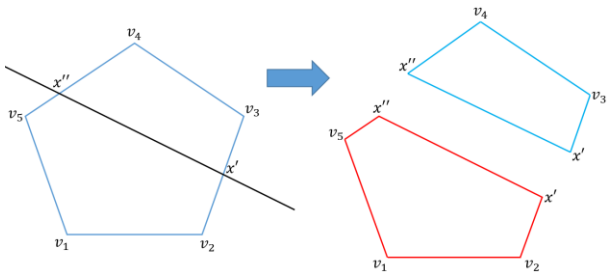

Fig. 12: Splitting an intersecting polygon by a plane

$$p_{new}=\text{conv}(\{\mathbf{x}_1,\ldots,\mathbf{x}_{2L}\})$$

After the theoretical overview let us suppose, that the $n$-th step of the VBA process we have the decomposition data structure $(V,F,A)$. Now if $\mathcal{C}(\mathcal{B}_n)=k$ then the $k$-th atom will be divided in this step. Let us consider a $P_{\mathbf{n},\mathbf{a}}$ plane, which divides the $B_k^{(n)}$ atom into two non-empty disjoint sets. Notice, that the $A_k$ element of the data structure corresponds to the $B_k^{(n)}$ , moreover as every atom is convex, $B_k^{(n)}$ equals the convex hull of the vertices corresponding to $A_k$ , i.e.

$$B_k^{(n)}=\text{conv}(\{v_j\sqsubset V \mid j\sqsubset F_i\ i\sqsubset A_k\})$$

Considering the foregoing, we can determine the $\mathcal{B}_{n+1}$ system described by the $(V',F',A')$ decomposition data structure. With our notations

$$\mathcal{B}_n\leftrightarrow(V,F,A)\Rightarrow(V',F',A')\leftrightarrow\mathcal{B}_{n+1}$$

Let us suppose, that the plane intersects exactly $L$ faces of the $k$-th atom, and the indices of the intersecting faces are

$$i_1,\ldots,i_L\sqsubset A_k$$

As every face creates exactly two new vertices, let be the coordinates of intersecting vertices $\mathbf{x}_1,\mathbf{x}_2,\ldots\mathbf{x}_{2L-1},\mathbf{x}_{2L}$ of faces $i_1,\ldots,i_L$ respectively.

Now let $\nu$ be a permutation of $\{1,\ldots,2L\}$ , such that the intersecting vertices are sorted by polar angle when $\nu$ is applied. Then we have a list of vertices

$$[\mathbf{x}_{\nu(1)},\ldots,\mathbf{x}_{\nu(2L)}]$$

wherein there are exactly two vertices at every polar angle which belong to adjacent faces. The new vertices can be obtained by taking every second vertices from the list, i.e.

$$V':=[V,\mathbf{x}_{\nu(2)},\mathbf{x}_{\nu(4)},\ldots,\mathbf{x}_{\nu(2L)}]$$

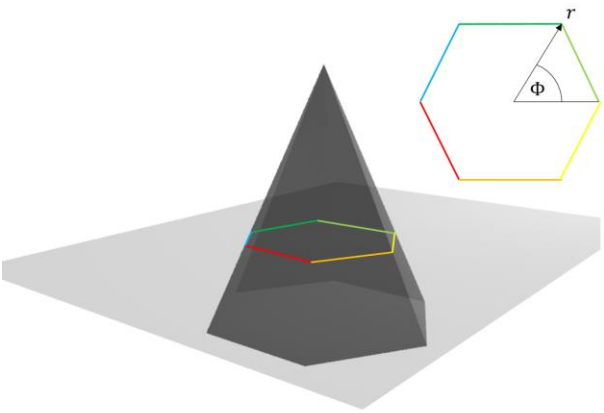

Fig. 13: Caps of the intersecting surface can be computed by sorting vertices by polar angle.

Notice, that the vertices belonging to the $i_j$ face are $\mathbf{x}_{2j-1}$ and $\mathbf{x}_{2j}$ , after sorting the vertices receive new indices $\nu(2j-1)$ and $\nu(2j)$ , leaving every second vertex and appending to the list $V$, we get, that the new indices of the intersecting vertices of $i_j$ face in the $V'$ list are

$|V|+\left|\frac{\nu(2j-1)}{2}\right|$ and $|V|+\left|\frac{\nu(2j)}{2}\right|$ where $|.|$ denotes the ceiling function. So we can determine the new faces $p_{i_j}'$ and $p_{i_j}''$ which are obtained after applying the dividing operation on the face $i_j$, where instead of

$$\mathbf{v}_m, \mathbf{x}', \mathbf{x}''$$

we use the corresponding indices:

$$\{m \mid m \sqsubset F_{i_j}\}, |V|+\left\lceil\frac{\nu(2j-1)}{2}\right\rceil, |V|+\left\lceil\frac{\nu(2j)}{2}\right\rceil$$

If we have all of the intersecting faces, we need to add them to the face list. As $F_{i_j}$ does not exist anymore, the $F_{i_j}$ elements will be replaced by $p_{i_j}'$ and $p_{i_j}''$ will be appended to the list.

$$F_{i_j} := p_{i_j'} \quad (\forall j), \quad F' := [F, p_{i_1'}, \ldots, p_{i_L''}]$$

Finally, we add the caps twice (in opposite direction too) to the list:

$$F' := [F', [|V|+1, \ldots, |V|+L], [|V|+L, \ldots, V+1]]$$

At the last step, we need to update the list of atoms. Let $i_1^+, \ldots, i_{L_1}^+$ and $i_1^-, \ldots, i_{L_2}^-$ be the face indices of the $A_k$ atom that are above and below the dividing plane, respectively (this property can be easily checked). Similarly, as the face lists we redefine the $A_k$ atom as

$$A_k := [i_1^+, \ldots, i_{L_1}^+]$$

and we append the other atom to the list

$$A' := [A, [i_1^-, \ldots, i_{L_1}^-]]$$

By this the decomposition data structure after the dividing operation is obtained.

### F. Getting results

The decomposition is refined iteratively, in every step exactly one atom is chosen and split. There are many ways to export data from the data structure. It is worth mentioning, that we can export every single atoms separately for e.g. fracture simulation, or we can export the boundary of the decomposition for e.g. mesh simplification. The boundary of the decomposition is defined by the faces of atoms, such that the face-adjacent atom is not a part of the approximation. So $f_i$ is a boundary face if and only if it is a face of $B_k^{(n)}$ and $B_l^{(n)}$, moreover $b_k^{(n)} > \alpha$, but $b_l^{(n)} \le \alpha$, see the definition of $S_n$. If it is needed the polygonal mesh trivially can be converted to a simple triangular mesh, since all the polygons are convex.

## VI. Conclusions and Further Work

We are working on the implementation of the data structure introduced in this paper to perform fast dividing operations on polyhedrons. Obviously the conversion between our data structure and the common data structures (vertexstrem-indexstream) should be possible to design and to supervise VBA processes.

Our goal was to give a theoretical foundation for some similar approaches of space partitioning. The topics mentioned are often thought to be different, but all them can be described by a simple iterative approximation process. This approximation process is based on the theory of Fourier-series in Hilbert-spaces, which is well-known topic in mathematics, there are many useful theorems, that can be used during our work. But there are not just theoretical advantages of the method, we have a convergence theorem, necessary and sufficient condition for the monotonicity, and an exact error formula. If we can calculate exactly the volume of an intersection of an atom and the input, then we can calculate the exact distance, as well. Intersection of polyhedra can be calculated, but it is a difficult problem. Our idea is to calculate the volume of intersection without producing the intersection itself. If we had the exact error formula, and the Fourier-coefficients, we could optimize the performance of the approximation process, and maybe it would be more widely used.

## References


[1] M. Campen, M. Attene, L. Kobbelt, "A Practical Guide to Polygon Mesh Repairing", *ACM Transactions on Graphics*, vol. 21, no. 2, pp. 88-105, 2002.

[2] B. Chazelle, D. P. Dobkin, "Intersection of Convex Objects in Two and Three Dimensions", *Journal of the ACM*, vol. 34, issue 1, pp. 1-27, 1987.

[3] E. W. Cheney, *Introduction to Approximation Theory*, AMS Chelsea Pub., 1982.

[4] J. L. Doob, *Measure Theory (Graduate Text in Mathematics)*, Springer, 1994.

[5] C. Ericson, *Real-Time Collision Detection*, Elsevier, 2005.

[6] G. Fábián, L. Gergó, "Adaptive algorithm for polyhedral approximation of 3D solids", *Stud. Univ. Babes-Bolyai Mathematica* vol. 60, no. 2, pp. 277–293, 2015.

[7] G. Fábián, L. Gergó, "Fast Algorithm to Split and Reconstruct Triangular Meshes", *Stud. Univ. Babes-Bolyai Informatica*, special issue 1, pp. 90-102, 2014.

[8] G. Fábián, L. Gergó, "On Constructing Volume Based Approximation Algorithms of Spatial Subsets", *Advances in Computer Science*, (Proceedings of the 6th European Conference of Computer Science, Rome, Italy, November 7-9), pp. 60-65, 2015.

[9] S. Ghali, *Introduction to Geometric Computing*, Springer-Verlag London Limited, 2008.

[10] M. Ghosh et al. "Fast approximate convex decomposition using relative concavity", *Computer-Aided Design* vol. 45, issue 2, pp. 494-504, 2013.

[11] K. Mehlhorn K. Simon, "Intersecting two polyhedra one of which is convex", *Fundamentals of Computation Theory* vol. 199 of the series Lecture Notes in Computer Science, pp. 534-542, 2005.

[12] M. Muller N. Chentanez, T. Kim, "Real Time Dynamic Fracture with Volumetric Approximate Convex Decompositions", *ACM Transactions on Graphics* vol. 32, issue 4, article no. 115, 2013.

[13] I. P. Natanson, *Constructive Function Theory*, vol. 1., Frederick Ungar Publishing Co., 1964.

[14] J. Neveu, *Discrete Parameter Martingales*, Elsevier, 1975.

[15] S. J. Owen "A Survey of Unstructured Mesh Generation Technology", *International Meshing Roundtable*, pp. 239-267, 1998.

[16] S. R. Tate, K. Xu, "General-Purpose Spatial Decomposition Algorithms: Experimental Results", *Chapman\&Hall/CRC*, 2004.

[17] M. E. Taylor, *Measure Theory and Integration*, Graduate Studies in Mathematics, vol. 76, 2006.